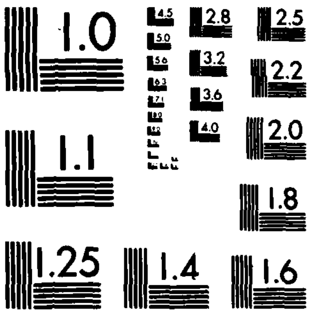
AD-A124 804 FORTRAN BASED LINEAR PROGRAMMING FOR MICROCOMPUTERS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING T R FRALEY ET AL. DEC 82

1/5

UNCLASSIFIED AFIT/GOR/OS/82D-4 F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFIT/GOR/OS/82D-4

FORTRAN BASED LINEAR PROGRAMMING FOR MICROCOMPUTERS

THESIS

AFIT/GOR/OS/82D-4

THEODORE R. E. FRALEY, MAJOR USAF

DALE A. KEM, CAPTAIN USA

AFIT/GOR/OS/82D-4

FORTRAN BASED LINEAR PROGRAMMING

FOR

MICROCOMPUTERS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Theodore R. E. Fraley
Major USAF

and

Dale A. Kem
Captain USA

Graduate Operations Research

December 1982

Approved for public release; distribution unlimited

Preface

We would like to thank our thesis advisor, Major Daniel B. Fox , for his guidance and support throughout the AFIT program and specifically during the period of our thesis research. Major Fox and our reader, Major Gerald R. Armstrong, have expended great effort and time to insure that this research was a learning experience and that a beneficial software package was produced. We would like to express our graditude to Lt/Col Ivy D. Cook for his assistance during the research.

We also wish to thank our families, especially our wives, Gail and Jean, for their support, typing and editing assistance, and, most of all, their patience during this time.

CONTENTS

List of Figures

List of Tables

Abstract

Linear programming is an analytical technique used in decision analysis. This paper describes the development and use of a highly interactive, non-programmer oriented, linear programming software package implemented on a microcomputer. This software, written in FORTRAN and supported by the UCSD Pascal Operating System, has allowed increased portability while providing the capability of solving moderate-sized LP models. Also available are extensive postoptimal sensitivity analysis capabilities.

The modularly implemented package provides interactive, instructional sessions with user input LP models. The user is guided through tableau formulation and pivot element selection to an optimal solution by a series of option displays and user selections. This module also provides instructors the ability to rapidly demonstrate the application of the simplex algorithm.

A separate module provides a more rapid problem solution with minimal interaction. Options allow either primal or dual problem solution with screen-oriented output to either a monitor or printer. The sensitivity analysis capabilities include right-hand-side, cost coefficient, and constraint coefficient ranging. Also provided is the ability to add constraints and variables to the original model.

FORTRAN BASED LINEAR PROGRAMMING
FOR
MICROCOMPUTERS

I INTRODUCTION

Managers at all levels of private and public organizations are continuously confronted with the burden of decision making and the subsequent accountability for such decisions. The criticality of these decisions may not be immediately obvious to the manager or to the organization, yet the outcome may contribute to the success or the failure of the organization.

A large fraction of these decisions involve the amount of organizational resources, such as manpower, equipment, or funds, to dedicate to a particular project or operation. Although one could attempt to dedicate the necessary resources required to maximize the output of each operation, one would soon realize a shortage in one or many of these organizational assets. *One might then attempt purely* subjective evaluations of the worth of various projects and allocate resources based upon this process. However, for high level managers of diverse organizations, this may be beyond the bounds of comprehension due to the magnitude of activities under their control.

Therefore, managers have sought methods which will allow a systematic and accurate analysis of numerous operations in a timely manner. A decision which is accurate, but late, may be of less value than an inaccurate decision which has been made in sufficient time. This search has lead to the development and implementation of several mathematical programming techniques. One important subset of these mathematical programming techniques is linear programming.

Linear Programming

Mathematical programming, which consists of several specific optimization techniques, has been defined as the use of mathematical representations (models) to plan (program) an allocation of scarce resources among competing activities. Linear programming is one such technique commonly used by analysts and is an optimization technique which involves only linear mathematical relationships (Ref 4:35). Although the term "programming" is used, in this context it does not refer to computer programming. The term in this setting refers to the selection of a particular course of action or program and is a synonym for planning.

Although mathematical optimization techniques have been present for many years, the last three decades have shown a great increased use of quantitative tools in aiding managerial decision making (Ref 22:XI). Many techniques have been developed in this new realm of application, but none so popular as linear programming. George B. Dantzig and his associates first developed and applied this technique in 1947 following a proposal that the interaction of the activities in an organization may be viewed as linear relationships (Ref 10:IX). In conjunction with this development, Dantzig also proposed the simplex algorithm which has been shown to be a systematic procedure for the solution of such linearly defined problems.

The linear programming technique, although fairly recent, was estimated to account for 25 percent of all scientific computations in 1970 (Ref 22:XVI). The extensive use of such a technique coupled with the increased use of the computer for all types of computational procedures has lead to the development of extensive software packages implementing linear programming on mainframe computer systems. These software packages are capable of quickly solving problems consisting of hundreds of variables and constraints. The manager now has the capability of performing complex linear programming computations within a matter of minutes.

Computer State-of-the-Art

Managers now have the analytical tools and computational capability to solve linear programming problems, but is the computational power accessible? As mentioned previously, extensive linear programming software packages have been developed and implemented on large mainframe computer systems. These computer systems have essentially unlimited storage capabilities and very rapid computational rates. These combined capabilities have given rise to the problem solving capabilities previously mentioned. However, availability of these large systems is somewhat limited due to the large acquisition expense and stationary support requirements. Also, access to such systems may be limited to those managers who are operating in the immediate vicinity of such systems. This is particularly true for military leaders who may be operating in remote locations yet still require quantitative decision analysis support.

Recent advances in communications links now allow the use of remote terminals and peripherals which greatly reduce the problem of computer accessibility. However, due to the increased use of computers in all aspects of management, the number of users attempting to access the computer is normally quite large. This aspect may then cause the response time of non-dedicated remote computer systems to be unacceptable in a time critical environment.

The recent explosive development of the microcomputers or "desk-top" computers may offer a solution to many of the problems associated with the large mainframe computer systems. Prior to the late 1970's, microcomputers were little more than toys, characterized by very limited memory capabilities, difficult input/output procedures, and awkward data storage facilities. From that meager start, the capabilities of microcomputers radically increased. Most business oriented microcomputers have a random access memory (RAM) of 64,000 (64K) bytes (approximately 64,000 characters) with the capability to expand to 256K RAM. The data storage medium has advanced from slow cassette tape to floppy disks to the present hard disks which can store many millions of bytes per disk.

Noteworthy advances have also been achieved in the programming languages available for use with these microcomputers. Until recently, microcomputers were usually limited to the machine specific BASIC language as the only available high-level language. Now, many microcomputers support more universal and powerful languages such as FORTRAN, PASCAL, and APL. It is the availability of these languages, based upon standardized rules, which has allowed increased portability of programs from machine to machine.

The recent microcomputer developments coupled with the even more recent language availability to these machines have surmounted the initial obstacles to the use of microcomputers for application of quantitative analysis techniques. However, the development of software has yet to be considered. The development of software for mainframe computer systems has occurred over several years. Also, software packages are readily available which allow the non-programming oriented manager access to efficient techniques applying both general and specific problem solving methods. This option is not so readily available for the more recent microcomputers. Although the development of software for the microcomputers is ever increasing, the packages often require the user to be quite knowledgable in programming in order to use the specific decision analysis aids.

Software availability is not the only difficulty that the microcomputer user will encounter. In exchange for the ease of availability, accessibility, and dedicated computational support, the user of "desk-top" computers will find a marked decrease in memory capabilities and computation rates. The limited memory capabilities greatly reduces the problems which may be attempted. The marked decrease in computational speed will considerably increase the length of time required to obtain results.

The above stated problems are not insurmountable, yet are serious limitations imposed by the use of microcomputers. The most serious problem is the critically limited, if existent, availability of user-oriented, portable software for the microcomputers. This problem renders the recent microcomputers virtually useless for the managers who have insufficient background and, possibly even more critical, insufficient time to formulate and implement a decision making algorithm when needed. In order for the advantages of the microcomputer to be extended to a larger percentage of the decision makers, software development is required of the various mathematical modeling algorithms, and in particular, the intensely used linear programming algorithm. Although the microcomputer can not replace the large mainframe computer systems, it may prove to be a supplement in areas of moderately sized problems and greatly aid in a more rapid response to less complex problems.

Current Microcomputer LP Software Development

A literature search conducted in June, 1982 revealed only one non-proprietary microcomputer linear programming software package documented. This analysis package, developed by Robert D. Conte (Ref 6), consists of several analysis techniques, including linear programming, implemented on an Apple II microcomputer in its machine specific language Applesoft. This interactive package has been well designed and implemented with true consideration for the non-programmer oriented user. Although the linear programming portion is capable of solving problems consisting of twenty constraints and twenty variables, true sensitivity analysis was not available. However, due to its extensive editing features, one may respecify various parameters and resolve the problem to arrive at equivalent sensitivity analysis results.

Four microcomputer based linear programming software developments were recently discussed and displayed at the

TIMS/ORSA meeting in Detroit, Michigan during April, 1982. The first which will be discussed was developed by Ralph W. Swain (Ref 20). Its primary purpose was that of graphical demonstration of techniques commonly utilized in operations research. Although a great aid in demonstrating the behavior of systems, its use to the manager and analyst is somewhat limited.

Rolf A. Daininger (Ref 7) has developed and implemented an instructional aid which will display the various iteration's tableaus for a maximum of nine constraints and twenty variables. Implemented on an Apple II microcomputer in Applesoft, it has proven to be a great aid in allowing students to concentrate on the simplex algorithm methodology and solution process rather than the numeric operations involved.

Gary E. Whitehouse and Yassar A. Hosni (Ref 21) presented an extensive software package consisting of forty-two small problem oriented programs. Several of these programs directly or indirectly involved linear programming. Examples are the application of the simplex algorithm to an LP problem and graphical solution of a two variable LP problem. An advantage of these programs is that even though written in BASIC, versions are available for both the Apple II and the TRS-80 microcomputers. Although each version is not portable between these or other systems, a larger potential set of users have access to such software.

The last current development in LP software was presented by Byron Gottfried (Ref 12). This package, as implemented on the Apple II microcomputer, is capable of solving problems of approximately forty-five constraints and ninety variables (after the augmented basis has been implemented). Another version has been implemented on an IBM microcomputer which has larger capabilities and this version includes limited sensitivity analysis. The sensitivity analysis included is right-hand-side and cost-coefficient ranging within the present feasible solution. Both versions were developed in their respective machine specific BASIC language and are currently not portable to other machines or between the two target systems.

The linear programming software presently found to exist for microcomputers has been implemented in the respective machine specific BASIC languages. Although each package individually is of significant value, each has its limitations in both significance and applicability. It would be advantageous to construct a single package which implements many of those already implemented plus expands the capabilities in many areas. Particular emphasis may be desired in the area of instructional aids designed for use by both instructors and students. Although the work of Daininger (Ref 7) has allowed the instructor to more easily demonstrate the computations of the simplex algorithm, little has been done in the area of software development for independent student use. Such software could allow the student with minimal linear programming background to reinforce the application of a linear programming solution technique to an LP problem. Also, if this implementation was in a high-level language which was more portable, it would enhance such a development even more.

Motivation for Further Research

To insure a tool is utilized to its potential, it must be developed with the user needs as a primary consideration. Also, the availability and accessibility of such a tool must be maximized for users to consider its use beneficial. Linear programming is no exception.

The problem addressed in this research was the development and implementation of a linear programming software package which allows the user extensive problem solving capabilities of small LP problems on a microcomputer system. Although the package was planned for use by analysts requiring responsive dedicated decision analysis support, features may be incorporated which will allow students and instructors of linear programming to be beneficial users. The package was developed with ease of user interface and minimum programming experience as primary considerations as well as the desire for maximum portability between available microcomputer systems. These objectives have lead to a modular package design with the requirement for user interaction being dependent upon user desires. The aforementioned goals must be balanced in light of the limitations as well as the advantages offered by a dedicated microcomputer system.

II Theoretical and Mathematical Background

Linear programming, as has been mentioned previously, is a very powerful optimization technique commonly used by today's leaders and managers. Although the subject of linear programming is found in numerous text and reference books which a manager may review, each approach the subject in different manners and elaborate to different levels of detail. Some discuss the theoretical development and background, others the methodology, and yet others focus primarily on the application of the optimization algorithms to specific type problems. This wide spectrum of literature may cause an aspiring manager to misinterpret the true power and validity of these techniques if an overview of the subject can not be captured.

The purpose of this chapter is to provide the reader an insight into the theoretical development of linear programming, with emphasis on the simplex algorithm. This theoretical background will be presented in conjunction with the simplex algorithm methodology in hopes of assisting the reader in gaining a more thorough understanding of the simplex algorithm and its application to problem solving.

For our purposes, the LP model to be discussed will be as shown below in EQ(1) through EQ(3). The dimension of m represents the number of functional constraints, excluding nonnegativity constraints, in EQ(2). The dimension of n represents the number of variables in the original problem including the slack variables required to transform the constraints into the equality form as shown in EQ(2) below. Therefore, the LP model is:

$$\text{maximize } z = \underline{C}\underline{X} \tag{1}$$

Subject to

$$A\underline{X} = \underline{B} \tag{2}$$

$$\underline{X} \geq \underline{0} \tag{3}$$

where

- z = scalar value of objective function
- $\underline{C}$ = row vector of dimension n
- $\underline{X}$ = column vector of dimension n
- A = m x n matrix
- $\underline{B}$ = column vector of dimension m
- $\underline{0}$ = n dimensional null vector

A few definitions will be presented to provide a basis for further discussion. First, a feasible solution to an LP problem is an n dimensional vector $\underline{X}$ which satisfies EQ(2) and (3) above. Therefore, each element of the vector is nonnegative and provides a solution to EQ(2). A basic solution is also a n dimensional vector $\underline{X}$ which satisfies EQ(2); however, a maximum of m elements of this vector are nonzero elements. A basic feasible solution is a basic solution which also satisfies EQ(3). Therefore, a basic feasible solution contains a maximum of m elements (called the basic variables) which are nonnegative with the remaining (n-m) elements (called nonbasic variables) having a value of zero. A basic feasible solution which contains fewer than m nonzero elements is called a degenerate solution. An optimal solution is a basic solution which also maximizes the value of z in EQ(1). If the optimal solution is also feasible, that is, EQ(3) is satisfied, then the solution is an optimal feasible solution. Otherwise, the solution is optimal but infeasible (superoptimal) for the LP problem as stated.

To introduce the simplex method, one may want to first review the geometric considerations of the problem. As the problem is stated, there are m functional constraints, represented by EQ(2), which may or may not be redundant. Also, n nonnegativity constraints are imposed by the problem. Considering a two dimensional space (n=2) and three constraints (m=3), one might find a graphical depiction of a problem as shown in Figure 1.

From the graph and the constraints shown below it, it should be recognized that the shaded area is the solution space of this problem (Note: the nonnegativity constraints are enforced in the graphical depiction). This solution space and the solution space for all LP problems forms a convex set, and therefore the convex combination of any two points in the solution set is also in the solution set (Ref 10:50). This solution set is bounded by a finite number of linear constraints which further implies that there are a finite number of intersection points of these constraints. It has been further shown that any point in a non-null

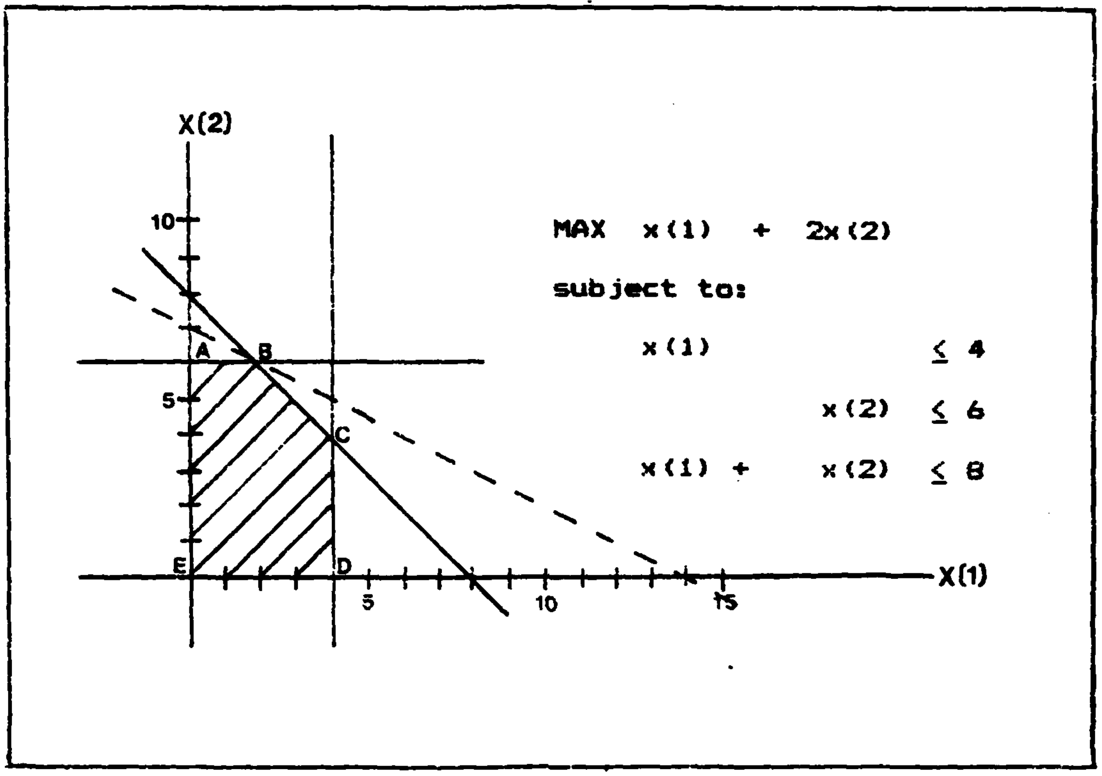


Figure 1. Graphical Solution

convex set may be represented by a convex combination of the extreme points [annotated by A, B, C, D, and E in Figure 1] of the convex set (Ref 10:29). The above discussion implies that of the infinite possible solutions to an LP problem, all may be represented by a convex combination of a finite number of solution space extreme points.

Assume that a solution set to an LP problem exists. Also assume that this solution set consists of an infinite number of points and that each solution in this set may be represented by an n dimensional vector $\underline{X}$. It has been proven that a point in n space, which includes our solution set, may be represented by the interaction of m linearly independent vectors, where $m \leq n$ (Ref 19:62-71). As a result, any point represented by m linearly independent n dimensional vectors will contain at most m nonzero elements and at least (n-m) zero elements.

It may be shown that the objective function, EQ(1), assumes its optimal value at an extreme point of the convex set or, if at more than one, the objective value, z, is the same for all convex combinations of these extreme points (Ref 10:50-51). Therefore, only the extreme points of the convex set must be investigated in the search for an optimal solution. The number of extreme points, although possibly large, is finite and greatly reduces the number of points which require investigation to determine the optimal solution. If a set of m linearly independent vectors may be found, with the solution vector containing at most m nonnegative elements and at least (n-m) zero elements, the solution corresponds to an extreme point of the convex set (Ref 10:53).

The above implies that only the extreme points generated by m linearly independent vectors must be investigated in the search for an optimal solution. Consider now the A matrix in EQ(2) and envision each column of the matrix as an m dimensional vector $\underline{V}$. Although m linearly independent $\underline{V}$ vectors may not be readily identifiable in the problem initially, the matrix may be augumented by a set of m linearly independent vectors to provide this set of m linearly independent vectors. With this m dimensional basis, it is known that, at most, [n!/m!(n-m)!] possible solutions exist and may require investigation since this is the number of combinations of m vectors from a set of n vectors (Ref 9:31).

Up to this point, it has been shown that of the infinite number of solutions which may exist, at most [n!/m!(n-m)!] require investigation. But now it must be asked, how are these extreme points determined? Again envision the A matrix consisting of n m-dimensional vectors $\underline{V}(1)$ through $\underline{V}(n)$. Assume that the first m vectors are linearly independent and that $\underline{X}$ is a basic feasi .e solution. In this form, EQ(2) may be expressed as follows:

$$xb(1)*\underline{V}(1) + xb(2)*\underline{V}(2) +...+ xb(n)*\underline{V}(n)= \underline{B} \qquad (4)$$

where

xb(i) are the elements of the basic feasible solution $\underline{X}$

$xb(i)\geq 0 \quad i=1,...,n$

It has been assumed that $\underline{V}(1)$ through $\underline{V}(m)$ are linearly independent. It will be further assumed that a nonnegative combination of these vectors equals the vector $\underline{B}$. Therefore, EQ(4) may be now expressed as shown in EQ(5) below. Note that the elements of $\underline{X}$ from xb(m+1) through xb(n) are equal to zero due to the linear independence of $\underline{V}(1)$ through $\underline{V}(m)$.

$$xb(1)*\underline{V}(1) + xb(2)*\underline{V}(2) +...+ xb(m)*\underline{V}(m) = \underline{B} \quad (5)$$

It is known that any of the n vectors may be represented as a linear combination of the basis vectors. Therefore a vector $\underline{V}(k)$, where $k > m$, may be represented as follows:

$$xk(1)*\underline{V}(1) + xk(2)*\underline{V}(2)... + xk(m)*\underline{V}(m) = V(k) \quad (6)$$

where

xk(i) represents the weight of the i(th) vector in the linear combination forming $\underline{V}(k)$ when $\underline{V}(k)$ is the selected entering basis vector

To determine a new solution vector $\underline{X}$ to EQ(5) which includes at most (m+1) nonzero elements, we may multiply EQ(6) by some value, say T, then subtract it from EQ(5) to find:

$$[xb(1)-T*xk(1)]*\underline{V}(1) + [xb(2)-T*xk(2)]*\underline{V}(2) ... + [xb(m)-T*xb(m)]*\underline{V}(m) = \underline{B}-T*\underline{V}(k) \quad (7)$$

or

$$[xb(1)-T*xk(1)]*\underline{V}(1) + [xb(2)-T*xk(2)]*\underline{V}(2)... + [xb(m)-T*xk(m)]*\underline{V}(m) + T*\underline{V}(k) = \underline{B} \quad (8)$$

(assume one $xk(i) \geq 0$ for $i=1,...m$)

The solution vector $\underline{X}$ is n dimensional, but now contains at most (m+1) nonzero elements. It must now be noted that only those vectors with no more than m nonnegative elements are desired and are possible basic feasible solutions. Therefore, the solution vector for EQ(8) must contain no more than m nonnegative elements to represent an extreme point of the convex set. The problem at this point is to determine the value of T which will force one of the elements in the EQ(8) solution vector to zero, thereby forcing one of the previous vectors [V(1) through V(m)] out of the basis.

With the above insight, it may be determined that if the multiplier T is positive, only those values of xk(i) which are positive need be checked. If the value of xk(i) was nonpositive, the solution element [xb(i)-T*xk(i)] would always be positive and would not approach zero. To determine which element of EQ(8)'s solution will be forced to exactly zero, it must also be considered that all other elements which are nonnegative must remain nonnegative in the new solution. Therefore we want to find that element which first reduces to zero. For i=1,...m, the element which first achieves

$$xb(i)-T*xk(i) = 0 \tag{9}$$

is the coefficient of the vector which will be forced out of the basis. This may be formulated to be:

$$T = \min_{i=1}^{m}[xb(i)/xk(i)] \tag{10}$$

Using the above value for T, one may determine the new basic solution corresponding to another extreme point of the convex set.

One should recognize that if a vector is selected to enter the basis, [$\underline{V}$(k) in this example] and it is found that all xk(i) are less than zero, this basis will contain m+1 elements. Since the new solution can not be expressed as a basic solution, that is, m nonnegative elements in solution vector, it does not correspond to an extreme point and therefore is not a basic solution. This situation indicates that the problem has no finite maximum solution (unbounded) and the solution process is terminated.

From the previous discussion, one may find all extreme point solutions by enumeration and evaluate each of these solutions in terms of EQ(1) to determine which basic feasible solution produces the maximum objective value z. Although the above process could, in theory, be performed, the number of calculations increases exponentially as the number of variables (n) and constraints (m) increase. The discussion of the simplex algorithm will show that once an initial basic feasible solution is found, an optimal solution may then be found, if it exists, in a finite number of steps. The simplex algorithm allows the user to find only those basic feasible solutions which have an objective function of equal or greater value than the present solution. Also, the algorithm identifies for the user when the optimal solution has been obtained or that a optimal solution does not exist (known as unbounded solution).

To illustrate the methodology of the simplex algorithm in light of the theoretical background, consider the numerical example given earlier. If one would express this problem in the stated form, each constraint would have a slack variable added to form an augmented basis as shown in EQ(12) through EQ(14). The initial basis feasible solution is x(3)=4, x(4)=6, and x(5)=8 since these vectors provide the basis. EQ(15) states the equivalent mathematical relationship of EQ(12) through EQ(14) in the form presented earlier [EQ(5)].

$$\text{Maximize } z = x(1) + 2x(2) + 0x(3) + 0x(4) + 0x(5) = 0 \qquad (11)$$

Subject to

$$x(1) + x(3) = 4 \qquad (12)$$

$$x(2) + x(4) = 6 \qquad (13)$$

$$x(1) + x(2) + x(5) = 8 \qquad (14)$$

$$xb(3)*\underline{V}(3) + xb(4)*\underline{V}(4) + xb(5)*\underline{V}(5) = \underline{B} \qquad (15)$$

Now if each basic variable is expressed in terms of only the non-basic variables, x(1) and x(2), the following is found:

$$x(3) = 4 - x(1) \qquad (16)$$

$$x(4) = 6 - x(2) \qquad (17)$$

$$x(5) = 8 - x(1) - x(2) \qquad (18)$$

The objective function may then be expressed in terms of the nonbasic variables to arrive at:

$$z = 0 + x(1) + 2x(2) \qquad (19)$$

At this point, review the previous section's thoughts. A basic feasible solution exists; $\underline{X}$= (0, 0, 4, 6, 8) and the basis is formed by m=3 linearly independent vectors [$\underline{V}$(3), $\underline{V}$(4), $\underline{V}$(5)]. At this point, we would select a vector not in the basis [$\underline{V}$(1) or $\underline{V}$(2)] to determine whether this new solution is also a basic feasible solution. Previously, no method has been discussed to select the incoming basis vector. Each nonbasis vector could have been selected to enter; however, random vector selection may cause the value of the objective function to decrease and move away from its optimal value. The simplex algorithm assists in this selection in that it guides the user to select an incoming basis vector which will increase, or at least maintain, the current objective function value. Looking at EQ(19), one will find that the current objective function value is zero since both x(1) and x(2) are currently nonbasic variables. To increase the objective function, either $\underline{V}$(1) or $\underline{V}$(2) may be selected to enter the basis since both have positive coefficients in the objective function. The objective function value, z, will increase as the value of the incoming variable increases since it has been stated that the variables [x(1), x(2)] must be nonnegative. However, if $\underline{V}$(2) is selected, the objective function value will increase at a rate or slope of two while $\underline{V}(1)$ would only increase the objective value at a rate of one. Therefore, the simplex algorithm will guide the user to select as the entering basis vector, that vector which will cause the most rapid increase in the objective function. One could at this time perform the computations of EQ(6) through EQ(10) to determine the new basis. These calculations will not be performed at this time but will be shown later using the simplex algorithm from beginning to end for the example problem.

Now that the logic for selecting an entering basis vector has been displayed, the theoretical development will be reviewed. Recall that the constraints of the problem were given in EQ(5). The objective function may be then expressed as follows:

$$xb(1)*c(1) + xb(2)*c(2) + \ldots + xb(m)*c(m) = z \qquad (20)$$

where

c(i) for i=1,..,n are the cost coefficients of the objective function.

Also recall that $\underline{X}$ has been assumed to be a basic feasible solution and that any vector $\underline{V}(n)$ may be expressed as a linear combination of the basis vectors, $\underline{V}(1)$ through $\underline{V}(m)$. At this point, let us define a term z(j) as

$$z(j) = \sum_{i=1}^{m} a(i,j)*c(i) \qquad j=1,\ldots.n \qquad (21)$$

where

a(i,j) are the i(th) coefficients of the j(th) vector $\underline{V}(j)$

c(i) represents the cost coefficient of the basic variable of row i.

The element z(j) for vector j [$\underline{V}$(j)] could be enumerated as:

$$z(j) = a(1,j)*c(1) + a(2,j)*c(2) + \ldots + a(m,j)*c(m) \quad (22)$$

The z(j) element has been defined by Hillier & Lieberman (Ref 13:88) as the net amount by which the initial coefficients in the objective function have been increased by the simplex method.

For a fixed value of j, if z(j)-c(j)<0, a feasible solution exists in which the new value of z is greater than or equal to the current value of z (Ref 10:66-67). The proof of this was shown by multiplying EQ(6) by some value, T in our example, and subtracting this from EQ(5). The results of this are shown in EQ(8). Also as part of this proof, EQ(22) was multiplied by T and subtracted from EQ(20). The results are shown below.

$$[xb(1) - T*a(1,k)]*c(1) + [xb(2) - T*a(2,k)]*c(2) + \ldots$$
$$+[xb(m) - T*a(m,k)]*c(m) + T*c(k) = z - T*[z(k) - c(k)] \quad (23)$$

Note that T*c(k) has been added to both sides of EQ(23).

EQ(23) represents the objective function of a feasible solution, assuming the coefficients of $\underline{V}$(1)...$\underline{V}$(m), $\underline{V}$(k) are positive. Also note that if a [z(k)-c(k)] exists which is negative and assuming that T is positive, the right hand side of EQ(23) will increase in value beyond z, the previous objective function value. This means that a feasible solution exists which possesses a higher objective value and that the simplex procedure requires further iterations to obtain optimality. In order to increase the objective function at the greatest rate, the simplex algorithm directs the user to select as the entering basis vector that vector which has the largest negative [z(k)-c(k)] value.

If a negative [z(k)-c(k)] does not exist for a problem, the current basic feasible solution is optimal. This condition then allows the simplex algorithm to be terminated (Ref 10:67).

For purposes of illustration, assume that a negative [z(k)-c(k)] exists and therefore an entering basis vector (basic variable) may be identified. At this point, one would determine the leaving basis vector by calculating the multiplier T as in EQ(10). Note that in EQ(8) and EQ(10), xk(i) is equivalent to the a(i,k) in EQ(21),(22), and (23) where k represents the vector V̲(k), the selected entering basis vector. Also remember that only those a(i,k) [or xk(i)] coefficients which are positive need to be checked since a basic feasible solution is being sought. Once the value of T is determined, one would solve EQ(21) and (22) for the new objective value and EQ(8) for the basic variable values. However, the a(i,k) values [or xk(i)] of EQ(8) have not yet been determined, so one further step is required.

Assume that vector k [V̲(k)] is the entering basis vector [largest negative z(j)−c(j)]. EQ(6) shows that $\underline{V}(k)$ may be expressed as a linear combination of the basis vectors [$\underline{V}(1)$ through $\underline{V}(m)$]. Also assume that $\underline{V}(l)$ has been found to the leaving basis vector ([T = min (xb(i)/a(i,k))] where i has been found to be l). One may then express $\underline{V}(l)$ as [from EQ(6)]:

$$\underline{V}(l) = [1/a(i,k)]*[\underline{V}(l) - \sum_{i=1}^{m} xk(i)*\underline{V}(i)] \qquad (24)$$

where

$i=1,2,...,l-1,l+1,..,m$

EQ(24) may then be substituted into EQ(5) to arrive at EQ(25).

$$\begin{aligned} &\left(xb(1) - [xb(l)/a(l,k)]*a(1,k)\right) *V(1) + \ldots \\ &+ \left([xb(l)/a(l,k)]\right)*V(k) + \ldots \\ &+ \left(xk(m) - [xb(l)/a(l,k)]*a(m,k)\right) *V(m) = B \end{aligned} \qquad (25)$$

The new solution $\underline{X}$ is then found to include the basic variables x(1), x(2), .. x(l−1), x(l+1), x(m) and x(k).

The above process may be performed, however the time required would be excessive for any significant number of basis changes. The simplex method shortens this process considerably by constructing a "tableau" which contains only the coefficients of the objective function and constraints in a form which greatly simplifies the above manipulations. Although different references form this tableau in slightly different manners, all perform the same function. The form for this review will be as shown in Table I.

TABLE I

Tableau Form Of Simplex Algorithm

	z	x(1)	x(2)	x(m)	x(m+1)	x(n)	RHS
obj fun	1	z(1)-c(j)	z(2)-c(2)	z(m)-c(m)	z(m+1)	z(n)	0
1	0	a(1,1)	a(1,2)	a(1,m)	a(1,m+1)	a(1,n)	b(1)
2	0	a(2,1)	.	.	.	.	.
3	0	a(3,1)	.	.	.	.	.
i	0	.	.	.	.	.	.
m	0	a(m,1)	a(m,2)	a(m,m)	.	.	b(m)

To place the objective function in the proper form for the tableau, it must be in a maximize z-[c(j)x(j)] form for the c(j)'s to be in the -c(j) form. For our numerical example, this would correspond to z - x(1) -2x(2) = 0. Had the problem been a minimization problem, one would simply multiply the entire objective function by -1 and this would then represent an equivalent maximization problem (Ref 10:78). The constraints must be first converted to equalities which may be done by adding a "slack" variable to each constraint (assume that the inequalities are less-than inequalities for the moment). It should be noted that these slack variables form the initial basis of this problem. With the above modifications, the initial tableau would be as shown in Table II.

TABLE II

Initial Basic Solution

z	x(1)	x(2)	x(3)	x(4)	x(5)	RHS
1	-1	-2	0	0	0	0
0	1	0	1	0	0	4
0	0	1	0	1	0	6
0	1	1	0	0	1	8

If the material which has been discussed is now applied, one would first determine the entering basis vector or basic variable. Only two possibilities exist for entering variables and the simplex method directs the selection of that variable with the largest negative [z(j)-c(j)]. Variable x(2) would then be selected for the entering variable. Next, one would determine the value of T, the multiplier which will force one of the present basic variables [x(3), x(4), and x(5)] to exactly zero while maintaining the other basic variables at a nonnegative level. The values of T which are found by EQ(9) are:

i=1 T = 4/0 = undefined

i=2 T = 6/1 = 6 (minimum)

i=3 T = 8/1 = 8

Therefore, T would equal 6 in this iteration and identifies the basic variable of row 2, x(4), as the leaving basic variable. In the notation of this review, l is the leaving basic variable of row 2 while k is the entering basic variable, column 4. Each element of the tableau may be transformed to that corresponding to the new basis using the following formulas (Note: x(i,j) represents any element of the tableau):

$$x'(i,j) = x(i,j) - [x(i,j)/x(l,k)]x(i,k) \qquad i \neq l \qquad (26)$$

$$x'(i,j) = x(i,j)/x(l,k) \qquad i = l \qquad (27)$$

The above formulas are simplifications of EQ(25) and are applicable to all rows and columns of the tableau (Ref 10:74).

If one applies the above, one will find a new tableau as is depicted in Table III. One would see that a negative [z(j)-c(j)] exists [x(1)] so an optimal solution has not yet been obtained. Therefore one would select x(1) as the entering basic variable. Performing this iteration, one would find that shown in Table IV.

One would examine this tableau and find that no negative [z(j)-c(j)] values exist which indicates an optimal solution. The solution vector $\underline{X}=(2,6,2,0,0)$ has all nonnegative elements indicating that it is also feasible. One should also check for degeneracy, which means that fewer than m elements of the basic feasible solution are nonzero. Since the basis dimension (m=3) equals the number of nonzero

TABLE III

Second Basic Solution

z	x(1)	x(2)	x(3)	x(4)	x(5)	RHS
1	-1	0	0	2	0	12
0	1	0	1	0	0	4
0	0	1	0	1	0	6
0	1	0	0	-1	1	2

TABLE IV

Third Basic Solution

z	x(1)	x(2)	x(3)	x(4)	x(5)	RHS
1	0	0	0	1	1	14
0	0	0	1	1	-1	2
0	0	1	0	1	0	6
0	1	0	0	-1	1	2

variables, the solution is also nondegenerate. One would note that the same solution was found earlier in the chapter by the graphical method.

If the graph is inspected, it will be found that we initially started at the extreme point labeled E [x(1)=x(2)=0] in the first tableau. Next, x(2) entered the basis and we moved to point A [x(1)=0, x(2)=6]. The final tableau corresponds to point B [x(1)=2, x(2)=6]. Had we not used the selection rule of the largest negative [z(j)-c(j)] for the entering basis vector, we could have selected x(1) as the first entering basis vector. This pivot would have moved us from point E to point D, and then next to point C, and finally to point B. Although the simplex selection rule does not always cause fewer pivots to be performed, this is an example of it doing so.

One area which has been passed over is that of unboundedness. Problems may arise which have no optimal solution since a basic variable or variables may be increased indefinitely without forcing another vector out of the basis. This occurs when a negative [z(k)-c(k)] exists but all a(i,k)<0 for i=1,....,m. This means that T may be made arbitrarily large, the basis is m+1 and the objective function increases without bound. If EQ(23) is examined with T being large and a(i,k)<0, it can be seen that this occurance is possible. In a practical sense, this means that the model has been formulated incorrectly and when this situation occurs, the simplex algorithm is terminated.

The discussion up to this point has assumed an initial basis was present or the problem was stated as $A\underline{X}\leq\underline{B}$ where slack variables will form the basis. The initial problem may be stated as $A\underline{X}=\underline{B}$ with an initial basis not readily identifiable. A technique which is used in this case is called an "artificial basis" technique. Each constraint has a unique basis variable added and also each variable is assigned an unspecified large negative number (often called "M") as a cost coefficient. It has been proven that if a feasible solution exists to the original problem, one will also exist for the augmented problem. Also if a feasible solution does not exist for the original problem, the optimal solution to the augmented problem will contain an artificial variable at a positive level (Ref 10:81).

This technique, although powerful, may increase the number of iterations required to obtain optimality. The artificial variables must be driven from the basis prior to determining optimality and therfore should be used only when required. If a basis vector exists in the problem as given, use this as an initial basis vector to minimize the iterations required.

The simplex algorithm is applied as discussed previously to the artifical basis with the exception of selecting the entering variable. Since the [z(j)-c(j)] values may now contain a unspecified large value "M", the selection of the largest negative [z(j)-c(j)] must consider two elements. All [z(j)-c(j)] values which contain a "M" must be examined and the selection differentiator is the numeric element. That [z(j)-c(j)] with an "M" value and the largest negative [z(j)-c(j)] should be selected as the entering basic variable. Once the "M" values has been removed from the objective row of the tableau, the algorithm proceeds as before.

Sensitivity analysis is based on the relationship between the coefficients of the original columns of the linear programmming model and the columns representing the slack and artificial variables.

Considering a problem with K constraints and V variables, the final K columns will initially form an identity matrix. As the simplex algorithm is performed, the identity matrix undergoes a transformation. This matrix is, upon completion of the algorithm, a record of all operations performed on the original equation. It is possible to determine from this record the change to any element of the final tableau which results from one or more changes to the original tableau.

To illustrate these changes, consider an original problem with three variables and four constraints. If a change was made to the original (3,2) element, the changes to the final tableau could be found by pre-multiplying the matrix with the single change by the transformed identity matrix which is commonly called B-inverse.

As can be seen in Figure 2, a change in the third row, second column will produce changes in the entire second column of the final tableau. Also, note that only the values in the third column of B-inverse were pertinent. This is because the third column of the identity matrix or B-inverse is associated with the third row of the original

$$\begin{bmatrix} V & V & W & V \\ V & V & X & V \\ V & V & Y & V \\ V & V & Z & V \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \Delta W & 0 \\ 0 & \Delta X & 0 \\ 0 & \Delta Y & 0 \\ 0 & \Delta Z & 0 \end{bmatrix}$$

B-inverse

Figure 2. Effect of Single Change to the Original Tableau

tableau. Although not shown in the illustration, the coefficient of the objective function above the third column of the B-inverse matrix, when multiplied by the change to the (3,2) element, will give the change to the coefficient in the objective function above the second column of the original matrix.

When the results of the changes have been determined and added to the respective elements of the final tableau, further manipulations may be necessary. If a change occurred to a column which was in the basis of the final tableau, that column will have to be returned to its final tableau form. This is done by: first, dividing the entire row which had a value of one in the column under investigation by the new value to reestablish the value of one, and second, adding multiples of that row to each of the other rows and the objective function to return all other values in the column to zero. If the value of any right-hand side or objective function coefficient has become negative, the tableau must be resolved using the LP algorithm.

Determining the range limits of the elements of the original tableau is merely a variation of the method previously described. Rather than solving for the changes which result from a change to an original element, the algorithm individually investigates each element to determine the smallest positive and negative changes which would cause either a multiple optimal (an objective function coefficient not in the basis goes to zero) or a degenerate (right-hand side goes to zero) condition.

		ΔC				C	
X2	0	ΔW	0	0	0	W	0
X1	0	ΔX	0	0	0	X	0
X5	0	ΔY	0	0	0	Y	0
X7	0	ΔZ	0	0	0	Z	0
				← B-inverse →			

Figure 3. Column, Constraint Relationship

As shown in Figure 3, a change in the (3,2) element may cause changes to all elements in the second column. This column must now be returned to the final tableau form (assume a 1 was in element (1,2) and the other elements were zero). The first constraint would now be divided by 1+ΔW and multiples of the first row would be added to the objective function and to the other rows to return their values in the second column to zero. To be specific, $-\Delta X$ times the first row would be added to row two, $-\Delta C$ times the first row would be added to the objective function, etc. Each of these cases can be set up as an equation to determine what value of Δ will cause a zero value to be reached in a non-basic column of the objective function or in the right-hand side. Each row and each non-basic column (except artificial variables which are excluded) will produce a Δ. From these Δ's, the smallest positive and the smallest (absolute) negative represent the bounds on the change to the element.

The specific equation used to determine the maximum positive change is:

```
Δ= MIN [MIN POS(-C(L)/(A(ROW,BCOL)*C(L)-A(ROW,L)*
  C(BCOL)), MIN POS (-B(M)/(B(M)*A(ROW,BCOL)-B(ROW)*
  A(M,BCOL))]                                          (28)
```

where

C = objective function coefficient in the final tableau

L = 1,2,....total variables---excluding artificials and the column with the delta

ROW = the row which represents the basis of the column under investigation (has a value of 1 in the final tableau)

A = the element coefficient in the final tableau

BCOL = the column in B-inverse associated with the change being investigated (The third column of B inverse if the change was to the (3,2) element.)

B = right-hand-side value in the final tableau

M = 1,2,...K --- all rows except the row in the basis for the column being investigated

The negative delta is similar except the largest negative (smallest absolute) values are determined.

To illustrate:

			ΔC	C3			C	
1	X2	0	1+ΔW	D	H	0	W	0
2	X1	1	ΔX	E	I	0	X	0
3	X5	0	ΔY	F	J	1	Y	0
4	X7	0	ΔZ	G	K	0	Z	1
		1	2	3	4	5	6	7

Figure 4. Change to the Objective Function Coefficient

For L = 3 and X2 is in the basis in ROW 1

$$\Delta(3,2) = -C(6)/C(6)*A(1,3)-C(3)*A(1,6) \qquad (29)$$

$$\Delta(3,2) = -C/(C*D-C3*W) \qquad (30)$$

The value of Δ(3,2) found in equation 30 will cause the objective function coefficient C3 (Figure 4) to be driven to zero. All other columns which are not in the basis must also be checked to find the minimum changes. Basic columns do not need to be checked since they have a zero in the critical element. Therefore, no multiple of the row could change any other row. If the element which is being investigated for range limits is not in a basic column, the only value to be determined is the relationship between the objective function coefficients of the desired column and of the column associated with the constraint. The maximum range limit is determined by Δ= -C(L)/C(BCOL).

The changes for the right-hand side are checked in a similar manner.

		ΔC				C		
X2	0	1+ΔW	A1,3	A1,4	0	W	0	B1
X1	1	ΔX	A2,3	A2,4	0	X	0	B2
X5	0	ΔY	A3,3	A3,4	1	Y	0	B3
X7	0	ΔZ	A4,3	A4,4	0	Z	1	B4

Figure 5. Right-Hand-Side Ranging

ROW = 1, M = 2

$$\Delta = -B(2)/(B(2)*A(1,6)-B(1)*A(2,6)) \quad (31)$$

$$\Delta = -B2/(B2*W-B1*X) \quad (32)$$

Again, each row (M=2,3,4) will produce a delta. All of the deltas (positive and negative) are searched to find the delta which first drives an objective function coefficient or a right-hand side to zero. This will be the range for the element under consideration.

Sensitivity analysis is subject to ill-conditioning which would not be present if the modified problem was solved from the initial tableau. This condition occurs when one of the two constraints (in a two-dimensional problem) which forms the corner-point solution of a final tableau is modified to cause the two constraints to be parallel. The current corner point is now at infinity. This condition would never occur using the full algorithm since one of these constraints would be outside the convex boundary and, therefore, not involved in any basic solution.

Figure 6 shows a two-dimensional problem. The optimal solution is shown in Table V. A critical value to cause

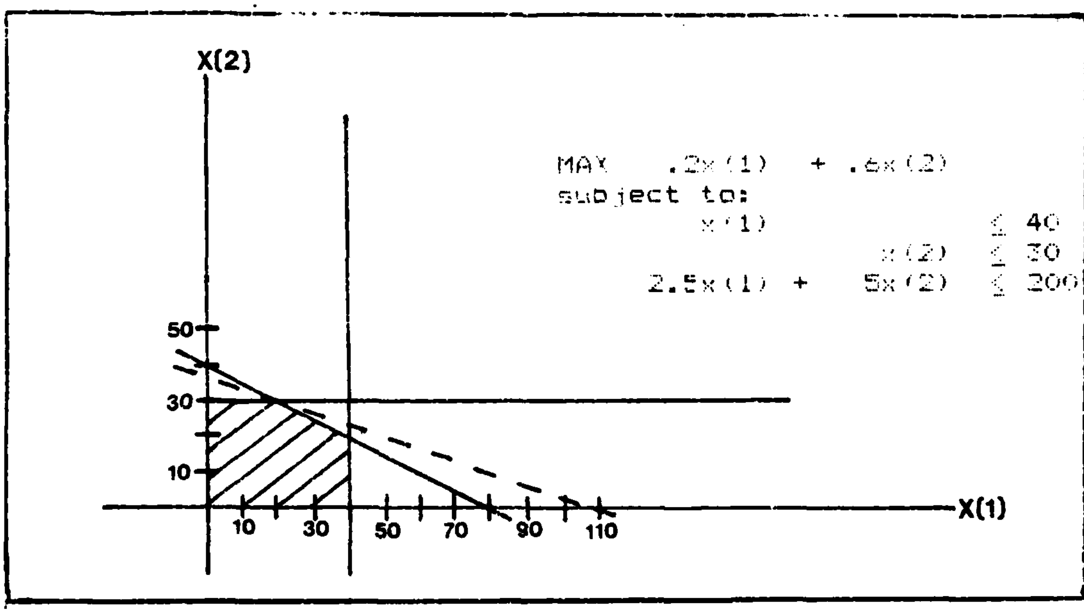


Figure 6. A Two-Dimensional Problem

TABLE V

A Final Tableau

FINAL TABLEAU - OPTIMAL

CN	NAME	VAR	X(1)	X(2)	X(3)	X(4)	X(5)
OBJ FUNCTION			.00000	.00000	.08000	.00000	.20000
1		1	1.00000	.00000	.40000	.00000	-2.00000
2		4	.00000	.00000	-.40000	1.00000	2.00000
3		2	.00000	1.00000	.00000	.00000	1.00000

CN	NAME	VAR	RHS
OBJ FUNCTION =			22.00000
1		1 =	20.00000
2		4 =	20.00000
3		2 =	30.00000

ill-conditioning is a change in the (1,1) element of -2.5. This change is shown in the revised problem in Figure 7. As can be seen, two constraints are now parallel, and one of them is not within the convex boundary. This new problem can be easily solved by the full algorithm (Table VI) but requires division by zero when sensitivity analysis is used to find a new solution from the original optimal solution (Table VII).

Ill-conditioned points exist in every tableau. A critical value may exist for each element in an original column which has its variable in the basic solution. Empirical results indicate that these critical values are usually outside of the range limits for individual elements. The only known exception is the unique case where the change makes the

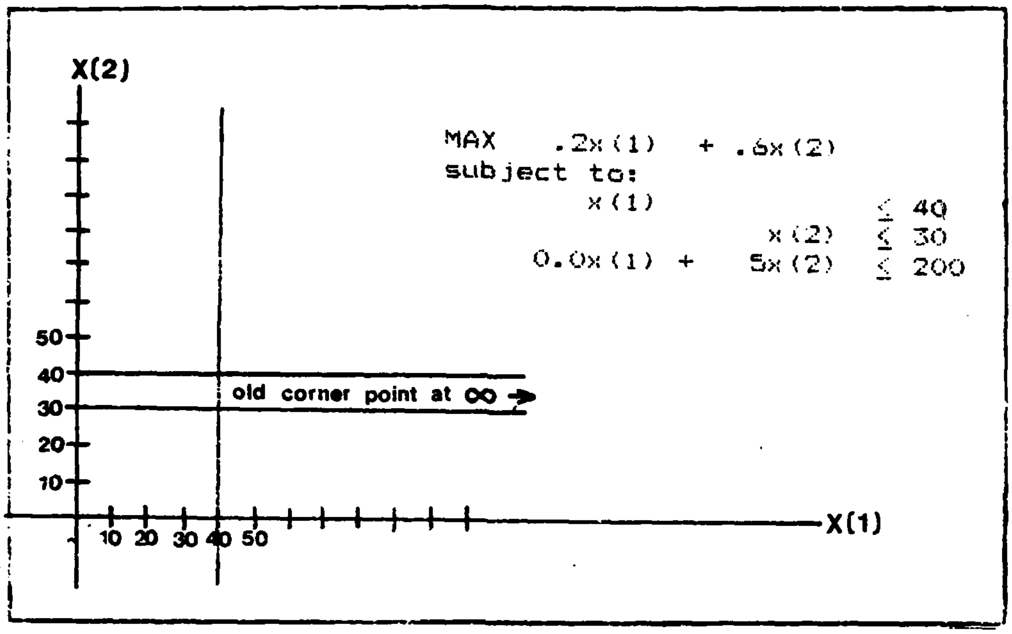


Figure 7. Corner Point at Infinity

TABLE VI

Modified Problem, Final Tableau

FINAL TABLEAU - OPTIMAL

		X(1)	X(2)	X(3)	X(4)	X(5)
OBJ FUNCTION		.00000	.00000	.00000	.20000	.60000
CN NAME	VAR					
1	3	.00000	.00000	1.00000	.00000	-5.00000
2	1	1.00000	.00000	.00000	1.00000	.00000
3	2	.00000	1.00000	.00000	.00000	1.00000

		RHS
OBJ FUNCTION =		26.00000
CN NAME	VAR	
1	3 =	50.00000
2	1 =	40.00000
3	2 =	30.00000

TABLE VII

Division By Zero

OBJ FUNCTION CN NAME VAR	X(1) 0 - .2	X(2) 0.0	X(3) .08	X(4) 0.0	X(5) 0.2
1 1	$\frac{1 - 1}{1 - 1}$	$\frac{0.0}{1 - 1}$	$\frac{.4}{1 - 1}$	$\frac{0.0}{1 - 1}$	$\frac{-2.0}{1 - 1}$
2 4	0 + 1	0.0	-.4	1.0	2.0
3 2	0 + 0	1.0	0.0	0.0	1.0

constraint parallel and identical. This new problem would be ill-conditioned for all solutions since the constraints would no longer be independent.

Ill-conditioning in sensitivity analysis is an interesting phenomena, but its effects are not major. The ill-conditioning can be avoided if the changes are varied a small amount. The full simplex algorithm can be used if the exact changes must be investigated.

The purpose of this chapter was to provide a brief theoretical review of linear programming in conjunction with the simplex algorithm methodology and postoptimal sensitivity analysis as they were applied in this software package. Those interested in a more thorough discussion of the simplex algorithm are directed to Gass (Ref 10) and also Garfinkel & Nemhauser (Ref 9). The area of postoptimal sensitivity analysis is covered in depth by Gal (Ref 8) with again a practical view given by Levin & Kirkpatrick (Ref 16). Many other important aspects of linear programming, such as duality theory, have not been presented but are present in numerous refereneces. A quite thorough and practical presentation of duality theory is given by Levin & Kirkpatrick (Ref 16) while a brief, but comprehensive theoretical view is given by Garfinkel & Nemhauser (Ref 9).

Although many areas were not discussed, hopefully the previous discussion has implanted or reinforced the mathematical background of the simplex algorithm. Also, the simplex algorithm has been intended to be shown not as an abstract technique which is blindly applied, but a valuable tool to assist analysts and managers in solving real life problems.

III Design Considerations

The developmental process involved in an effort to produce a well-designed product requires careful consideration of all aspects which may influence the outcome. The objective of this section is to present the major considerations of the software design phase. Chapter IV will then discuss the method of implementation which has evolved from this analysis.

User Considerations

The user must feel that the software is beneficial in terms of the time and effort required to use it in problem solution and analysis. Therefore, several important factors must be considered with the user in mind. One such consideration is that the program should be developed in a logical sequence from model formulation to problem solution to analysis of results. The user should be carefully guided through this sequence, being allowed to correct either incorrect entries or incorrect problem formulation without resorting to complete model reformulation and input. Furthermore, the input of an option selection or the input of a model should occur in a sequence which coincides with the logical progression of problem solution in order to lessen the anticipation and doubts of infrequent users.

Another important area is the ability of the user to quickly and accurately locate the results of each step in the sequence. The prompts to the user for data or response input should provide meaningful, concise guidance to the user. In some instances, graphically supplemented output may be desirable. This may occur when one value being studied is valid for a range of values for another variable. In this manner, the user is able to visually determine the range, and to a degree, the sensitivity of one variables relationship to another.

Although linear programming and the simplex algorithm are capable of producing the desired solution, alternative methods of applying the simplex algorithm are available which may allow more efficient and timely solutions to an LP model. It may be desirable to provide the user with the capability and option to solve the problem by one of these alternatives. Again, the presentation of these options should be performed as clearly and concisely as possible.

Further software enhancements which may be user desirable include minimal programming and operating system interface. The user should not be required to alter program source code to use the software; however, the programs should be designed and presented in a manner which will not preclude future enhancements or modifications. These considerations suggest that the source code be modularly designed, developed, and documented to reduce the effort required to locate the code of a specific function and then interpret the source code.

One additional area of concern is program control. Since the user may be inexperienced in either programming or use of the operating system, the program should require minimal and infrequent guidance. This consideration leads to a menu-driven program which displays available program options enroute to problem solution and analysis. Upon input of a desired option by the user, the program ideally will perform all interface with the operating system to pass control to the desired program, unit, or subroutine. Compromises to this ideal environment may be necessary. If so, precise instructions to the user on the required operating interface commands needed to progress through the desired sequence should be appropriately displayed to minimize the required familiarity with the operating system.

Hardware Considerations

Microcomputers offer many advantages not available with large, stationary computer systems. These advantages include a substantial decrease in acquisition cost plus virtual elimination of support requirements. Also, the transportability of the "desk top" computers is ever increasing due to recent advances and design considerations. In conjunction with these advantages, one would expect and soon finds areas of diminished capabilities compared to a mainframe system. Two primary areas are the decreased memory capacities and reduced computational rates of the microcomputers. This exchange of decreased cost and increased transportability for decreased capabilities does impair the size and speed of developed software; therefore, one must consider possible avenues which will counter these decreased capabilities.

In determining the target system for software development, one should consider the availability of the various microcomputers in conjunction with the capabilities of each. Software which is developed on a system with limited availability to the target users will not be used extensively. Even if a readily available system is used for software development, it is necessary to consider the modifications and peripherals of the target system. If these modifications or peripherals are unique to the development system, it limits the use of the software.

Although the available microcomputers vary extensively in their memory capacities and peripherals, a range from 48K to 128K bytes random access memory (RAM) is not unusual (48K is approximately equal to 48,000 characters). Peripherals, such as printers, communication links (modems), and disk drives are quite common, if not necessary, among those who use microcomputers. The dependency of the developed software on these peripherals must be strictly specified, or options must be provided which will allow the user to designate only those peripherals possessed. In this way, many potential users could gain access to the software without an added hardware requirement.

Language Considerations

A primary factor in the selection of a language for this research was the portability of the language from one microcomputer system to another. BASIC was once the only high-level language generally available for use with microcomputers. Presently, microcomputer users have access to other high level languages such as Pascal, FORTRAN, and APL. The three languages felt to be most accessible to users were BASIC, Pascal, and FORTRAN and were considered for implementation in this research.

BASIC. As previously noted, that the portability of the language is a primary consideration. From this viewpoint, BASIC does not gain much support since each microcomputer system has modified the BASIC language in order to coincide with the needs of that particular system. This has caused many versions of the BASIC language and results in extremely limited portability.

Execution time is also dependent on the language selection. A program written in BASIC must be interpreted line by line to machine language each time the program is executed. This process causes execution time to be considerably slower than for other languages which are first compiled and then executed. BASIC programs may be compiled; however, the compilers are machine dependent due to language and hardware differences between systems. This compiled code is then unique to the system and further deters portability.

Pascal. Pascal, the most recently developed language of those considered, emerged in the early 1970's and has since proven to be a powerful, high-level language. A significant factor in the growth of this language was the work of Kenneth Bowles who directed the development of UCSD Pascal at the University of California at San Diego (Ref 14:118). UCSD Pascal, originally formulated as a teaching tool, has allowed for larger programs to be implemented on microcomputers and has lead to an increased portability of high-level languages between systems.

The enhancements to Pascal and FORTRAN are the result of the "P-code" and "P-machine" of the UCSD Pascal system. The source code, either Pascal or FORTRAN, is first compiled into P-code and stored as the object code for the P-machine. The P-machine, which interprets the P-code upon execution, emulates instructions to the machine-specific central processing unit. Only the emulator of the UCSD Pascal system must be revised to coincide with each system to allow portability, and this may be performed in a relatively short time. This one-time revision then allows compiled Pascal or FORTRAN programs to be portable between microcomputers. Another advantage of the P-code is that it requires considerably less memory than the equivalent object code or machine language. This allows larger programs to be resident in memory at any one time. Also, UCSD Pascal has implemented an overlay capability which loads P-code into memory as needed and discards this code upon completion. This capability allows the user to specify, to a great extent, the amount of code in memory during the various phases of program execution (Ref 14:114). The advantages of the UCSD Pascal system are applicable to both Pascal and FORTRAN programs; therefore, little preference is gained for either high-level language.

A disadavantage of Pascal is that it does not possess extensive output formatting capabilities which are often found in other languages. Although not an insurmountable problem, it is a factor when carefully formatted output is desired.

FORTRAN. FORTRAN, the most widely used language within the scientific programming community, was developed in the early 1950's by IBM (Ref 3:1). In the succeeding years, the FORTRAN variations have increased, leading to a need for standardization of such an intensely-used language. Two attempts have been made, with the most recent (1970-1977) specifying a full language and subset language. Attempts to standardize the language have been of assistance to programmers, however, discrepancies between implementations still exist and are a major downfall of the language.

As previously discussed, the portability of FORTRAN has been greatly aided by the development of the UCSD Pascal operating system. Also, due to FORTRAN's evolution process, formatting features are available which are not present in BASIC and Pascal. A consideration which has not been mentioned is that although increased formatting capabilities do exist, output time is increased when this option is utilized. Dependent upon the type and amount of output desired, this may become a significant factor in program execution time.

The above sections have discussed those features which are desired in the software and also the availability of hardware and languages with respect to microcomputers. The next chapter will discuss the selection of a particular microcomputer system and language. Also discussed are the methods in which those desired features were incorporated into the software within the constraints of the hardware and language selections.

IV Implementation

The objective of developing an extensive microcomputer-supported linear programming software package combined with the design considerations discussed in the previous chapter have formed the foundation for this research. Each area discussed in Chapter III was felt to be of significance; however, each must be reviewed in light of the capabilities, advantages, and disadvantages of microcomputers as well as the desires and abilities of the intended users. Due to the requirements placed upon the design by each of these factors, conflicts may arise which will allow less than full incorporation of one or more of those desired areas.

The purpose of this chapter is to present the method of implementation used in the software development of this research. First, the method of incorporation of those significant areas discussed in Chapter III will be discussed as they apply to the software package in general. Next, the method and underlying logic of implementation will be reviewed as it applies to each module of the software package. Then, specific problems or special areas of consideration will be discussed. Chapter V will present findings and recommendations which are felt to be noteworthy to future efforts in similiar research.

Hardware

One of the first major decisions of the implementation phase concerned the selection of the microcomputer system to be used in the development. It was felt that the system should be a commonly available system without extensive or unique modifications or peripherals. Also, the memory capacity of the chosen system should be comparable to those most likely accessible by the intended users.

In light of the above considerations in conjunction with the availability of such a system to the developers, an Apple II-plus microcomputer was selected as the developmental system. The Apple II-plus, although not the most advanced microcomputer on the market, has become one of the largest selling systems and is presently selling approximately *20,000 microcomputers per month* (Ref 18:19). Other favorable features of the Apple II include high-resolution graphics, memory expansion capabilities, readily available peripherals, and the ability to support the Pascal and FORTRAN languages.

The system which was used for the implementation of this software package is shown in Figure 2. The system, as shown, was felt to be representative in capabilities of those microcomputers available to the intended users.

1. Apple II-plus microcomputer with 48K RAM
2. Two disk drives (5-1/4 inch)
3. 16K memory expansion card (language card)
4. Printer
5. Video display (monitor or TV)

Figure 8. Developmental Hardware Configuration

<u>Language</u>

The three languages which were examined for implementation in this research were BASIC, Pascal, and FORTRAN. FORTRAN, even though it does have limitations and drawbacks, was felt to be the most appropriate language as supported by the Apple II microcomputer.

The BASIC language, either Applesoft or its machine specific counterparts, was found not to be portable to any extent, thereby violating one of the primary language selection criteria. Furthermore, as an interpreted language, the execution times were known to be in excess of compiled languages such as Pascal and FORTRAN.

The selection between Pascal and FORTRAN was much more subjective than the elimination of the BASIC language. In the past, Pascal had offered a greater degree of portability. However, the recent development and use of the UCSD Pascal Software System (Ref 14) has allowed increased portability of both FORTRAN and Pascal. This system allows both Pascal and FORTRAN to run on most microcomputers thereby virtually eliminating portability problems due to hardware configurations.

A frequently discussed obstacle to FORTRAN is its lack of standardization among the various implementations. Although the magnitude of this problem has been reduced, it still exists for large computer systems as well as microcomputers. It was found that even though a standard Pascal language exists, the microcomputer implementations of this language often do not coincide fully with the standard, allowing for portability problems similar to FORTRAN.

Although Pascal has come to be known as a powerful high-level language, it is relatively new and not as intensely used by the scientific community. It was felt, even at the possible expense of perpetuating an outdated language, that FORTRAN would be an acceptable language by the scientific community due to its familiarity. Also, the well-developed intrinsic output formatting capabilities of FORTRAN, which were extensively used in the research, would allow for a more rapid implementation than those available with Pascal.

User Interface

The requirement for user interaction while employing a microcomputer is inevitable; however, the degree of such interaction is largely programmer controllable. The previous selection of hardware and language has also delineated the magnitude and frequency of user interaction to some extent, dependent upon program size and complexity. It has been noted that the targeted users should not be required to possess extensive programming or operating system knowledge. This further requires that any interface be preceeded by well structured, concise guidance to the inexperienced user. This user interface requires user input of responses or data which may be entered incorrectly causing an execution error. To preclude possible operating system errors due to erroneous input, a system of screening user input has been constructed to prevent such losses. This section describes the manner in which the interface system has been designed within the confines of the hardware and language parameters as well as the design considerations outlined in Chapter III.

Apple FORTRAN, the FORTRAN version implemented on the Apple II, allows the use of a "turnkey" system which automatically begins running a programmer-designated program following prescribed startup procedures (Ref 1:9) (See Appendix A for startup procedures). A turnkey system requires minimal operating system interface with the user for a program to initially gain control and begin to guide the user. Such a system has been included to lessen the interface initially required for software use. Therefore, upon startup, the first program of the package, Module 1, is automatically executed.

The package consists of four separate programs due to the physical size of the LP package and memory capabilities of the selected hardware system. This factor, coupled with the inability to "chain" programs in Apple FORTRAN (in chaining, one program may cause the execution of another program without user intervention), has forced the design to include, to a degree, user interface with the operating system. This interface, which occurs when the user requires a different program to continue the solution process, has been designed such that the user will select from a menu the course of action desired. The program which is currently in control will then terminate and display specific instructions to the user. These instructions will enable the user to execute the desired program of the package.

The previous discussion noted that the user would be presented a list of alternatives. The user would then select one, after which specific operating systems commands would be presented. This method of menu display and user selection input may also be applied within a specific program to cause execution of a particular subroutine or a sequence of code. This method, often designated as "menu-driven", requires minimal user input to cause the desired actions. This method has been implemented, where appropriate, in this software package. User inputs are normally limited to either numeric input (Options 1, 2, or 3), character input (P for printer, S for screen), or Yes/No inputs (Y for YES, N for NO). User inputs are screened to prevent undesired program termination.

The high-level languages such as FORTRAN do not support intrinsic input error checks which prevent premature program termination and data loss when user inputs are of an improper type or range. Therefore, to assist in prevention of such an event, all user inputs except problem, constraint, and variable names are first screened to insure that they will be acceptable to the program. All user inputs are first placed in character strings which allow any type of numeric or character input. Next, depending on whether the input is a character or a numeric representation, these inputs are inspected character by character.

Character inputs, such as an option selection involving options P, S, or B or a Yes/No response, are checked to insure that one of the possible responses for that particular input has been entered. If so, the option represented by the user input is performed. However, if the input does not coincide with the possible alternatives, a message indicating an invalid input is displayed and the user is directed to reenter the input. This process is repeated indefinitely for any required user input.

Numeric inputs have been separated into real and integer numbers. Inputs which require integers are inspected character by character to insure that all individual entries are numerics or blanks. If so, the character string representation is converted to its equivalent numeric representation. At this point, the numeric value is checked to insure that it is within the allowable range for the specific response. If a non-numeric is found or the range is violated, the user is informed of an invalid input and directed to reenter the number. Real numbers are treated similiarly to integers except that a decimal point has been included in the set of valid entries. Also, range screening is not performed on real input. It should be noted that real numbers may be input as integers (i.e. if 1 is entered, it will be internally represented by 1.0) and that commas (i.e. 10,000) will not be accepted in real number inputs. Also, a value of zero may be entered by simply pressing RETURN without numeric input.

These steps allow the user to enter an invalid response and recover without data loss; however, an undesired input which is valid to the program will be accepted and the user will be forced to correct this erroneous input. In the case of an erroneous option selection, the user will be forced to complete the process under the option entered.

Software Description

This linear programming software package consists of four distinct FORTRAN programs. There were two primary reasons for the separate programs. First, the memory capacity of the microcomputer would not allow the software to be developed in fewer than the four which were used. Second, if the software could have been implemented in one program, the compiled code would have been too large to be stored on one disk. Therefore, these programs (annotated as Module 1 through Module 4) have been designed and implemented in a sequence which coincides with the LP model formulation, solution, and analysis sequence used in most analyses.

The four modules serve separate purposes, but are related. Module 1 must be used for problem entry before Modules 2 or 3 may be used. Module 2 (the instructional module) or Module 3 (the problem solver module) must be used to generate a final solution before sensitivity analysis (Module 4) can be performed. The following sections will discuss the purpose of each module and their relationship to each of the other modules of the package. Problems which were encountered and the methods of correction will be discussed.

Module 1 - Data Base Entry Module. This module, which is the program automatically executed upon startup through the use of the turnkey system, serves two primary purposes. First, it provides an initial entry point into the software package where the program may begin to guide the user through the sequence of model entry, solution, and sensitivity analysis. This module's second primary function is data base entry. The user is guided through the steps of selecting the type of model to be entered as well as entering of the various parameters of the LP model desired to be studied.

Module 1, when executed, presents the user with several options related to the possible alternatives available in the complete LP package. The user may elect to enter a model data base; in which case, Module 1 continues to guide the user. If the user elects to solve an LP problem using either Module 2 or 3, Module 1 provides the necessary guidance. The last choice is sensitivity analysis. If this option is selected, Module 1 informs the user of the required data bases and the required steps to implement Module 4.

The above selection allows Module 1 to provide specific instructions to the user on the operating system commands required to enter the desired module. As noted, this is one of the primary functions of this module. It should also be pointed out that, upon the completion of any of the other modules, the user is given instructions to execute Module 1 which will then again present the various options. The user, at that point, may select the desired step of the sequence and continue the analysis.

The second purpose of this module is model entry. If the user elects to enter a data base, Module 1 continues to guide the user through the required steps. This portion of the module consists of three primary sections: data base entry, data base management, and execution management.

Data Base Entry: This section allows the user to enter a model with or without names associated with the various parameters of the model. Also, a model which has previously been stored on a disk may be retrieved for inspection or editing. If the user elects to enter a model, a series of prompts are presented, which requires either an option selection or parameter input. Prior to the entry of the model parameters, instructions are displayed which inform the user of the method and order to be followed in parameter input. During this series of prompts, the user has access to extensive editing functions which allow correction of any previously entered option selection or parameter input.

These editing features, in conjunction with the screening of user inputs allow the inexperienced user to correct invalid responses as well as incorrect input (meaning valid to the program but not the user intended input) without having to resort to complete model reentry to correct either type of error. The editing features, available during data base entry and data base management, constitute the majority of this module in the terms of FORTRAN code; however, this feature was felt to be important in terms of usability of this software package. The number and degree of editing features to be employed was subjective in nature; however, previous work by Conte (Ref 6) aided greatly in the selection of those which would be most beneficial to the user.

Data Base Management: Upon completion of the initial data base entry or retrieval of an model from a disk, the user is presented several new options. The user may display the model to insure that all parameters have been entered as desired. Also, this option allows the output device to be either a monitor or printer. With this option, those users who possess printers may then receive a hard copy of the input, while those who do not have printing capabilities still have the ability to review input. Should the user find an error in the model, access is provided to the editing features of this model and changes may then be performed. The user may elect to save the model to a disk under a user-specified filename. It should be noted that each model must be saved to disk following data entry and editing to allow further study. This requirement arose from the inability of the package to be implemented as one program. To prevent the user from inadvertently leaving this module without first saving the model to disk, a prompt appears which warns the user that the current model will be lost if not saved.

The ability to read a model from a disk, combined with the editing features, allows a user to use one model as a starting point, perform changes and then solve to determine the effects of those changes. The above procedure would not normally be required due to the extensive sensitivity analysis features of Module 4; however, the user may wish to see the complete solution process with certain parameter variations applied.

Execution Management: Once the initial model has been saved to disk, the user may elect to enter another model with the data base entry section; however, one would normally elect to solve the model at this point. A menu of options allowing Modules 2, 3 or 4 to be selected is displayed when the user elects to solve the problem. An option of returning to the data base management menu is also provided to allow recovery from an incorrect or undesired input.

The user selects the desired module and is then guided through the required entries. A problem which was encountered during the transition phase was the transmittal of the diskname:filename. To solve to this problem, two data files were created, LP1:LPDATA and LP2:LPDATAW with these files being placed on the disks LP1 and LP2, respectively. These files contain the diskname:filename of the model currently being studied. LP1:LPDATA contains the name of the most recently input or edited model while LP2:LPDATAW contains the diskname:filename of the file which contains the results of the most recently solved problem by Module 2 or 3. LP1:LPDATA is automatically read by Modules 2 and 3 while LP2:LPDATAW is read by Module 4 to determine the file which contains the inputs for that module.

One other area which required careful planning was the arrangement of the various editing subroutines in the overlayed compilation units. Due to the extensive interaction of the various editing functions, attention was required to avoid several overlayed units being resident in memory simultaneously thereby overloading the capacity of the microcomputer. This problem was eliminated by cross-referencing the subroutine interaction and then forming the compilation units so that an overloading of memory would not occur.

Module 1 does not perform or enhance the LP model solution process but provides valuable organization support for the overall package. Organizing the package so that one program provides this support allows more memory to be available for the individual functions of each of the remaining modules whose specific functions will be discussed.

Module 2 - LP Instructional Module. The goal during the development of Module 2 was two-fold. First, the module would allow the user to select a previously entered LP model and apply the simplex algorithm to that model to determine an optimal solution. Second, it was envisioned that this module would be designed so that the user could be assisted in learning the application of the simplex algorithm to an LP model. Although the primary concern of the instructional portion was to assist students in their initial contact with the simplex algorithm, these features also allow instructors to demonstrate to students the outcome of procedures or selections not complying with the simplex algorithm.

The initial guidance in reference to notation and the sequence of steps representative of simplex algorithm computer implementations was drawn from Gillett (Ref 11:101-105). Although the FORTRAN code presented in Gillett was of assistance in forming the code of this module, extensive modifications and extensions were necessary in order to implement the desired instructional features.

This module and Module 3 are quite similar in that they both allow the solution of an LP model by the simplex algorithm. Due to the desire to provide extensive feedback to the user during the instructional process, this module was much larger than that of Module 3. The size of this module's FORTRAN code required, as did Module 1's, careful consideration of memory capacities in the forming of the compilation units. As is discussed in Appendix B, the manner in which the compilation units are formed may dictate the size of the overall program.

Several steps are required to insure the LP model is in the proper form to apply the simplex algorithm. The standard LP form upon which this and the following modules have been based has been illustrated in Chapter II. The process of modifying the LP model to coincide with the stated form has been labeled as "tableau formulation" and involves the manipulation of the objective function and resource constraints. Once the tableau formulation is completed, the iterative process of the pivot element selection and moving to an adjacent feasible solution is performed. After moving to each new feasible solution, one must check to insure that an optimal solution has not been reached prior to continuing the iterative process. Therefore, the application of the simplex algorithm has the following three major steps:

1. Tableau formulation
2. Pivot element selection and determination of new basic feasible solution
3. Check for optimality, unboundedness, etc.

The instructional areas of this module have been divided into three areas coinciding with the above steps. In order to allow the user to concentrate on the specific step or steps desired, options have been provided which allow either: 1.) the user to direct the manner each step is performed or 2.) the program to perform these actions without user interaction.

The three areas in which instructional assistance has been provided have been implemented differently. During tableau formulation, several options which modify the objective function or constraints are presented to the user. The user selects that option which is applicable to the objective function or to the constraint under consideration. At that time, the user is provided feedback as to whether or not the selection was correct; and, if not, the correct selection is displayed. Although the user's selections may be incorrect, the objective function constraint is modified properly.

The pivot element selection process has been implemented differently than the process described above. The user is presented two methods to select the pivot element: with or without feedback. If the user elects to receive feedback, the user is allowed to correct improper pivot element selections to coincide with the algorithm selection which is also displayed. The user may also elect not to receive feedback on the validity of the selection, and the user-selected pivot element is used for further computations. Under both methods, the user could request that the algorithm divide by zero and cause an execution error. In order to allow the maximum possible freedom of pivot element selection for demonstration purposes yet prevent inadvertant execution errors, a system of checks has been implemented which warns the user that an execution error may occur if the pivot is performed as selected. The user is then allowed to either continue with the previous selection or select a new pivot element.

The step in which the optimality, unboundedness, and infeasibility of the current solution are checked has been implemented similarly to that of tableau formulation. Regardless of user input, the cycle of steps 2 and 3 will continue until true optimality, unboundedness, or infeasibility exist. During each cycle of the steps, the user must enter responses to questions pertaining to the status of the last tableau and receives feedback on these responses.

Module 2 has implemented, at the expense of less descriptive feedback on user selections, a capability to perform dual pivots. This capability allows the user to become familiar with the dual simplex algorithm while also allowing a more efficient solution technique to be used in certain situations. Again, a system of checks has been incorporated which prevents the user from improperly applying the dual (or regular) pivot method.

Due to memory size limitations, the expansion of any one portion of this module required reduction in another. Considerable effort was required to allow both dual pivots and descriptive and informative feedback to the user. In order to allow the most detailed comments possible, standard formats for comments were used with variables representing the areas applicable to the situation. The comments were more descriptive than would have been possible if separately implemented comments were used for each situation.

Several features have been incorporated into this module and Module 3 which increase the ease of use and decrease the time and peripherals required. First, the user may elect that output be displayed in either scientific notation (e.g. 12.01E+02) or in the more common representation (e.g. 1201). Although the output format has no influence on the computations performed, the numbers may become too large or too small to be accurately represented in the space allocated in the output without the use of scientific notation. Any numbers larger than 999,999 or less than 0.00001 will be represented by asterisks or 0.00000, respectively, if the scientific notation is not requested.

A second feature allows the user to have the tabular output displayed on the screen or on a printer. This option allows users without printing capability to use the software and allows those with printers to receive hard-copy output for later use and study. The format, which is identical for both output devices, has been constructed in a manner to facilitate viewing on a screen. The tableaus are displayed in sections from left to right with a user-controlled pause between screen displays. This allows the user to study information from that portion of the tableau prior to the next screen being displayed.

The last feature is primarily applicable to those who wish to use this module as a problem solver without interaction. This option allows the user to designate the specific tableaus to be displayed. This option, if minimal output is required, allows a solution to be obtained more rapidly since less output time is required for the intermediate tableaus to be displayed. The user should be cautioned against changing the programmer-defined defaults for tableau output when using this module for instructional purposes. Should these defaults be changed, the user will not be able to determine the tabular outcome of the previous iteration and, therefore, may not be able to select the proper pivot elements for future iterations.

When an optimal solution has been obtained, the user may save the solution on disk to allow for future sensitivity analysis using Module 4. It should be noted that once a problem has been solved using this module, there is no reason to use Module 3 for the same problem. Module 3 performs the same computations as Module 2 with minimal user interaction.

Module 3 - Problem Solver Module. Module 3 is similar in purpose to Module 2. The major distinction is that instructional comments have been deleted allowing the addition of features not possible in Module 2 due to memory size limitations. The primary feature available in Module 3 not previously available is the ability to explicity solve the dual LP model of an LP problem entered in Module 1. Although user interaction is not required in the transformation process from the primal to the dual problem, the program does inform the user when variables are added to allow for unconstrained variables. One may wish to review duality theory to fully understand this requirement.

Hillier & Lieberman (Ref 13:91-109) offers a basic review of this area. The message which advises the user of added variables denotes the subscript of the original variable and the corresponding added variable so that later tableaus may be correctly interpreted.

This module, as does Module 2, displays the programmer-defined default options upon execution of the module. The options displayed include the type of problem to solve (primal or dual), whether or not to use dual pivots, the output format and destination, and the specific tableaus to be displayed. The selection of these default options was based on the subjective judgement of the frequency of use of the various options. Also, the default options were defined to be consistent between Modules 2 and 3. In this way, a user who has become familiar with the operation of one module may transition smoothly to use of its counterpart.

A problem not encountered in Module 2 implementation arose during the incorporation of the dual problem solving capability. Module 4, the sensitivity analysis module, requires access to the parameters of the problem as entered or, for dual problem solution, as it would have been entered as a primal problem. Since Module 3 performs the transformation of the primal to the dual, the parameters of the dual problem have not previously been saved to disk. This requires that a new data file be created to contain these parameters. Initially, it was envisioned that the original data file containing the input to Module 1 would be rewritten to contain the dual problem parameters. Two problems were encountered. First, the original data file would be lost, and the user could no longer use this file to edit the model for further problem formulation. Second, and most importantly, due to the possible addition of variables, the new file possibly could not be stored on disk in the space allocated to the original data file. This problem would cause an input/output error and premature program termination. Both of the above problems were overcome by the creation of a new file with a user-specified diskname:filename.

The primary areas noted above were those which differed from that of Module 2 or in which special problems arose. The user may save the results of problem solution to disk for later sensitivity analysis with Module 4. The user is then allowed to retrieve another LP model from disk for solution or to receive instructions on the operating system commands to enter Module 1. From Module 1, the user may elect to perform any of the functions available in the software package.

Module 4 - LP Sensitivity Analysis. Module 4 provides several types of sensitivity analysis for linear programming problems previously solved by Modules 2 or 3. The types are 1) finding range limits for the right-hand side, the objective function coefficients, or the variable coefficients; 2) solving for a new final tableau after one or more changes to any combination of right-hand side or other coefficients; 3) solving for a new final tableau after a new variable or a new constraint has been added. The sensitivity analysis program is structurally separated into eight parts. A master menu calls one of four sections which allow the user to specify the type of analysis desired and to enter required data. These four sections, in conjunction with three additional sections, perform the analysis and display the results on screen or printer. Each of the seven sub-programs are overlayed on the main program when needed. This allows the total package to greatly exceed the immediate memory of the computer.

The sensitivity analysis package originally included a section which would have presented the results in a high-resolution graphical display; however, the graphics package could not be overlayed. With the graphics held in memory, the remaining space was not sufficient to run any of the other sensitivity analysis sections. Because of this problem, the graphics portion was deleted.

All sections of the sensitivity analysis require data from either Module 2 or 3. This data includes the original tableaus as entered by the user in Module 1 (except for problems solved by the dual method). If the dual of the problem was solved, the sensitivity analysis receives and displays all data as though the dual of the problem had originally been entered. This means that the right-hand-side values and the objective function coefficients will be switched and all elements of the matrix will be transposed. The sensitivity analysis section also requires the full final tableau from Modules 2 or 3 as well as a number of flags and parameters indicating the solution conditions and method.

Sensitivity analysis is based on relationships between values in the tableau. The data received from Modules 2 or 3 have the ordering of the final K (number of constraints) columns based on the original constraint type ($\leq, \geq, =$). Sensitivity analysis is least complicated when the last K columns are aligned according to their association with the constraints. That is, the first column of the last group is associated with column 1, etc. This reordering is accomplished in Unit 48 immediately after the data is read from the data file. (See Appendix B for specific information on the units and their purposes.)

Unit 41 provides right-hand-side upper and lower bounds for the current optimal solution and displays the value of the basic variables and the objective function (z) at each of these bounds. Unit 42 provides upper and lower bounds for the objective function coefficients and for all original matrix coefficients. Units 41 and 42 require no user input other than selecting the option and the desired display (screen, printer, or both).

Unit 43 allows changes to one, any, or all of the original values of problems. After the effects of all changes are summed, the tableau is returned to a basic solution form, and a new optimal solution is obtained using Unit 45. Unit 43 requires user inputs for each desired change.

Unit 44 allows the addition of a new constraint or variable to the original problem. These additions are a special case of the changes investigated in Unit 43. In this case, the original values are assumed to be zero and the changes which are a result of the added constraint or variable are determined. Units 43 and 44 use Unit 47 to check the validity of inputs.

Unit 45 solves the modified problem (if possible) and then displays the new optimal solution. The display routines are generally limited to values between 10^5 and 10^-5. Since the optimal solution must always have a coefficient value of 1 in the element representing the basic variable, the solutions tend to be "sized". Any value outside of the displayed range is suspect and may be caused by ill-conditioning. Such values are usually printed as either zeros or *'s. Ill-conditioning may be the result of the sensitivity analysis techniques and does not necessarily imply that the original problem (as modified) would be ill-conditioned if solved by the other modules.

Several special conditions may arise which will preclude completion of the desired sensitivity analysis. If the total change to any element of the primary matrix sums to approximately -1 for a variable which is in the basis and has a value of one, the new value will approximate zero. Since this value is used as a divisor for the entire row, any non-zero value in the row will approach infinity after division, and the problem will not be solvable. In this case, the user is returned to the main menu.

Whenever the new problem, developed through sensitivity analysis, is either unbounded or infeasible, these conditions are noted, and the solution process is terminated. Other than these noted special cases, the sensitivity analysis should produce the requested output.

Summary. The considerations discussed in Chapter III, combined with the desire to design an accurate and responsive analytical tool, have caused some compromises in each area. The ability to extensively edit or recover from erroneous input has decreased the possible thoroughness of instructional comments. This exchange was felt to be justified since erroneous inputs could require a complete input of data causing the user to be hesitant in future use of the software.

An area not previously discussed is the use of prompts on which disk must be available in a disk drive to insure that an input/output error does not occur. Initially, all prompts on disk availability assumed a two-disk drive system configuration with one drive of the disk containing the module being used (See Appendix B for disk file structure). It was found that by inserting additional prompts, the LP package could be used with a one-disk drive system. This modificaiton, although it slows the overall solution or analysis process, allows users not possessing two drives to be potential users. This was felt to be an improvement to the package because of an increased potential audience.

The above enhancement, combined with the user being allowed to select an output device, has allowed the minimal system configuation shown in Figure 3 below.

1. Apple II-plus microcomputer with 48K RAM
2. One disk drive (5 1/4 inch)
3. 16K memory expansion card
4. Video display (TV)

Figure 9. Minimal Hardware Configuration

V Conclusions and Recommendations

The major objective of this thesis was the development of an extensive linear programming software package which could be used on a microcomputer system. Several sub-objectives were also defined in the preliminary phases of the research. These sub-objectives were: 1) the design and implementation of the software package would allow maximum portability among microcomputer systems; 2) the software would be user-oriented and designed for use by non-programmer oriented persons; 3) the programs would be able to provide interactive instructional sessions on the simplex algorithm; 4) the software would provide true sensitivity analysis which could be supplemented with a graphical depiction of the parameters and their ranges.

The sub-objectives were often found to be in conflict with each other. The amount of FORTRAN code required to implement all of the desired features could not be implemented in one program due to memory size limitations. Furthermore, the code could not be stored on one disk. To retain many of the desired capabilities, a decision was made to compromise the simplicity of use of the software. The software now resides on two disks in four separate programs. This is somewhat confusing to the new user; but the disadvantages are short-lived while the advantages of retaining the desired features are long-lasting.

The decision to use two disks and four programs also resolved several other problems which were computer-memory size dependent. These retained features include: the ability to run the software on a single-drive system; user prompts abundant with instructions; and the ability to choose the formatting parameters. A desired feature which was not retained was the capability to graphically depict sensitivity range and parameters. The Pascal Operating System, as supported by the Apple II, does not allow the graphics package to be overlayed in memory when using FORTRAN. When programming in Pascal, a different memory allocation scheme is used which permits graphics overlay (Ref 15). Because of this limitation in FORTRAN, graphics and large computational programs could not be combined. A graphics capability would have required a fifth program on a third disk. A compromise had already been made in the size and complexity of the linear programming software package and, therefore, the determination was made to forego the graphical capability.

As a result of using FORTRAN, several other limitations or deficiencies are resident in the programs. There are no provisions in the programs to recover from incorrect file access. If an attempt is made to open an old file where none existed, or to open a new file when an old one of the same name existed, the program will abort. Provisions have been placed in the programs to give the user an opportunity to correct the filename input, but mistakes are possible which will terminate the program with an execution error. Another deficiency is the lack of character string manipulation capabilities. If concantentation existed in the Apple FORTRAN subset, the appearance and meaning of some prompts and other output could have been improved. These limitations are not present in Pascal. However, implementation of these features in Pascal would require additional source code and could result in memory overload. This would require the elimination of other desired features.

There are several features which could be added to this linear programming package; however, the risk of increased complexity and size may be incurred with these additions. The first feature is the graphical capability previously discussed. The ranges and parameters derived in Module 4 could be written to a disk file. A fifth program could then be executed to display the desired values. This feature could be further enhanced with the capability to print the graphics to hard-copy.

A second area of investigation could be the problem size limitations of the programs. The current system limits the final problem to twenty constraints and sixty variables. This size approaches the system limit as currently coded. By developing a version with reduced features, the maximum size could be enlarged.

Other mathematical programming techniques, such as integer programming and goal programming are also primary candidates for future enhancements. Both techniques could be partially supported by the present modules; however, extensive additions would be required.

This software package has accomplished the objectives, with the exception of the graphics capability. Future research and microcomputer developments may allow this product to be enhanced or variations may be developed for specific purposes. Many of those included features, as well as envisioned modifications to this software, are dependent on the capabilities of the microcomputer system. It must be realized that microcomputers have finite limitations due to memory capabilities and the current language implementations. These factors and their interactions have placed obvious constraints on this linear programming implementation and requires careful research before attempting an effort of comparable magnitude.

Bibliography

1. Apple Computer Inc. Apple FORTRAN Language Reference Manual. Cupertino, California: Apple Computer Inc., 1980.

2. Apple Computer Inc. Apple Pascal Operating System Reference Manual. Cupertino, California: Apple Computer Inc., 1980.

3. Balfour, A. and D. H. Marwick. Programming In Standard FORTRAN 77. New York: North-Holland Inc., 1979.

4. Bradley, S. P., A. C. Hax, and T. L. Magnanti. Applied Mathematical Programming. Massachusetts: Addison-Wesley, 1977.

5. Chung, An-min. Linear Programming. Columbus, Ohio: Charles E. Merrill Books, Inc., 1963.

6. Conte, Robert D. Computer Assisted Analysis for Military Managers. MS Thesis, Wright-Patterson AFB, Ohio: Air Force Institute of Technology, December, 1979. (AD A080215)

7. Daininger, Rolf A. "Teaching Linear Programming on a Microcomputer," TIMS-ORSA Meeting, Detroit, Michigan, April, 1982.

8. Gal, Tomas. Postoptimal Analyses, Parametric Programming, and Related Topics. New York: McGraw-Hill Book Company, 1979.

9. Garfinkel, Robert S. and George L. Nemhauser. Integer Programming. New York: John Wiley & Sons, 1972.

10. Gass, Saul I. Linear Programming - Methods and Applications (Fourth Edition). New York: McGraw-Hill Book Company, 1975.

11. Gillett, Billy E. Introduction To Operations Research: A Computer-Oriented Algorithm Approach. New York: McGraw-Hill Company, 1976.

12. Gottfried, Byron. "Micro-LP : A Microcomputer-Based Linear Programming System," TIMS-ORSA Meeting, Detroit, Michigan, 21 April, 1982.

13. Hillier, Fredrick S. and Gerald J. Lieberman. Introduction to Operations Research (Third Edition). San Francisco: Holden Day, Inc., 1980.

14. Irvine, C. A. "USCD System Makes Programs Portable," Electronic Design, 28 (14): 113-118 (August 1980).

15. Lee, David B. Enhanced Decision Analysis Support System. MS Thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, March, 1981.

16. Levin Richard I. and Charles A. Kirkpatrick. Quantitative Approaches to Management (Third Edition). New York: McGraw-Hill Book Company, 1975.

17. Orchard-Hayes, William. Advanced Linear-Programming Computing Techniques. New York: McGraw-Hill Book Company, 1968.

18. Shaffer, Richard A. "In a Rapidly Changing Field, Apple II Shows Staying Power," The Wall Street Journal, pg. 19, 14 May, 1982.

19. Strang, Gilbert. Linear Algebra and Its Applications (Second Edition). New York: Academic Press, Inc., 1980.

20. Swain, Ralph W. "Microcomputers in the Classroom," TIMS-ORSA Meeting, Detroit, Michigan, 21 April, 1982.

21. Whitehouse, G. E. and Yassar Hosni. "Use of Microcomputers to Solve Industrial Engineering and Operations Research Problems," TIMS-ORSA Meeting, Detroit, Michigan, 21 April, 1982.

22. Wu, Nesa and Richard Coppins. Linear Programming and Extensions. New York: McGraw-Hill Book Company, 1981.

APPENDIX A

USERS GUIDE

CONTENTS

I Introduction

The primary emphasis in the main body of this thesis has been the design considerations and the method of implementation of this linear programming software package. Specific comments on the use of the package have been limited; therefore, it is the purpose of this appendix to provide the user with specific information and guidance on the use of various programs included in this LP software package. Included in this section are the specific capabilities, limitations, methods of access, and use of each program. Also, suggestions are provided which will reduce the quantity of input required by the user of these programs.

An Apple II microcomputer with 48K RAM is required for the use of this software as implemented. Also required are a 16K memory expansion card, at least one 5 1/4 inch disk drive, and a monitor. Although one disk drive is sufficient, a second drive requires significantly fewer disk changes during problem solution and speeds the process considerably. Also, note that the instructions and suggestions in this appendix apply to the Apple II microcomputer and may not be applicable to other systems.

II Capabilities and Limitations

This section specifically outlines the type of linear programming problems which may be solved and analyzed with this software. Limitations exist and they will be explicitly noted below.

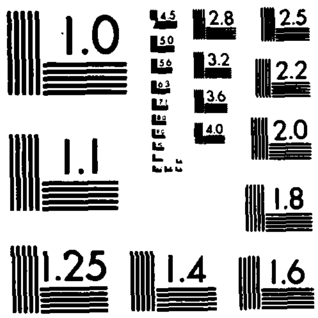
Capabilities

This linear programming software package allows the user to interactively enter LP models with the variables and constraints identified only by numerical identifiers or with names associated with each. These models may be stored to a disk for later review, editing, and solution. Extensive editing features of all entries allow correction during data input or editing at a later time. These editing features also allow the user to input a model, save this model to a disk, then change selected parameters to form new LP models. After the model has been input, it may be displayed for review on the screen or output to a printer. All displays of output, prompts, or comments are screen-oriented with pauses inserted at the end of each screen display to allow inspection of the information prior to proceeding to the next display.

The remaining programs of this package are dedicated to problem solution and sensitivity analysis. Module 2, the instructional module, allows the user to input option selections regarding the application of the simplex

AD-A124 804 FORTRAN BASED LINEAR PROGRAMMING FOR MICROCOMPUTERS(U) 2/5
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING T R FRALEY ET AL. DEC 82

UNCLASSIFIED AFIT/GOR/OS/82D-4 F/G 12/1 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

algorithm and receive feedback concerning the validity of the option selection. Areas in which instructional comments and guidance have been provided are tableau formulation, pivot element selection, and identification of optimal, infeasible, or unbounded solutions. Options have been provided which allow the user to specify whether to perform and receive feedback on each of the areas or to allow the program to perform the functions without user input. This permits the program to require varying degrees of user interface and corresponding degrees of feedback and instructional aid.

The instructional module only allows the solution of the primal problem of the LP model; however, it permits this solution by either the regular or the dual simplex method. When the solution process is performed using the dual simplex procedure, instructional comments similiar to those available with the regular simplex procedure are presented.

The third module, annotated as the problem solver module, allows the user to solve either the primal or the dual problem of the LP model using the regular or the dual simplex procedure. This expanded capability allows the user to specify the most efficient method of problem solution. Once the type of problem and method of solution have been identified, the program performs all formulation, pivot element selection, and identification of the final solution. Prior to the solution process, the user may specify the tableaus to be displayed. Also, the numerical format and location of output is user-defined.

The last module provides the sensitivity analysis capabilities of this package. This module requires that an LP problem have been solved using either problem solving module and the solution parameters saved to disk. The user may then select one of five alternatives offered for sensitivity analysis. Resource constraint right-hand-side ranging within the present optimal solution is provided as well as constraint and cost coefficient ranging. The user is also able to change one or any combination of the original model parameters and proceed to the corresponding new final solution. This solution may be displayed in the same tabular form as discussed earlier. Two more features of the sensitivity analysis module are the ability to add a variable or a constraint to the LP model and the ability to obtain the corresponding solution. This capability allows the user to examine modified examples of the original problem without resorting to model editing and solution with Modules 1 and 2 or Modules 1 and 3.

Limitations

The limitations associated with this package are primarily a result of implementation on a microcomputer rather than a large, stationary system. The principle areas of concern are the size of the LP model and the number of digits in the various parameters of the model.

Due to memory restrictions of the Apple microcomputer, as well as most comparably sized microcomputers, the number of constraints and variables are limited in this software implementation. The maximum number of constraints and variables which the user may input has been set to:

Maximum number of constraints: 20
Maximum number of variables: 20

Note that since an augmented basis is used in problem solution, this translates to a possible twenty constraint, sixty variable problem for solution.

The other area in which limitations exist is the number of significant digits which may be accurately maintained by the microcomputer. A maximum of ten digits may be entered as coefficients and resource limits; however, any number of significant digits larger than six may be subject to round-off during computation and display. The ability to enter ten digits has been allowed since the user may need to enter a negative sign and decimal point in addition to the significant digits. Therefore, the user should be cautious in the use of numbers which have six or more significant digits.

III User Input

The amount of user input differs among the various modules of this package; however, the amount required is extensive in the data entry and instructional portions of the software. Therefore, minimization of input is necessary whenever possible. Listed below are several suggestions, as well as warnings, regarding user input.

Option Selection: The user will be required to make several option selections requiring one or two digit inputs or single character inputs. An error during this input may have one of two results. If the user inputs a valid entry (one of the possible alternatives) but not the desired option designator, the user will not be able to prevent execution of the selected alternative. If the entry is not a valid alternative, the program will inform the user of an invalid entry and the proper option designator may be entered.

Model Parameters: All numeric parameters (cost and constraint coefficients, constraint right-hand sides) are checked for invalid characters in the input. Any non-numeric input except sign designators and decimal points will cause an invalid entry display followed by a request for the user to reenter the data. Due to this check of input, commas should not be entered (i.e. enter 10000, not

10,000). Positive values are assumed unless a negative sign is input. Also, all real numbers which contain no significant digits to the right of the decimal may be entered as integers (i.e. 1.0 may be entered as 1).

Inequality Input: Less-than or equal (≤) must be input as less-than [<] inequalities while greater-than or equal (≥) must be entered as greater-than inequalities [>].

Yes/No Inputs: All [YES] or [NO] responses may be entered by a [Y] or [N] single entry.

All Inputs: All user inputs must be completed by depressing the [RETURN] key. The computer does not attempt to read input until this action is taken. A zero may be entered as a numeric input by pressing only the [RETURN] key; this allows faster input of a model which has many zero coefficients.

Printer Option: This software package offers the user the capability to have selected output routed to a printer; however, the selection of this option without a printer being available will cause an execution error. Also, if the printer option is selected, the user must insure that the printer has been turned on and is in a printing mode. Otherwise, the system will wait indefinitely for the printer to accept information, and it will "hang" the system.

IV. System Startup

The compiled FORTRAN code files which form this LP package have been stored on two 5 1/4 inch floppy disks with volume names LP1 and LP2 (see Appendix B for disk file structure). As was discussed in Chapter IV, a turnkey system has been used which causes Module 1, the data base entry module, to execute automatically when the computer is turned on. However, for this system to function properly, two data files must be placed on the disk LP1 prior to its use. These files, SYSTEM.PASCAL and SYSTEM.MISCINFO, must be transferred from disk APPLE1 (Version 1.0) to LP1 for the turnkey system to operate. For those who are not familiar with the procedures required to transfer files, please refer to the operating system reference manual (Ref 1:156).

Once the transfers have been completed, the following steps should be performed to allow Module 1 to execute.

1. Place disk APPLE1 (Version 1.0) in disk drive #4 (the first disk drive is numbered as #4 with the Pascal Operating System while the second drive, if present, is numbered #5). If two drives are present, also place disk LP2 in drive #5.
2. Turn on the power switch of the Apple-II (#4 drive should activate and run for approximately five seconds).

3. Remove APPLE1 from #4 drive and replace it with LP1.

4. Press the [RESET] key (#4 drive should again run, followed by #5, if a second drive present, and then the #4 drive runs again).

At this point, a title page should appear on the screen and Module 1 has executed.

For those users not familiar with this software, the next section is devoted to an example problem and explanation. The problem demonstrates the use of all modules and may be of assistance in learning the various alternatives available and their method of activation.

V Sample Problem

This section is dedicated to a step-by-step guide through the major portions of this software package. Users may review this section to learn the specific alternatives available and their sequence of availability. It would not be feasible to demonstrate all of the possible alternatives and their use; however, the sequence and methods felt to be most commonly used will be shown with explanatory comments given for many of the other features available.

Numerical Example

The following numerical example will be used for the purposes of this demonstration. The example, although not a large problem, will allow the user to become familiar with the method of data entry and the output formats used in this LP package.

Problem: An analyst has been asked to determine the number of helicopters, by type, which would be required to move at least 2000 men and 1200 tons of equipment to a new area of operation. Three types of helicopters are available with the following capabilities:

	FUEL COST($100)	MEN	EQUIP(tons)
TYPE1	30	5	6
TYPE2	22.5	8	3
TYPE3	25	4	6

The analyst has been directed to determine the mix of aircraft which minimizes cost; however, the analyst has been restricted to the use of 400 helicopters (Ref 6:75).

The above situation may be formulated as a linear programming problem as follows:

x(1) = number of TYPE1 aircraft
x(2) = number of TYPE2 aircraft
x(3) = number of TYPE3 aircraft

$$\text{minimize } 30\,x(1) + 22.5\,x(2) + 25\,x(3)$$

subject to

$$5\,x(1) + 8\,x(2) + 4\,x(3) \geq 2000$$

$$6\,x(1) + 3\,x(2) + 4\,x(3) \geq 1200$$

$$x(1) + x(2) + x(3) \leq 400$$

Module Demonstration

In the demonstration which follows, all inputs by the user have been placed in square brackets ([ ]) to distinquish input from screen displays and prompts. All user inputs are completed by depressing the [RETURN] key to allow the program to read this input. Another area which may require clarification is the PAUSE statement which appears after displays not requiring data input. The user must depress the [RETURN] key or the [SPACE] bar whenever a PAUSE statement appears before the program will continue. The input of any other type than the two stated will cause a new PAUSE statement to appear.

Module 1. The first step required in the use of this LP software package is system initialization which is performed as described in Part IV of this appendix, System Startup. Once those procedures have been completed, the following title page will be displayed.

```
              **********************
              *   FORTRAN BASED    *
              *                    *
              * LINEAR PROGRAMMING *
              *                    *
              *        FOR         *
              *                    *
              *   MICROCOMPUTERS   *
              **********************


                        BY

              THEODORE R. E. FRALEY

                        AND

                   DALE A. KEM

       ARE INTRODUCTORY REMARKS DESIRED?
                 (Y/N, RETURN) [Y]
```

Should one wish to review the introductory remarks, a [Y] would be entered (remember that the brackets indicate user input followed by a [RETURN]) and the following comments will be displayed. Note that in the lower left corner of each display the word "PAUSE" appears. This indicates that no further displays or program progress will occur until the [SPACE] bar or [RETURN] key have been depressed.

```
  LINEAR PROGRAMMING SOFTWARE PACKAGE

THIS PACKAGE IS DESIGNED TO ALLOW
STUDENTS TO IMPROVE THEIR UNDERSTANDING
OF THE SIMPLEX ALGORITHM AND ALSO TO
PROVIDE THE MANAGERS AND ANALYSTS WITH A
PROBLEM SOLVING TOOL.

THE PACKAGE CONSISTS OF FOUR DISTINCT
PROGRAMS (ANNOTATED AS MODULES) WHOSE
FUNCTIONS ARE AS FOLLOWS:

     MODULE 1: DATA BASE ENTRY
     MODULE 2: LP INSTRUCTION
     MODULE 3: LP PROBLEM SOLVER
     MODULE 4: SENSITIVITY ANALYSIS

ALL LP PROBLEMS MUST BE ENTERED INTO A
DATABASE USING MODULE 1.  MODULES 2 OR
3 MAY BE USED TO DETERMINE A SOLUTION TO
A PROBLEM AND THIS MUST OCCUR PRIOR TO
ENTERING MODULE 4.

PAUSE
```

```
INSTRUCTIONS ON HOW TO ENTER EACH MODULE
WILL BE PRESENTED WHEN APPROPRIATE.

ANSWERS TO SPECIFIC QUESTIONS CONCERNING
ANY MODULE WILL BE FOUND IN THE USERS
GUIDE (APPENDIX A) OF THE THESIS
DOCUMENTATION.

ALL RESPONSES REQUIRE A [RETURN] TO NOTE
THE COMPLETION OF INPUT.

ALSO, ALL YES/NO INPUTS MAY BE ENTERED
BY [Y] OR [N], RESPECTFULLY.

PAUSE
```

After the introductory remarks, the first menu displayed is that of module selection. Since a data base has not previously been entered, one would select option [1] to continue in Module 1 and data base entry.

```
                MODULE SELECTION

THE FOLLOWING OPTIONS ARE AVAILABLE:

1. DATA BASE ENTRY (ENTER DATA BASE OR
   EDIT CURRENT DATA BASE)

2. LP INSTRUCTIONAL MODULE

3. LP PROBLEM SOLVER MODULE

4. LP SENSITIVITY ANALYSIS MODULE

(NOTE: OPTIONS 2 OR 3 REQUIRE THAT A
DATA BASE BE CURRENTLY STORED ON DISK)

(NOTE: OPTION 4 REQUIRES THAT A DATA
FILE HAS BEEN SAVED UPON LEAVING THE
OPTION 2 OR 3 MODULES ABOVE.)

             WHICH OPTION? [1]
```

The following header confirms entry into the data base entry portion of Module 1.

```
         **********************
         *                    *
         *       DATA         *
         *                    *
         *       BASE         *
         *                    *
         *       ENTRY        *
         *                    *
         *      MODULE        *
         *                    *
         *                    *
         *     MODULE 1       *
         *                    *
         **********************

PAUSE
```

Following the data base entry header, the user is offered several alternatives of entering a data base. Again, no previous data base exists so either option 1 or 2 must be selected. Option [2] allows the same input as option 1, but allows the constraints and variables to be identified by names as well as subscripts.

```
                DATA BASE ENTRY

          TO ENTER LP MODEL DATA BASE
       YOU HAVE THE FOLLOWING OPTIONS:


1. CREATE MODEL INTERACTIVELY:SUBSCRIPTS
   (VARIABLES ANNOTATED BY SUBSCRIPTS,
   CONSTRAINTS ANNOTATED BY NUMBER ONLY)

2. CREATE MODEL INTERACTIVELY:NAMED
   VARIABLES AND CONSTRAINTS ARE
   ASSIGNED NAMES)

3. READ FROM DISK
   (PREVIOUSLY CREATED BASE)

4. DISPLAY INTRODUCTORY REMARKS

5. QUIT PROGRAM

               WHICH OPTION? [2]
```

If a data base had been previously entered and saved to disk, one could select option 3. At that time, the following three displays of comments and prompts would appear. If this previous file had been placed on disk LP1 (volume name) with a filename of BUSES, one would enter this as shown below to retrieve that model. The next two prompts appear sequentially to inform the user that the designated disk must be in the #4 disk drive for a one drive system or in either drive for a multiple drive system. A habit of leaving disk LP1 in drive #4 and disk LP2 in drive #5 has been found to be beneficial for a two drive system. In this configuration, only when files are to be written or read from another disk are the two disks not accessible in the disk drives.

```
                READ LP MODEL FROM DISK


ENTER THE DISK DRIVE NUMBER AND FILE
NAME WHICH HOLDS THE MODEL DESIRED.

                ENTER EXACTLY AS FOLLOWS
                  DISK DRIVE:FILENAME

                      EG.   #4:FILENAM

               DISK:FILENAME = [LP1:BUSES]
```

```
             INSURE THE DISK CONTAINING THE

                         LP1:MAX

                   MODEL IS AVAILABLE.

PAUSE
```

```
 INSURE DISK    LP1    IS AVAILABLE.

PAUSE
```

Continuing with the data base entry for the example problem, one would enter an identifier for later reference. Also, the type of problem is entered at this time as shown below.

```
            ENTER A PROBLEM IDENTIFIER
            (MAXIMUM OF 20 CHARACTERS)

       PROBLEM ID = [SAMPLE PROBLEM]
****************************************

 IS PROBLEMS OBJECTIVE FUNCTION TO BE:

                1. MAXIMIZED
                     OR
                2. MINIMIZED

              WHICH OPTION? [2]
```

Next, one identifies the objective name of the problem. This input is not allowed for models with only subscript designators.

```
WHAT IS THE NAME OF THE OBJECTIVE YOU
          WANT TO MINIMIZE?
(FOR EXAMPLE, COST, MANPOWER,  ETC.)

MAXIMUM OF 10 CHARACTERS ALLOWED

     OBJECTIVE NAME = [COST]
```

At this point, the number of constraints and variables which form the LP model are entered. Nonnegativity constraints are not to be included in the number of constraints entered below.

```
ENTER NUMBER OF CONSTRAINTS IN PROBLEM
            (MAXIMUM OF 20)

      NUMBER OF CONSTRAINTS = [3]
****************************************
 ENTER NUMBER OF VARIABLES IN PROBLEM
            (MAXIMUM OF 20)

       NUMBER OF VARIABLES = [3]
```

Since a model with names has been selected, the following three displays are presented allowing the input and correction of the decision variable names. First the names are input and in the next display, a [Y] may be entered to allow correction of the names. An [N] has been entered since corrections were not needed.

```
          VARIABLE NAME INPUT

 ENTER VARIABLE NAMES WHICH CORRESPOND
   TO THE  3 VARIABLES THAT AFFECT
               COST

NAMES ARE TO BE 6 CHARACTERS OR LESS.

PAUSE
```

```
     PROBLEM ID: SAMPLE PROBLEM
          VARIABLE NAME INPUT

             X( 1) = [TYPE1]
             X( 2) = [TYPE2]
             X( 3) = [TYPE3]
```

```
PROBLEM ID: SAMPLE PROBLEM
    VARIABLE NAME INPUT


  ARE CORRECTIONS NEEDED? [N]
```

The same sequence appears next for the input and correction of constraint names. Note that variable and constraint name input is not allowed for models with subscripts only. Although these names are useful for ease of variable identification, input of the data base is slowed considerably by this requirenment.

```
          CONSTRAINT NAME INPUT

ENTER CONSTRAINT NAMES WHICH CORRESPOND
TO THE  3 CONSTRAINTS WHICH AFFECT
COST

NAMES ARE TO BE 6 CHARACTERS OR LESS.

PAUSE
```

```
PROBLEM ID: SAMPLE PROBLEM
    CONSTRAINT NAME INPUT

    CONSTRAINT  1 = [PERSON]
    CONSTRAINT  2 = [EQUIP]
    CONSTRAINT  3 = [PLANES]
```

```
PROBLEM ID: SAMPLE PROBLEM
    CONSTRAINT NAME INPUT


  ARE CORRECTIONS NEEDED? [N]
```

Next, the objective function is input following comments regarding the restrictions on numerical input. After completion of input, the option of correcting input is presented. This allows changes of any type to the objective function. Note that the coefficients input below do not include decimals for those without significant digits to the right of the decimal. This allows for more rapid input of the coefficients values.

```
OBJECTIVE FUNCTION INPUT

INPUT THE FUNCTION AS IF IT WERE IN THE
FOLLOWING FORM
Z = X(1) + X(2) + X(3) + ETC.

A MAXIMUM OF 10 ENTRIES PER COEFFICIENT
INCLUDING DECIMAL AND SIGN ARE ALLOWED.

IF COEFFICIENT IS ZERO, HIT "RETURN"
WITHOUT DIGIT ENTRY.

PAUSE
```

```
PROBLEM ID: SAMPLE PROBLEM
  OBJECTIVE FUNCTION INPUT
 COST        MINIMIZATION

 C( 1) = TYPE1  = [30]
 C( 2) = TYPE2  = [22.5]
 C( 3) = TYPE3  = [25]
```

No corrections were needed so in the objective function an [N] was entered below.

```
PROBLEM ID:  SAMPLE PROBLEM
  OBJECTIVE FUNCTION INPUT
 COST        MINIMIZATION


 ARE CORRECTIONS NEEDED? [N]
```

The constraint coefficients are input next in the same type sequence as the objective function. One constraint is entered per display with the option of corrections being presented after the last constraint has been entered. This editing allows the user to change any constraint and any part of the input, including the inequality. Note that the nonnegativity constraints are not entered into the model. Also, note that the constraint coefficients are input as integers since no significant digits exist to the right of the decimal point.

```
              CONSTRAINT  INPUT

INPUT CONSTRAINT VARIABLE COEFFICIENTS
AS IF THE CONSTRAINT WAS IN THE
FOLLOWING FORM
      X(1)  + X(2)  + X(3)   <=> RHS

THE VARIABLE COEFFICIENTS ARE A MAXIMUM
OF 10 CHARACTERS

IF COEFFICIENT IS ZERO, ENTER 0 OR HIT
"RETURN" WITHOUT ENTRY.

THE LESS-THAN (<) REPRESENTS A LESS-THAN
OR EQUAL INEQUALITY.

THE GREATER-THAN (>) REPRESENTS A
GREATER-THAN OR EQUAL INEQUALITY.

NEGATIVE RHS IS PERMITTED.

PAUSE
```

```
     PROBLEM ID: SAMPLE PROBLEM
        CONSTRAINT   1 = PERSON

      X( 1) = TYPE1   = [5]
      X( 2) = TYPE2   = [8]
      X( 3) = TYPE3   = [4]
      INEQUALITY     [>]
      RHS              = [2000]
```

```
     PROBLEM ID: SAMPLE PROBLEM
        CONSTRAINT   2 = EQUIP

      X( 1) = TYPE1   = [6]
      X( 2) = TYPE2   = [3]
      X( 3) = TYPE3   = [6]
      INEQUALITY     [>]
      RHS              = [1200]
```

```
PROBLEM ID: SAMPLE PROBLEM
   CONSTRAINT  3 = PLANES

  X( 1) = TYPE1  = [1]
  X( 2) = TYPE2  = [1]
  X( 3) = TYPE3  = [1]
  INEQUALITY    [<]
  RHS            = [400]
```

```
ARE CORRECTIONS NEEDED?  [N]
```

Following the input and correction of the constraints, the data base management menu is displayed. Option 1 allows the user to review input to insure correctness, after which the data base management menu appears again. The user may then select to edit any parameters with option 2. Once the model is corrected, it may be saved to disk by selecting option [3].

```
              DATA BASE MANAGEMENT

THE FOLLOWING OPTIONS ARE AVAILABLE:
1. DISPLAY CURRENT LP MODEL
   (SCREEN OR PRINTER)

2. EDIT CURRENT LP MODEL
   (CHANGE ANY PARAMETER)

3. SAVE CURRENT MODEL TO DISK FILE
   (MAY THEN EDIT TO ANOTHER MODEL)

4. ENTER NEW MODEL
   (CURRENT MODEL LOST IF NOT ON DISK)

5. SOLVE PROBLEM
   (INCLUDES EDUCATIONAL, PROBLEM SOLVER,
    AND SENSITIVITY ANALYSIS)

6. QUIT:UNSAVED FILES DESTROYED!

                 WHICH OPTION? [3]
```

The next four displays pertain to saving the model to a disk. The user first enters the diskname:filename which the current model is to be saved under. Next, the user is given the opportunity to correct this diskname. The user is then directed to insure that the specified disk is in a disk drive. Next, the user is required to input the status of the diskname:filename combination. If one has previously saved a model to the same diskname:filename combination, but answers [N] in the third display, all input data will be lost. This will then require reinitialization of the system and reentry of the last data base.

```
              SAVE LP MODEL TO DISK


   ENTER THE DISK DRIVE NUMBER AND FILE
        NAME WHICH YOU WANT PROBLEM
            SAMPLE PROBLEM
                SAVED UNDER.

          ENTER EXACTLY AS FOLLOWS
            DISK DRIVE:FILENAME

               EG.    #4:FILENAM

THE DRIVE:FILENAME MUST BE 10 CHARACTERS
                  OR LESS

IF THE ABOVE IS ENTERED INCORRECTLY,
       YOUR MODEL WILL BE LOST!!

       DISK:FILENAME = [LP1:SAMPLE]

       ARE CORRECTIONS NEEDED? [N]
```

```
INSURE THE DISK TO CONTAIN THE FILE

LP1:SAMPLE

IS AVAILABLE.

PAUSE
```

```
HAS THIS DISK:FILENAME COMBINATION BEEN
USED PREVIOUSLY?

(ARE YOU UPDATING A CURRENTLY EXISTING
FILE?)

(Y/N) [N]
```

```
INSURE DISK    LP1    IS AVAILABLE.
```

After the model has been saved, control returns to the data base management menu as shown below. At that point, one may select option 2 and the data base editor menu would be displayed as shown below the data base management menu.

```
              DATA BASE MANAGEMENT

THE FOLLOWING OPTIONS ARE AVAILABLE:
1. DISPLAY CURRENT LP MODEL
   (SCREEN OR PRINTER)

2. EDIT CURRENT LP MODEL
   (CHANGE ANY PARAMETER)

3. SAVE CURRENT MODEL TO DISK FILE
   (MAY THEN EDIT TO ANOTHER MODEL)

4. ENTER NEW MODEL
   (CURRENT MODEL LOST IF NOT ON DISK)

5. SOLVE PROBLEM
   (INCLUDES EDUCATIONAL, PROBLEM SOLVER,
    AND SENSITIVITY ANALYSIS)

6. QUIT:UNSAVED FILES DESTROYED!

                 WHICH OPTION? [5]
```

```
                DATA BASE EDITOR

YOU MAY EDIT THE CURRENT MODEL IN ANY OF
THE FOLLOWING MANNERS:
1.   ADD A VARIABLE
2.   ADD A CONSTRAINT
3.   DELETE A VARIABLE
4.   DELETE A CONSTRAINT
5.   CHANGE COEFFICIENT BY CONSTRAINT
6.   CHANGE COEFFICIENTS BY VARIABLE
7.   CHANGE RHS OF CONSTRAINT
8.   CHANGE CONSTRAINT INEQUALITY
9.   CHANGE OBJECTIVE FUNCTION COST
     COEFFICIENTS
10.  CHANGE MAXIMIZATION/MINIMIZATION
     CHOICE
11.  CHANGE VARIABLE NAMES
12.  CHANGE CONSTRAINT NAMES
13.  RETURN TO LAST MENU
     (DATA BASE MANAGEMENT)

                 WHICH OPTION? [13]
```

As shown, extensive editing features are available and could also have been used prior to saving the model to disk. These features could also be used to form a new LP model by changing selected parameters of the model in memory. Upon completion of the editing, one may return to the data base management menu by selecting option [13].

Option 4 of the data base management menu above could have been selected if the user wished to enter a new model after saving the first model to disk. The user , if attempting to enter a new model prior to saving the first to disk, receives a warning that the first model must be saved or will be lost.

Option [5] has been selected in the last data base management menu signifying that the problem is now to be solved. Again, if this option had been selected prior to saving the last entered model to disk, a warning message would have appeared and the user would have been allowed to save the last model before progressing to problem solution.

Following the selection to solve the problem, the following menu, execution management, is displayed. Option [1] has been selected for this demonstration; however, option 2 could also have been selected. Option 3 is not possible at this time since a problem must have previously been solved with Module 2 or Module 3 and the results saved to disk. The user is also permitted to return to the data entry section prior to exiting this module. The user, upon selection of option 1, 2, or 3, is prompted to insure a required disk is accessible.

```
                EXE/UTION MANAGEMENT

THE FOLLOWING OPTIONS ARE AVAILABLE:

1. LP INSTRUCTIONAL MODULE
   (EACH TABLEAU MAY BE DISPLAYED)

2. PROBLEM SOLVER MODULE
   (NO USER INTERACTION)

3. SENSITIVITY ANALYSIS MODULE

4. RETURN TO DATA BASE MANAGEMENT MENU

5. QUIT:UNSAVED FILES WILL BE LOST!

                   WHICH OPTION? [1]
```

```
 INSURE DISK    LP1    IS AVAILABLE.

PAUSE
```

After the selection of a module, the following display allows the user to specify a data base other than the last entered for solution by the selected selected module. One may enter [N] in the response below and is then allowed to specify a file as was done earlier for reading a file from disk. As shown, the last file entered is desired so a [Y] was entered. The last display of this module then appears.

```
            LP INSTRUCTIONAL MODULE

TO USE THIS MODULE, A DATA BASE MUST
HAVE BEEN PREVIOUSLY CREATED USING THE
DATA BASE ENTRY (MODULE 1) AND SAVED TO
DISK.

THE DATA BASE WHICH IS CURRENTLY
IDENTIFIED AS THE PROBLEM TO BE STUDIED
IS:
                 LP1:SAMPLE


IS THIS THE MODEL YOU WISH TO STUDY? [Y]
```

Simultaneously with the display below, the control returns to the operating system. The user need only enter [X] (no [RETURN] key should be used after the input of [X]). Next, the user enters [LP1:ED] and Module 2, which contains the instructional code, will be executed. If the user had selected option 2 of the execution management menu, the only difference would have been that [LP2:TAB] would have replaced [LP1:ED] above. One need not memorize this fact since the prompt will reflect the option input by the user.

As a final note, users must insure that the disk containing the desired module is in a disk drive when the above operating system commands are entered. The user may identify the required disk by the LP1 or the LP2 preceeding the colon and filename entered to the operating system. This will coincide with one of the two disks provided with this software package.

TO ENTER THE LP INSTRUCTIONAL MODULE:

TYPE

X
LP1:ED

Module 2. The last entries of the Module 1 demonstration were [X] and [LP1:ED]. These are the operating systems commands which direct the execution of the code file ED on disk LP1. Therefore, disk LP1 must be accessible when these commands are entered. Once these steps have been completed, the following header appears which confirms the execution of Module 2, the instructional module.

```
                    **********************
                    *                    *
                    *       LINEAR       *
                    *                    *
                    *    PROGRAMMING     *
                    *                    *
                    *    EDUCATIONAL     *
                    *                    *
                    *       MODULE       *
                    *                    *
                    *                    *
                    *       MODULE 2     *
                    *                    *
                    **********************
PAUSE
```

Following the header, the default options are displayed as shown below. On the right of the display, the options which have been programmer defined are shown. These options control the extent of user interface required in the instructional areas as well as the method of solution. Options 5 through 7 allow the user to designate those tableaus to be displayed as well as their location and format. Those options desired to be changed may be selected in any order. The number of alternatives available within the options vary; however, several options have only two possibiliities (options 3, 6, and 7). These options require only that they be selected while the other options require further input to cause a change in their value. After the selection and change of any option, the default values are again displayed with the noted changes. When the user is satisfied with the defaults as displayed, option 8 should be selected to allow the program to continue. The major options and their alternatives are shown in the following sequence.

For instance, if the user would like to review the alternatives available under option 1, [1] would be input as shown below.

```
                DEFAULT OPTIONS
         ENTER OPTION NUMBER TO CHANGE
1.TABLEAU FORMATION                      USER
2.PIVOT ELEMENT SELECTION            USER SEL
                                     ALGOR CHK

3.DUAL PIVOTS                               N
4.INFEASIBLE,UNBOUNDED,OPTIMAL
    SELECTION IDENTIFICATION             USER

5.TABLEAUS TO BE DISPLAYED
    INITIAL                                 Y
    INTERMEDIATE                        N =  1
    FINAL                                   Y
6.OUTPUT LOCATION                      SCREEN

7.OUTPUT FORMAT                      F FORMAT
8.NO CHANGES

*SEE DOCUMENTATION FOR EXPLANATION

       WHICH OPTION (ENTER 1-8)? [1]
```

The display below describes the alternatives available for this area of the instructional aid. Option [1] was selected as shown below which will allow instructional comments to be provided for the user. Note that this was the same alternative as was originally designated as a default.

```
 EDUCATIONAL MODULE OPTION SELECTION

IN ORDER TO PLACE THE LP MODEL INTO THE
PROPER FORM FOR THE SIMPLEX ALGORITHM
(OBJECTIVE FUNCTION CHANGES, ADDITION OF
SLACK OR ARTIFICAL VARIABLES), WHICH
METHOD IS DESIRED?

1. USER SELECTS MODIFICATION AND
   ALGORITHM CHECKS

                   OR

2. ALGORITHM PERFORMS MODIFICATIONS.
   (NO USER INPUT)

              WHICH OPTION? [1]
```

The default menu then appears and the next option to be changed may be entered. Option [2] was selected as the next to be changed.

```
        DEFAULT OPTIONS
ENTER OPTION NUMBER TO CHANGE
1.TABLEAU FORMATION                     USER
2.PIVOT ELEMENT SELECTION           USER SEL
                                   ALGOR CHK

3.DUAL PIVOTS                              N
4.INFEASIBLE,UNBOUNDED,OPTIMAL
    SELECTION IDENTIFICATION            USER

5.TABLEAUS TO BE DISPLAYED
    INITIAL                                Y
    INTERMEDIATE                       N = 1
    FINAL                                  Y
6.OUTPUT LOCATION                     SCREEN

7.OUTPUT FORMAT                     F FORMAT
8.NO CHANGES

*SEE DOCUMENTATION FOR EXPLANATION

        WHICH OPTION (ENTER 1-8)? [2]
```

The user is now shown the alternatives available for the pivot element selection. Selections 2 and 3 do not provide feedback to the user; however, the use of selection 2 may be beneficial to instructors who wish to demonstrate the outcome of an inappropriate pivot element selection. Option [1] has been selected below.

```
 EDUCATIONAL MODULE OPTION SELECTION

IN SELECTION OF PIVOT ELEMENTS FOR THE
SIMPLEX ALGORITHM, WHICH METHOD WOULD
YOU LIKE?

1. USER SELECTS, ALGORITHM CHECKS.
   (MAY CHANGE SELECTION AFTER CHECK)

2. USER SELECTS, NO ALGORITHM CHECK.

3. ALGORITHM SELECTS, NO USER INPUT.

              WHICH OPTION? [1]
```

The default menu would again appear (not shown) and option [4] could be input, resulting in the following display. Again, the user may select the degree of user interaction desired. Option [1] has been selected to allow the demonstration of the instructional comments available. This option requires the user to respond to questions concerning the status of the tableau while option 2 would have allowed the program to perform these actions without user interface.

```
 EDUCATIONAL MODULE OPTION SELECTION

AS OPTIMAL, INFEASIBLE, OR UNBOUNDED
SOLUTIONS OCCUR, WHICH METHOD WOULD YOU
LIKE?

1. USER ATTEMPTS TO IDENTIFY, ALGORITHM
   CHECKS.

2. SYSTEM IDENTIFIES AND REPORTS AS
   OCCURS.

                WHICH OPTION? [1]
```

The default menu is displayed reflecting any changes which have been requested. Since the user may have no initial feeling for the magnitude of the final solution values, option [7] has been input to allow larger numbers to be displayed.

```
             DEFAULT OPTIONS
      ENTER OPTION NUMBER TO CHANGE
1.TABLEAU FORMATION                  USER
2.PIVOT ELEMENT SELECTION        USER SEL
                                 ALGOR CHK

3.DUAL PIVOTS                           N
4.INFEASIBLE,UNBOUNDED,OPTIMAL
    SELECTION IDENTIFICATION         USER

5.TABLEAUS TO BE DISPLAYED
    INITIAL                             Y
    INTERMEDIATE                    N = 1
    FINAL                               Y
6.OUTPUT LOCATION                  SCREEN

7.OUTPUT FORMAT                  F FORMAT
8.NO CHANGES

*SEE DOCUMENTATION FOR EXPLANATION

       WHICH OPTION (ENTER 1-8)? [7]
```

The option 7 selection changes the default to "E FORMAT" as shown above. Now that all desired changes have been made, option [8] is entered as shown and the program continues.

```
              DEFAULT OPTIONS
       ENTER OPTION NUMBER TO CHANGE
1.TABLEAU FORMATION                     USER
2.PIVOT ELEMENT SELECTION           USER SEL
                                    ALGOR CHK

3.DUAL PIVOTS                              N
4.INFEASIBLE,UNBOUNDED,OPTIMAL
   SELECTION IDENTIFICATION             USER

5.TABLEAUS TO BE DISPLAYED
   INITIAL                                 Y
   INTERMEDIATE                        N =  1
   FINAL                                   Y
6.OUTPUT LOCATION                     SCREEN

7.OUTPUT FORMAT                     E FORMAT
8.NO CHANGES

*SEE DOCUMENTATION FOR EXPLANATION

      WHICH OPTION (ENTER 1-8)? [8]
```

The next three displays inform the user, and in particular the one-disk-drive system user, that a particular disk must be accessible before continuing.

```
 INSURE DISK    LP1     IS AVAILABLE.

PAUSE
```

```
     INSURE THE DISK CONTAINING THE

              LP1:SAMPLE

         MODEL IS AVAILABLE.
PAUSE
```

```
 INSURE DISK    LP1     IS AVAILABLE.

PAUSE
```

The following display informs the user of requirements regarding the objective function modification. If the user had selected the algorithm to perform the tableau modification, this sequence would not be displayed. The display following the instructions is the objective function as entered in Module 1.

```
    OBJECTIVE FUNCTION MODIFICATION

THE OBJECTIVE FUNCTION, AS ENTERED, WILL
BE DISPLAYED NEXT.  AFTER THE DISPLAY,
YOU WILL BE ASKED TO SELECT THE OPTION
WHICH WILL TRANSFORM THE OBJECTIVE
FUNCTION INTO THE PROPER TABLEAU FORM
FOR THE SIMPLEX ALGORITHM.
PAUSE
```

```
                 OBJECTIVE FUNCTION MODIFICATION
                   PRESENT OBJECTIVE FUNCTION

                            TYPE1          TYPE2          TYPE3
                            X( 1)          X( 2)          X( 3)
         MIN Z  = +  3.0000E+01 +  2.2500E+01 +  2.5000E+01

       PAUSE
```

The user has now reviewed the objective function and must select the designator for the option which will place the objective function in the standard LP form defined in Chapter II of the main body. As shown below, option [1] was selected and the feedback concerning this selection is presented. The user may then review the entered and the correct response to determine the reason for an incorrect input.

```
OBJECTIVE FUNCTION MODIFICATION

TO PLACE THE OBJECTIVE FUNCTION IN THE
PROPER FORMAT FOR THE SIMPLEX ALGORITHM
WHICH OF THE FOLLOWING SHOULD BE DONE?

1. ADD -C(J) TO BOTH SIDES OF EQUATION.

2. MULTIPLY EQUATION BY -1 AND THEN ADD
   -C(J) TO BOTH SIDES OF EQUATION.

3. NO CHANGES ARE NECESSARY.

      WHICH OPTION IS CORRECT? [1]

        OPTION #1 IS INCORRECT.

   THE PROPER RESPONSE WAS OPTION #2.

PAUSE
```

Regardless of user input, the objective function is properly modified and displayed as shown below. This allows the user to further review the objective function and its modification.

```
AFTER THE PROPER MODIFICATION, THE
   OBJECTIVE FUNCTION FORM IS:

                  TYPE1         TYPE2         TYPE3
                  X( 1)         X( 2)         X( 3)
MAX (-Z) +   3.0000E+01 +  2.2500E+01 +  2.5000E+01 =0

PAUSE
```

The constraint modification sequence is similiar to that shown above for the objective function. The user is given instructions followed by the display of the constraints as entered. After reviewing these constraints, the options which the user may later select are shown.

```
                CONSTRAINT MODIFICATION

THE CONSTRAINTS, AS ENTERED, WILL BE
DISPLAYED NEXT.  AFTER THE DISPLAY, YOU
WILL BE SHOWN EACH OF THE CONSTRAINTS
INDIVIDUALLY AND ASKED TO SELECT THE
OPTION WHICH TRANSFORMS THE CONSTRAINT
INTO THE PROPER SIMPLEX ALGORITHM FORM.

PAUSE
```

```
           SAMPLE PROBLEM
           CURRENT CONSTRAINTS

                 TYPE1        TYPE2        TYPE3          RHS
                 X( 1)        X( 2)        X( 3)
CN  NAME   ****************************************************************
 1 PERSON     5.00000E+00  8.00000E+00  4.00000E+00 >  2.00000E+03
 2 EQUIP      6.00000E+00  3.00000E+00  6.00000E+00 >  1.20000E+03
 3 PLANES     1.00000E+00  1.00000E+00  1.00000E+00 <  4.00000E+02

PAUSE
```

```
EACH CONSTRAINT WILL BE SEPARATELY
DISPLAYED, THEN THE FOLLOWING OPTIONS
WILL BE DISPLAYED FOR EACH CONSTRAINT.
YOU WILL SELECT THE OPTION WHICH WILL
PLACE THE CONSTRAINT IN THE PROPER
SIMPLEX ALGORITHM FORM.

1. ADD SLACK VARIABLE ONLY.

2. SUBTRACT SURPLUS VARIABLE, ADD
   ARTIFICAL VARIABLE.

3. ADD ARTIFICAL VARIABLE ONLY.

4. MULTIPLY BY -1, SUBTRACT SURPLUS
   VARIABLE, ADD ARTIFICAL VARIABLE.

5. MULTIPLY BY -1, ADD SLACK VARIABLE.

6. MULTIPLY BY -1, ADD ARTIFICAL
   VARIABLE.

PAUSE
```

The constraints are next shown separately. This allows the user to study the constraint individually. When the user is confident that the proper option is known, the [RETURN] key or [SPACE] bar is pressed allowing the display of the options.

```
                   TYPE1         TYPE2         TYPE3
                   X( 1)         X( 2)         X( 3)           RHS
CN# 1 PERSON  5.00000E+00  8.00000E+00  4.00000E+00 \ 2.00000E+03
PAUSE
```

The user inputs the desired option (option [2] in this case) and receives immediate feedback. The user then may continue on to the remaining constraints for the same sequence of steps.

```
                 CONSTRAINT # 1

1. ADD SLACK VARIABLE ONLY.

2. SUBTRACT SURPLUS VARIABLE, ADD
   ARTIFICAL VARIABLE.

3. ADD ARTIFICAL VARIABLE ONLY.

4. MULTIPLY BY -1, SUBTRACT SURPLUS
   VARIABLE, ADD ARTIFICAL VARIABLE.

5. MULTIPLY BY -1, ADD SLACK VARIABLE.

6. MULTIPLY BY -1, ADD ARTIFICAL
   VARIABLE.

                 WHICH OPTION? [2]

              OPTION #2 IS CORRECT.

PAUSE
```

```
                   TYPE1         TYPE2         TYPE3
                   X( 1)         X( 2)         X( 3)           RHS
 CN# 2 EQUIP    6.00000E+00   3.00000E+00   6.00000E+00 > 1.20000E+03

 PAUSE
```

Note that when an incorrect response is entered, the correct response is shown while the corresponding function of each response is still visible. In this manner, the user may review any mistakes made while still being able to review the available options. Also note that the constraints are modified properly regardless of user input.

```
                 CONSTRAINT # 2

1. ADD SLACK VARIABLE ONLY.

2. SUBTRACT SURPLUS VARIABLE, ADD
   ARTIFICAL VARIABLE.

3. ADD ARTIFICAL VARIABLE ONLY.

4. MULTIPLY BY -1, SUBTRACT SURPLUS
   VARIABLE, ADD ARTIFICAL VARIABLE.

5. MULTIPLY BY -1, ADD SLACK VARIABLE.

6. MULTIPLY BY -1, ADD ARTIFICAL
   VARIABLE.

                 WHICH OPTION? [1]

             OPTION #1 IS INCORRECT

        THE PROPER RESPONSE WAS OPTION #2

PAUSE
```

The third constraint is now examined in the same manner as the first two.

```
                  TYPE1         TYPE2         TYPE3
                  X( 1)         X( 2)         X( 3)             RHS
CN# 3 PLANES   1.00000E+00   1.00000E+00   1.00000E+00 < 4.00000E+02

PAUSE
```

```
                CONSTRAINT # 3

1. ADD SLACK VARIABLE ONLY.

2. SUBTRACT SURPLUS VARIABLE, ADD
   ARTIFICAL VARIABLE.

3. ADD ARTIFICAL VARIABLE ONLY.

4. MULTIPLY BY -1, SUBTRACT SURPLUS
   VARIABLE, ADD ARTIFICAL VARIABLE.

5. MULTIPLY BY -1, ADD SLACK VARIABLE.

6. MULTIPLY BY -1, ADD ARTIFICAL
   VARIABLE.

                WHICH OPTION? [1]


             OPTION #1 IS CORRECT.

PAUSE
```

Next, the user is presented instructions on the next sequence of steps. This is followed by the display of the tableau as it has been modified by the previous two sections, the objective function and constraint modification sections.

```
THE TABLEAU AS MODIFIED PREVIOUSLY,
WILL BE DISPLAYED.

YOU WILL THEN BE ASKED IF THE TABLEAU IS
IN THE CORRECT FORM FOR THE SIMPLEX
ALGORITHM.

PAUSE
```

```
                    TYPE1         TYPE2         TYPE3         SURPLS        SURPLS
                    X( 1)         X( 2)         X( 3)         X( 4)         X( 5)
OBJ FUNCTION -3.00000E+01 -2.25000E+01 -2.50000E+01  0.00000E+00  0.00000E+01
CN  NAME  VAR  ################################################################
 1 PERSON  7  5.00000E+00  8.00000E+00  4.00000E+00 -1.00000E+00  0.00000E+00
 2 EQUIP   8  6.00000E+00  3.00000E+00  6.00000E+00  0.00000E+00 -1.00000E+00
 3 PLANES  6  1.00000E+00  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00



                    SLACK         ARTIF         ARTIF         RHS
                    X( 6)         X( 7)         X( 8)
OBJ FUNCTION  0.00000E+00  0.00000E+00  0.00000E+00 =  0.00000E+00
CN  NAME  VAR  ################################################################
 1 PERSON  7  0.00000E+00  1.00000E+00  0.00000E+00 =  2.00000E+03
 2 EQUIP   8  0.00000E+00  0.00000E+00  1.00000E+00 =  1.20000E+03
 3 PLANES  6  1.00000E+00  0.00000E+00  0.00000E+00 =  4.00000E+02

PAUSE
```

The user is next asked for a response concerning the form of the tableau. This input is followed by immediate feedback concerning the accuracy of this input. Note that this and the subsequent displays concerning the form of the tableau would not have appeared if the user had allowed the program to perform tableau modification without user interface.

```
  IS THE TABLEAU IN THE PROPER FORM FOR
           THE INITIAL PIVOT? [Y]

       YOUR RESPONSE WAS INCORRECT.


ARTIFICAL VARIABLES HAVE BEEN ADDED, YET
THE OBJECTIVE FUNCTION HAS NOT BEEN
MODIFIED (BIG M) TO REFLECT THIS.

PAUSE
```

Further instructions are presented below followed by the display of the tableau as modified up to this point.

```
THE TABLEAU WILL BE DISPLAYED AND YOU
WILL BE ASKED TO IDENTIFY THOSE
VARIABLES WHICH THE BIG M METHOD IS TO
BE APPLIED.

PAUSE
```

```
        SAMPLE PROBLEM
      CURRENT LP MODEL: MAXIMIZE COST


                    TYPE1        TYPE2        TYPE3        SURPLS       SURPLS
                    X( 1)        X( 2)        X( 3)        X( 4)        X( 5)
OBJ FUNCTION -3.00000E+01 -2.25000E+01 -2.50000E+01  0.00000E+00  0.00000E+01
CN  NAME  VAR  ***********************************************************************
 1 PERSON  7  5.00000E+00  8.00000E+00  4.00000E+00 -1.00000E+00  0.00000E+00
 2 EQUIP   8  6.00000E+00  3.00000E+00  6.00000E+00  0.00000E+00 -1.00000E+00
 3 PLANES  6  1.00000E+00  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00



                    SLACK        ARTIF        ARTIF        RHS
                    X( 6)        X( 7)        X( 8)
OBJ FUNCTION  0.00000E+00  0.00000E+00  0.00000E+00 =  0.00000E+00
CN  NAME  VAR  ***********************************************************************
 1 PERSON  7  0.00000E+00  1.00000E+00  0.00000E+00 =  2.00000E+03
 2 EQUIP   8  0.00000E+00  0.00000E+00  1.00000E+00 =  1.20000E+03
 3 PLANES  6  1.00000E+00  0.00000E+00  0.00000E+00 =  4.00000E+02

PAUSE
```

The user is now asked to identify those variables which require modification by the Big M method. Note that the variable subscript is entered to designate the selected variable. The input of [6] below refers to variable x(6) of the previously displayed tableau. Again feedback and instructional comments are provided immediately. The following display also requires input similiar to that discussed above.

```
WHICH VARIABLES REQUIRE THE USE OF THE
           BIG M METHOD?
      (ENTER SUBSCRIPT VALUES)

         FIRST VARIABLE? [6]

      YOUR RESPONSE WAS INCORRECT.
 THE CORRECT RESPONSE WAS VARIABLE  7
THIS IS THE FIRST ARTIFICAL VARIABLE AND
  REQUIRES THE USE OF THE BIG M METHOD.

PAUSE
```

```
   VARIABLES  7 THRU X(?) REQUIRE THE
            BIG M METHOD

         LAST VARIABLE? [8]

        YOUR RESPONSE WAS CORRECT.
 THE LAST ARTIFICIAL VARIABLE IS # 8 AND
  IS THE LAST TO REQUIRE THE USE OF THE
             BIG M METHOD.

PAUSE
```

The instructions below are followed by the display of the tableau as modified at this point. This allows the user to review the modifications performed in the previous steps. Once the user has reviewed the tableau, the [RETURN] key or [SPACE] bar is depressed to allow the program to proceed.

```
 THE TABLEAU WILL BE DISPLAYED, THEN YOU
  WILL BE ASKED IF IT IS IN THE PROPER
      FORM FOR THE INITIAL PIVOT.

PAUSE
```

```
       SAMPLE PROBLEM
     CURRENT LP MODEL: MAXIMIZE COST


                  TYPE1         TYPE2         TYPE3        SURPLS        SURPLS
                  X( 1)         X( 2)         X( 3)        X( 4)         X( 5)
OBJ FUNCTION -3.00000E+01 -2.25000E+01 -2.50000E+01  0.00000E+00  0.00000E+01
CN  NAME  VAR ******************************************************************
 1 PERSON  7  5.00000E+00  8.00000E+00  4.00000E+00 -1.00000E+00  0.00000E+00
 2 EQUIP   8  6.00000E+00  3.00000E+00  6.00000E+00  0.00000E+00 -1.00000E+00
 3 PLANES  6  1.00000E+00  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00



                  SLACK         ARTIF         ARTIF        RHS
                  X( 6)         X( 7)         X( 8)
OBJ FUNCTION  0.00000E+00 -3.00000E+02 -3.00000E+02 =  0.00000E+00
CN  NAME  VAR ******************************************************************
 1 PERSON  7  0.00000E+00  1.00000E+00  0.00000E+00 =  2.00000E+03
 2 EQUIP   8  0.00000E+00  0.00000E+00  1.00000E+00 =  1.20000E+03
 3 PLANES  6  1.00000E+00  0.00000E+00  0.00000E+00 =  4.00000E+02

PAUSE
```

As shown below, the user is asked again whether or not the tableau is ready for the first pivot. The opinion of the user is entered and the corresponding feedback is presented.

```
 IS THE TABLEAU IN THE PROPER FORM FOR
          THE INITIAL PIVOT? [Y]

       YOUR RESPONSE WAS INCORRECT.
THERE IS NO INITIAL BASIC SOLUTION SINCE
THE OBJECTIVE FUNCTION COEFFICIENTS OF
THE ARTIFICAL VARIABLES ARE NOT ZERO.

PAUSE
```

Following the above input and comments, the final modification of the tableau occurs. As shown below, the next step is the display of the initial tableau. Note that the above sequence of steps from the last default option display to this point would not have been performed if the user had selected the algorithm to perform the tableau modification.

```
          SAMPLE PROBLEM
          BASIC SOLUTION # 1

                   TYPE1        TYPE2        TYPE3        SURPLS       SURPLS
                   X( 1)        X( 2)        X( 3)        X( 4)        X( 5)
OBJ FUNCTION -3.27000E+03 -3.27750E+03 -2.97500E+03  3.00000E+02  3.00000E+02
CN  NAME  VAR  ****************************************************************
 1 PERSON  7  5.00000E+00  8.00000E+00  4.00000E+00 -1.00000E+00  0.00000E+00
 2 EQUIP   8  6.00000E+00  3.00000E+00  6.00000E+00  0.00000E+00 -1.00000E+00
 3 PLANES  6  1.00000E+00  1.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00


                   SLACK        ARTIF        ARTIF
                   X( 6)        X( 7)        X( 8)         RHS
OBJ FUNCTION  0.00000E+00  0.00000E+00  0.00000E+00 = -9.60000E+05
CN  NAME  VAR  ****************************************************************
 1 PERSON  7  0.00000E+00  1.00000E+00  0.00000E+00 =  2.00000E+03
 2 EQUIP   8  0.00000E+00  0.00000E+00  1.00000E+00 =  1.20000E+03
 3 PLANES  6  1.00000E+00  0.00000E+00  0.00000E+00 =  4.00000E+02
```

After this tableau display, the user who has elected to identify optimal, unbounded or infeasible solutions will be asked the following four questions. Each question is displayed separately and, as shown, instructional comments accompany the user input. Note that prior to this input, [T] mav be entered allowing the user to reexamine the tableau.

```
       TO REVIEW TABLEAU, ENTER T


  WAS THE PREVIOUS TABLEAU OPTIMAL? [N]

       YOUR RESPONSE WAS   CORRECT
   THE LAST TABLEAU WAS  NOT OPTIMAL

PAUSE
```

```
       TO REVIEW TABLEAU, ENTER T


         IS THE SOLUTION FEASIBLE? [Y]

       YOUR RESPONSE WAS INCORRECT
   THE LAST TABLEAU WAS  INFEASIBLE



THE SOLUTION IS INFEASIBLE SINCE THE
ARTIFICIAL VARIABLE X( 8) IS AT A
POSITIVE LEVEL.

PAUSE
```

```
       TO REVIEW TABLEAU, ENTER T


WAS THE PREVIOUS SOLUTION DEGENERATE?[N]

       YOUR RESPONSE WAS   CORRECT
   THE LAST TABLEAU WAS  NOT DEGENERATE

PAUSE
```

```
        TO REVIEW TABLEAU, ENTER T


   WAS THE PREVIOUS SOLUTION UNBOUNDED
  BASED UPON THE NEXT PIVOT COLUMN (ROW)
BEING THE COLUMN (ROW) WITH THE LARGEST
     NEGATIVE Z(J)-C(J)  (B(J)) VALUE? [Y]

       YOUR RESPONSE WAS INCORRECT
   THE LAST TABLEAU WAS       BOUNDED

   THE CURRENT TABLEAU IS BOUNDED SINCE
  ALL THE A(I,J) VALUES IN COLUMN  2 ARE
          NOT NEGATIVE  OR ZERO.

PAUSE
```

The user may elect to have the basic variable values and objective function value displayed or to continue without this display. As shown, the user elected not to display the values by entering a [3].

```
WOULD YOU LIKE THE BASIC SOLUTION VALUES
               DISPLAYED?

         1. DISPLAY ON SCREEN

         2. DISPLAY ON PRINTER

         3. DO NOT DISPLAY


            WHICH OPTION? [3]
```

The next four displays request the user to enter the selected pivot column and row. Each input is accompanied by appropriate feedback. As shown, column number 3 was selected initially , but did not coincide with the algorithm choice. At that point, the user must select the option representing the pivot column to be used.

```
WHICH COLUMN CONTAINS THE CANDIDATE
      ENTERING VARIABLE?

         COLUMN = [3]
```

```
YOUR SELECTION OF PIVOT COLUMN DOES NOT
     MATCH THAT OF THE ALGORITHM.

  WHICH SELECTION DO YOU WISH TO USE?

   1. YOUR SELECTION   COLUMN =  3

                 OR

   2. ALGORITHM SELECTION  COLUMN =  2


           WHICH OPTION? [2]
```

Based upon the above pivot column selection, the ratios for the column are calculated and displayed. The user then enters the number of the row which is felt to be correct for the pivot element. If the user selection had not matched the algorithm selection, the user would have been allowed to change the selection as was shown above for the column selection.

```
              RATIOS FOR COLUMN   2

              ROW   1 =   2.50000E+02
              ROW   2 =   4.00000E+02
              ROW   3 =   4.00000E+02

       WHICH ROW CONTAINS THE CANDIDATE
               LEAVING VARIABLE?
                   ROW = [1]
```

```
  YOUR PIVOT ROW SELECTION MATCHES THE
            ALGORITHM SELECTION.

PAUSE
```

Once the first pivot has been completed, the resulting tableau is displayed.

```
          SAMPLE PROBLEM
          BASIC SOLUTION # 2

                    TYPE1        TYPE2        TYPE3        SURPLS       SURPLS
                    X( 1)        X( 2)        X( 3)        X( 4)        X( 5)
OBJ FUNCTION -1.22156E+03  0.00000E+00 -1.33625E+03 -1.09687E+02  3.00000E+02
CN  NAME  VAR ##################################################################
 1 PERSON  2  6.25000E-01  1.00000E+00  5.00000E-01 -1.25000E-01  0.00000E+00
 2 EQUIP   8  4.12500E+00  0.00000E+00  4.50000E+00  3.75000E-01 -1.00000E+00
 3 PLANES  6  3.75000E-01  0.00000E+00  5.00000E-01  1.25000E-01  0.00000E+00


                    SLACK        ARTIF        ARTIF
                    X( 6)        X( 7)        X( 8)         RHS
OBJ FUNCTION  0.00000E+00  4.09687E+02  0.00000E+00 = -1.40625E+05
CN  NAME  VAR ##################################################################
 1 PERSON  2  0.00000E+00  1.25000E-01  0.00000E+00 =  2.50000E+02
 2 EQUIP   8  0.00000E+00 -3.75000E-01  1.00000E+00 =  4.50000E+02
 3 PLANES  6  1.00000E+00 -1.25000E-01  0.00000E+00 =  1.50000E+02
```

After the tableau has been reviewed, the same sequence of questions, displays, and feedback are repeated for the last tableau calculated. This sequence is shown below.

```
        TO REVIEW TABLEAU, ENTER T


    WAS THE PREVIOUS TABLEAU OPTIMAL? [N]

        YOUR RESPONSE WAS   CORRECT
   THE LAST TABLEAU WAS  NOT OPTIMAL

PAUSE
```

```
        TO REVIEW TABLEAU, ENTER T


          IS THE SOLUTION FEASIBLE? [Y]

        YOUR RESPONSE WAS INCORRECT
   THE LAST TABLEAU WAS  INFEASIBLE


THE SOLUTION IS INFEASIBLE SINCE THE
ARTIFICIAL VARIABLE X( 8) IS AT A
POSITIVE LEVEL.

PAUSE
```

```
        TO REVIEW TABLEAU, ENTER T


WAS THE PREVIOUS SOLUTION DEGENERATE?[Y]

        YOUR RESPONSE WAS INCORRECT
   THE LAST TABLEAU WAS  NOT DEGENERATE


 THE CURRENT TABLEAU IS NOT DEGENERATE
SINCE ALL BASIC VALUES ARE AT A NON-ZERO
                LEVEL.

PAUSE
```

```
        TO REVIEW TABLEAU, ENTER T


   WAS THE PREVIOUS SOLUTION UNBOUNDED
  BASED UPON THE NEXT PIVOT COLUMN (ROW)
BEING THE COLUMN (ROW) WITH THE LARGEST
    NEGATIVE Z(J)-C(J) (B(J)) VALUE? [N]

        YOUR RESPONSE WAS   CORRECT
   THE LAST TABLEAU WAS      BOUNDED

PAUSE
```

```
WOULD YOU LIKE THE BASIC SOLUTION VALUES
            DISPLAYED?

       1. DISPLAY ON SCREEN

       2. DISPLAY ON PRINTER

       3. DO NOT DISPLAY


            WHICH OPTION? [3]
```

```
  WHICH COLUMN CONTAINS THE CANDIDATE
         ENTERING VARIABLE?

            COLUMN = [3]
```

```
 YOUR PIVOT COLUMN SELECTION MATCHES THE
        ALGORITHM SELECTION.

PAUSE
```

```
RATIOS FOR COLUMN  3

ROW  1 =  5.00000E+02
ROW  2 =  1.00000E+02
ROW  3 =  3.00000E+02

WHICH ROW CONTAINS THE CANDIDATE
LEAVING VARIABLE?
ROW = [2]
```

```
YOUR PIVOT ROW SELECTION MATCHES THE
ALGORITHM SELECTION.

PAUSE
```

The following tableau is the result of the second pivot.

```
        SAMPLE PROBLEM
        BASIC SOLUTION # 3

                    TYPE1        TYPE2        TYPE3        SURPLS       SURPLS
                    X( 1)        X( 2)        X( 3)        X( 4)        X( 5)
OBJ FUNCTION  3.33337E+00  0.00000E+00  0.00000E+00  1.66667E+00  3.05554E+00
CN  NAME  VAR  ###################################################################
 1 PERSON  2  1.66667E-01  1.00000E+00  0.00000E+00 -1.66667E-01  1.11111E-01
 2 EQUIP   3  9.16667E-01  0.00000E+00  1.00000E+00  9.33333E-02 -2.22222E-01
 3 PLANES  6 -8.33333E-02  0.00000E+00  0.00000E+00  8.33333E-02  1.11111E-01


                    SLACK        ARTIF        ARTIF
                    X( 6)        X( 7)        X( 8)        RHS
OBJ FUNCTION  0.00000E+00  2.98333E+02  2.96944E+02 = -7.00000E+03
CN  NAME  VAR  ###################################################################
 1 PERSON  2  0.00000E+00  1.66667E-01 -1.11111E-01 =  2.00000E+02
 2 EQUIP   3  0.00000E+00 -8.33333E-02  2.22222E-01 =  1.00000E+02
 3 PLANES  6  1.00000E+00 -8.33333E-02 -1.11111E-01 =  1.00000E+02
```

The sequence of displays which follows is the same as above with a few exceptions. Since it is found in the next display that the last tableau was optimal, the user is questioned concerning the existence of multiple optimal solutions in addition to the previous questions.

```
        TO REVIEW TABLEAU, ENTER T


    WAS THE PREVIOUS TABLEAU OPTIMAL? [Y]

        YOUR RESPONSE WAS    CORRECT
  THE LAST TABLEAU WAS        OPTIMAL

PAUSE
```

```
        TO REVIEW TABLEAU, ENTER T


  IS THE OPTIMAL SOLUTION ALSO FEASIBLE?
                   [Y]

        YOUR RESPONSE WAS    CORRECT
  THE LAST TABLEAU WAS        FEASIBLE

PAUSE
```

```
        TO REVIEW TABLEAU, ENTER T


WAS THE PREVIOUS SOLUTION DEGENERATE?[N]

        YOUR RESPONSE WAS    CORRECT
  THE LAST TABLEAU WAS  NOT DEGENERATE

PAUSE
```

```
TO REVIEW TABLEAU, ENTER T

ARE THERE MULTIPLE OPTIMAL SOLUTIONS?
[Y]

YOUR RESPONSE WAS INCORRECT
THERE ARE NO MULTIPLE SOLUTIONS.

THIS IS SINCE ALL NON-BASIC VARIABLES
HAVE A VALUE OF OTHER THAN ZERO IN THE
OBJECTIVE FUNCTION ROW. IF A ZERO VALUE
WAS PRESENT FOR A NON-BASIC VARIABLE,
INCREASING THE VALUE OF THIS VARIABLE
WOULD NOT CHANGE THE Z VALUE.

PAUSE
```

Since an optimal solution has been obtained, the pivot element selections are no longer required.

The final tableau display is repeated following the above questions since an optimal solution has been obtained. This also occurs when unbounded or infeasible solutions exist. This is followed by the opportunity to display the basic values and objective function value. Option [1] has been selected for screen output as shown below.

BASIC SOLUTION # 3
FINAL TABLEAU - OPTIMAL

CN	NAME	VAR	TYPE1 X(1)	TYPE2 X(2)	TYPE3 X(3)	SURPLS X(4)	SURPLS X(5)
OBJ FUNCTION			3.33337E+00	0.00000E+00	0.00000E+00	1.66667E+00	3.05554E+00
1	PERSON	2	1.66667E-01	1.00000E+00	0.00000E+00	-1.66667E-01	1.11111E-01
2	EQUIP	3	9.16667E-01	0.00000E+00	1.00000E+00	8.33333E-02	-2.22222E-01
3	PLANES	6	-8.33333E-02	0.00000E+00	0.00000E+00	8.33333E-02	1.11111E-01

CN	NAME	VAR	SLACK X(6)	ARTIF X(7)	ARTIF X(8)		RHS
OBJ FUNCTION			0.00000E+00	2.98333E+02	2.96944E+02	=	-7.00000E+03
1	PERSON	2	0.00000E+00	1.66667E-01	-1.11111E-01	=	2.00000E+02
2	EQUIP	3	0.00000E+00	-8.33333E-02	2.22222E-01	=	1.00000E+02
3	PLANES	6	1.00000E+00	-8.33333E-02	-1.11111E-01	=	1.00000E+02

WOULD YOU LIKE THE BASIC SOLUTION VALUES DISPLAYED?

1. DISPLAY ON SCREEN
2. DISPLAY ON PRINTER
3. DO NOT DISPLAY

WHICH OPTION? [1]

SAMPLE PROBLEM
BASIC SOLUTION # 3

TYPE2 = X(2) = 2.00000E+02
TYPE3 = X(3) = 1.00000E+02
SLACK = X(6) = 1.00000E+02

Z= -7.00000E+03

The user may elect to perform additional pivots at this point. This has been provided to allow the user to recover from improper pivot element selections resulting in an infeasible solution. If this option is selected, the ability to perform dual pivots is automatically provided. As shown, this option was not elected.

```
WOULD YOU LIKE TO PERFORM FURTHER PIVOTS
            ON THIS TABLEAU?  [N]
```

The results of the problem must be saved to disk to allow for sensitivity analysis. Since the analyst would like to further study the above solution, a [Y] has been entered.

```
TO PERFORM SENSITIVITY ANALYSIS ON THIS
MODEL, THE INFORMATION OF THE CURRENT
TABLEAU MUST BE SAVED TO DISK.


DO YOU WISH TO SAVE THIS FILE TO DISK?[Y]
```

The sequence shown below requires the input of a diskname:filename of the disk and file in which the results are to be saved. The subsequent displays prompt the user to place the correct disk in a drive.

```
             SAVE LP MODEL TO DISK


   ENTER THE DISK DRIVE NUMBER AND FILE
   NAME YOU WANT THE CURRENT TABLEAU OF
    SAMPLE PROBLEM       SAVED UNDER.

          ENTER EXACTLY AS FOLLOWS
            DISK DRIVE:FILENAME

               EG.   #4:FILENAM

THE DRIVE:FILENAME MUST BE 10 CHARACTERS
                  OR LESS

 DO NOT USE THE SAME NAME USED WHEN THE
       ORIGINAL MODEL WAS ENTERED.

        DISK:FILENAME = [LP1:SAMCM2]

        ARE CORRECTIONS NEEDED? [N]
```

Note that this prompt is for disk LP2 and not LP1. The users of a one-drive system must remove LP1 and insert LP2 at this time.

```
 INSURE DISK    LP2    IS AVAILABLE.

PAUSE
```

One disk-drive users must reinsert disk LP1.

```
INSURE THE DISK TO CONTAIN THE FILE

LP1:SAMCM2

IS AVAILABLE.

PAUSE
```

The user must insure the following question is answered correctly. If entered as shown below when a file already exists on LP1 with the name SAMCM2, an output error will cause the loss of the solution parameters in memory.

```
HAS THIS DISK:FILENAME COMBINATION BEEN
USED PREVIOUSLY?

(ARE YC.I UPDATING A CURRENTLY EXISTING
FILE?)

(Y/N) [N]
```

The prompt below advises the user that the file of the original model input into Module 1 must be available at this time.

```
INSURE THE DISK CONTAINING THE.

LP1:SAMPLE

MODEL IS AVAILABLE.

PAUSE
```

```
INSURE THE DISK TO CONTAIN THE FILE
                LP1:SAMCM2
              IS AVAILABLE.
PAUSE
```

```
 INSURE DISK    LP1    IS AVAILABLE.
PAUSE
```

If another LP model were available, the user could enter [Y] and would be asked to input the diskname:filename of the model desired. Since no other models are available, [N] was entered.

```
 WOULD YOU LIKE TO STUDY ANOTHER MODEL
    WHICH HAS BEEN SAVED TO DISK? [N]
```

```
 INSURE DISK    LP1    IS AVAILABLE.
PAUSE
```

The last display of this module is shown below. This provides the user with the required operating system commands to return to Module 1. The user is cautioned that the period following STARTUP must be entered or Module 1 will not execute.

TO ENTER THE LP DATABASE MODULE:
TYPE
X
LP1:SYSTEM.STARTUP.

The above sequence has given an outline of the use of Module 2. Not all options were employed in the demonstration; however, those able to perform the above steps should not encounter problems in other methods of application.

The sequence of steps normally used in problem solution and analysis would lead the analyst now to the sensitivity module. Since the method of access and use of Module 4, the sensitivity analysis program, is identical following both Modules 2 and 3, this explanation will be presented after the Module 3 demonstration.

Module 3. The method of accessing Module 3 when Module 1 has terminated was briefly noted earlier. It was shown that to enter Module 2, the user would enter [X] and [LP1:ED]. The commands for entering Module 3 are [X] and [LP2:TAB]. Note that no [RETURN] is required following the [X]. Also note that the command [LP2:TAB] communicates that the file TAB on disk LP2 be executed. This requires that disk LP2 be accessible when entering the above commands. After these commands have been entered, the following header will be displayed. This confirms entry into Module 3.

```
              **********************
              *                    *
              *       LINEAR       *
              *                    *
              *    PROGRAMMING     *
              *                    *
              *      PROBLEM       *
              *                    *
              *       SOLVER       *
              *                    *
              *       MODULE       *
              *                    *
              *                    *
              *      MODULE 3      *
              *                    *
              **********************

PAUSE
```

Once either the [RETURN] key or the [SPACE] bar has been depressed, the user is informed to insert disk LP1. One-drive-system users must carefully read these prompts and insure the required disk is available to avoid output errors and data loss.

```
INSURE DISK     LP1     IS AVAILABLE.

PAUSE
```

The user is presented the diskname:filename of the file currently identified as the model to be studied. Should the user not want to study the file shown, an [N] may be entered and the user may then identify the file desired. As shown, the file currently identified is the one desired so a [Y] was entered.

```
        PROBLEM SOLVER OPTION SELECTION

THE PROBLEM CURRENTLY IDENTIFIED AS THE
        PROBLEM TO BE STUDIED IS:

                LP1:SAMPLE


IS THIS THE PROBLEM YOU DESIRE TO STUDY?
                    [Y]
```

The one-drive system user must now reinsert disk LP2.

```
INSURE DISK     LP2     IS AVAILABLE.

PAUSE
```

The default options are displayed next with the programmer-defined defaults shown on the right. The first option is the only one which is not available in Module 2. The selection of option 1 would change the default to "DUAL" and the module would then convert the primal LP model into its dual model. This transformation would be performed without user interface; however, the user would be required to input a new diskname:filename which the dual problem formulation would be stored under. Although this option was not selected for the demonstration, it may be useful in reducing the number of iterations required to solve a selected LP problem.

For this demonstration, option [3] has been entered to show that the output format for the printer is identical to that used in Module 2 for screen output.

```
                DEFAULT OPTIONS
          ENTER OPTION NUMBER TO CHANGE

1. PROBLEM TO SOLVE                       PRIMAL

2. SOLVE BY DUAL PIVOTS                        N

3. OUTPUT LOCATION                        SCREEN

4. OUTPUT FORMAT                        F FORMAT

5. TABLEAUS TO BE DISPLAYED
      INITIAL                                  Y
      INTERMEDIATE                        N =  1
      FINAL                                    Y

6. NO CHANGES

* SEE DOCUMENTATION FOR EXPLANATION

      WHICH OPTION (ENTER 1-6) ? [3]
```

As shown below, the output default value reflects the previous change. One must insure at this time that the printer is turned on and is in a mode which allows printing. Otherwise, the system will wait indefinitely for the printer to accept information. To insure no confusion exists, the output location refers to the device to which tabular data will be transmitted. This selection has no effect on the location of user prompts and instructions. These will always be displayed on the screen.

The option referring to which tableaus are to be displayed will be demonstrated below. As an initial default, all tableaus are to be displayed; however, since these tableaus were shown in the Module 2 demonstration, only the final tableau will be requested here. To change these defaults, option [5] was entered.

```
              DEFAULT OPTIONS
       ENTER OPTION NUMBER TO CHANGE

1. PROBLEM TO SOLVE                    PRIMAL

2. SOLVE BY DUAL PIVOTS                     N

3. OUTPUT LOCATION                    PRINTER

4. OUTPUT FORMAT                     F FORMAT

5. TABLEAUS TO BE DISPLAYED
      INITIAL                               Y
      INTERMEDIATE                      N = 1
      FINAL                                 Y

6. NO CHANGES

* SEE DOCUMENTATION FOR EXPLANATION

      WHICH OPTION (ENTER 1-6) ? [5]
```

The user is asked the sequentially shown questions below. Since only the final tableau is desired, the responses [N], [N], [Y] were entered. If the user had desired to see a selected number of the intermediate tableaus, a [Y] would have been entered for the second response. The user would then be asked to enter a value for the length of cycle between intermediate tableau output. If a [2] were entered, the second, fourth, sixth, etc. intermediate tableaus would be displayed on the selected device.

```
        PROBLEM SOLVER OPTION SELECTION
WHICH TABLEAUS WOULD YOU LIKE DISPLAYED?
        INITIAL TABLEAU? (Y/N) [N]
        INTERMEDIATE TABLEAUS? (Y/N) [N]
        FINAL TABLEAU? (Y/N) [Y]
```

As shown below, the option 5 default value reflects the changes to this point. Once the user has made all desired changes, option [6] is entered to continue.

```
                DEFAULT OPTIONS
         ENTER OPTION NUMBER TO CHANGE

1.   PROBLEM TO SOLVE PRIMAL

2.   SOLVE BY DUAL PIVOTS N

3.   OUTPUT LOCATION PRINTER

4.   OUTPUT FORMAT F FORMAT

5.   TABLEAUS TO BE DISPLAYED
       INITIAL N
       INTERMEDIATE N = 0
       FINAL Y

6.   NO CHANGES

* SEE DOCUMENTATION FOR EXPLANATION

       WHICH OPTION (ENTER 1-6) ?  [6] --
```

The user is next prompted to insert the disk which contains the original model. Following that action, the user is informed that disk LP2 must be available.

```
        INSURE THE DISK CONTAINING THE

                 LP1:SAMPLE

             MODEL IS AVAILABLE.

PAUSE
```

```
 INSURE DISK    LP2    IS AVAILABLE.

PAUSE
```

After the [RETURN] key or [SPACE] bar is depressed in the above prompt, the program begins the formulation and iterative solution process. Since the initial and intermediate tableaus were not requested, the next display is the final tableau. This tableau will be displayed on the printer in the format identical to that of Module 2. The tableau is shown below.

SAMPLE PROBLEM
BASIC SOLUTION # 3
FINAL TABLEAU - OPTIMAL

			TYPE1	TYPE2	TYPE3	SURPLS	SURPLS
			X(1)	X(2)	X(3)	X(4)	X(5)
OBJ FUNCTION			3.33337	.00000	.00000	1.66667	3.05554
CN	NAME	VAR					
1	PERSON	2	.16667	1.00000	.00000	-.16667	.11111
2	EQUIP	3	.91667	.00000	1.00000	.08333	-.22222
3	PLANES	6	-.08333	.00000	.00000	.08333	.11111

			SLACK	ARTIF	ARTIF		
			X(6)	X(7)	X(8)		RHS
OBJ FUNCTION			.00000	293.33300	296.94400	=	-7000.00000
CN	NAME	VAR					
1	PERSON	2	.00000	.16667	-.11111	=	200.00000
2	EQUIP	3	.00000	-.08333	.22222	=	100.00000
3	PLANES	6	1.00000	-.08333	-.11111	=	100.00000

The user is asked whether or not the basic variable values and objective function value are to be displayed. Again, to show the printer output format, a [2] has been entered followed by the output received.

```
WOULD YOU LIKE THE BASIC SOLUTION VALUES
            DISPLAYED?

        1. DISPLAY ON SCREEN

        2. DISPLAY ON PRINTER

        3. DO NOT DISPLAY

          WHICH OPTION? [2]
```

```
          SAMPLE PROBLEM
          BASIC SOLUTION # 3

   TYPE2  = X( 2) =    200.00000
   TYPE3  = X( 3) =    100.00000
   SLACK  = X( 6) =    100.00000

              Z=  -7000.00000
```

The user must next respond if sensitivity analysis will be performed on the solution. If so, a [Y] is entered followed by a request for a diskname:filename to which the solution parameters of Module 3 will be saved. As shown below, [LP1:SAMCM3] has been entered with [N] entered to show no corrections are needed on the filename. The user must be careful not to use a previously used diskname:filename of a file which is still required. If a previously used name has been entered (for example LP1:SAMCM2 from Module 2), this would cause the previous file to be destroyed.

```
TO PERFORM SENSITIVITY ANALYSIS ON THIS
MODEL, THE INFORMATION OF THE CURRENT
TABLEAU MUST BE SAVED TO DISK.


DO Y   WISH TO SAVE THIS FILE TO DISK?
                [Y]
```

```
          SAVE LP MODEL TO DISK


  ENTER THE DISK DRIVE NUMBER AND FILE
  NAME YOU WANT THE CURRENT TABLEAU OF
   SAMPLE PROBLEM       SAVED UNDER.

        ENTER EXACTLY AS FOLLOWS
          DISK DRIVE:FILENAME

            EG.   #4:FILENAM

THE DRIVE:FILENAME MUST BE 10 CHARACTERS
               OR LESS.

 DO NOT USE THE SAME NAME USED WHEN THE
      ORIGINAL MODEL WAS ENTERED.

      DISK:FILENAME = [LP1:SAMCM3]

      ARE CORRECTIONS NEEDED? [N]
```

The following two messages reference disk availability and should be carefully read, especially for the one-drive-system users.

```
 INSURE DISK    LP2:    IS AVAILABLE.

PAUSE
```

```
INSURE THE DISK TO CONTAIN THE FILE

LP1:SAMCM3

IS AVAILABLE

PAUSE
```

An [N] has been entered below signifying that the diskname:filename combination has not been used previously. If one wishes to overwrite an old file, one may enter the previously used diskname:filename above and a [Y] below to accomplish this.

```
HAS THIS DISK:FILENAME COMBINATION BEEN
USED PREVIOUSLY?

(ARE YOU UPDATING A CURRENTLY EXISTING
FILE?)


(Y/N) [N]
```

The user is again prompted to insure the availability of specific disks and files.

```
INSURE THE DISK CONTAINING THE

LP1:SAMPLE

MODEL IS AVAILABLE.

PAUSE
```

```
INSURE THE DISK TO CONTAIN THE FILE FOR

              LP1:SAMCM3

            IS AVAILABLE.

PAUSE
```

```
 INSURE DISK    LP2    IS AVAILABLE.

PAUSE
```

The user may specify that another model be solved at this time by entering [Y] below. This would then be followed by a diskname:filename input of the desired model. This allows the user to enter several models with Module 1 and then transition to Module 3 and solve all the models without repeated moves between modules. This is the recommended procedure for a multiple problem solving session.

Since another model does not currently exist, [N] has been entered followed by a prompt for disk LP2.

```
 WOULD YOU LIKE TO STUDY ANOTHER MODEL
    WHICH HAS BEEN SAVED TO DISK? [N]
```

```
 INSURE DISK    LP2    IS AVAILABLE.

PAUSE
```

The last inputs required in this module are those commands which cause the transition to Module 1. The commands [X] and [LP1:SYSTEM.STARTUP.] are entered with control being returned to Module 1. From that point, instructions on the commands to enter any module may be requested.

TO ENTER THE LP DATABASE MODULE: TYPE X LP1:SYSTEM.STARTUP.

The next section will discuss the sensitivity analysis module, Module 4, and its method of access and use.

Module 4. This module can only be used after a data base has been established using Modules 2 or 3. Upon completion of those modules, you will be directed to type [X] (execute) followed by [LP2:SEN]. When this has been done the following page will appear.

```
PLEASE SELECT ONE ITEM BY NUMBER

   1) RANGE LIMITS------RIGHT-HAND-SIDE
      AND ASSOCIATED Z VALUES

   2) RANGE LIMITS------A(I,J) & C(J)

   3) CHECK OPTIMALITY FOR MULTIPLE
      A(I,J), B(I), OR C(J) CHANGES

   4) ADD A VARIABLE OR A CONSTRAINT

   5) EXIT PROGRAM

[1]
```

Four different sensitivity analysis options are available. The first selection does right-hand-side ranging and determines the associated value of z. The second option does constraint coefficient and objective function coefficient ranging. The third selection allows multiple changes to the original problem and finds a new optimal solution if desired. The fourth option finds a new optimal solution after a new constraint or variable has been added.

```
    DO YOU WANT THE OUTPUT TO GO TO:

          S)CREEN

          P)RINTER

              OR

          B)OTH

    SELECT S, P, OR B
[S]
```

Output is available on the screen or the printer or both simultaneously if desired. The letter preceeding the choice must be entéred.

```
     ENSURE DISK LP2: IS AVAILABLE
PAUSE
```

Disk LP2: contains a file which holds the name of the current data file. This disk must be available or an execution error will occur. This is fatal.

```
THE CURRENT DATA FILE IS
        LP2:SAMPLE

DO YOU WISH TO USE THIS TABLEAU
[Y]
```

The file name read from LP2: is shown. If a different file name is desired, enter N. The program will then request the new file name. The new file name must then be entered.

```
ENSURE THE DISK CONTAINING
      LP2:SAMPLE

      IS AVAILABLE
PAUSE
```

The disk cntaining the data file must be present to avoid a fatal execution error.

```
            ENSURE LP2: IS AVAILABLE

PAUSE
```

The program disk, LP2, must be returned if it had been removed.

```
****************************************

      RIGHT HAND SIDE RANGE LIMITS
            CONSTRAINT #  1

ORIGINAL RIGHT HAND SIDE =   2000.00000
             LOWER BOUND =    800.00000
             UPPER BOUND =   3200.00000
****************************************

AT THE LOWER BOUND--AT THE UPPER BOUND
X(2) =        .00000 X(2) =    400.00000
X(3) =     200.00000 X(3) =       .00000
X(8) =     200.00000 X(8) =       .00001
Z    =   -5000.00000 Z    =  -9000.00000

PAUSE
```

Output from selection 1 is shown. A similar amount of data is presented for each constraint.

```
****************************************
COEFFICIENT      LOWERLIMIT      UPPERLIMIT
 A(1,1)    =      NO LIMIT         7.00000
 A(1,2)    =       5.00000        NO LIMIT
 A(1,3)    =       1.33333         8.88884
 A(2,1)    =      NO LIMIT         7.09093
 A(2,2)    =      -6.00000         4.80000
 A(2,3)    =       5.05881        NO LIMIT
 A(3,1)    =      NO LIMIT        NO LIMIT
 A(3,2)    =      NO LIMIT         1.50000
 A(3,3)    =      NO LIMIT         2.00000

PAUSE
```

If option 2 had been selected, the same data input routine would have been encountered. The constraint coefficient ranging is shown.

```
****************************************
COEFFICIENT      LOWERLIMIT      UPPERLIMIT

    C(1)   =      26.65660        NO LIMIT
    C(2)   =      12.50000        42.50020
    C(3)   =      11.25010        28.63640

PAUSE
```

The objective function coefficient ranging is presented on a separate page.

```
   THIS PROGRAM ACCEPTS MULTIPLE CHANGES
   TO A FINAL TABLEAU AND CHECKS WHETHER
   OR NOT THE CURRENT SOLUTION IS OPTIMAL
   FOR THE NEW PARAMETERS

PAUSE
```

If option 3 had been selected, the caption shown above would appear following the data input routine.

```
   SELECT THE PARAMETERS TO BE CHANGED

       1) C(J)

       2) A(I,J)

       3) B(I)

       4) CHANGES COMPLETE

       5) RETURN TO MAIN MENU

[2]
```

Option 3 allows changes to any or all coefficients of the original problem. After each type of change (1, 2, or 3 above) the menu is presented. The choice shown is [2] (changes to the constraint coefficients).

```
PLEASE ENTER THE ROW TO BE CHANGED

PRESS D)ONE IF COMPLETE

[2]
```

The user must first enter the row to be changed.

```
PLEASE ENTER THE COLUMN TO BE CHANGED

[2]
```

The column to be changed is entered next.

```
THE ORIGINAL VALUE OF A(2,2) WAS
3.000

ENTER NEW VALUE (10 CHARACTERS MAX)

[1.5]
```

The original value is shown, the new value is entered.

```
PLEASE ENTER THE ROW TO BE CHANGED

PRESS D)ONE IF COMPLETE

[D]
```

When all changes to the constraint coefficients have been completed, a [D] is entered. If you desire to make more changes to these coefficients after other changes have been entered, it is permissible and has no ill effect on the outcome.

```
SELECT THE PARAMETERS TO BE CHANGED

    1) C(J)

    2) A(I,J)

    3) B(I)

    4) CHANGES COMPLETE

    5) RETURN TO MAIN MENU

[4]
```

You may select 1, 2, or 3 as many times as desired, including changes to coefficients which have already been changed. On the second change, the original value will be shown. When all desired changes have been entered, select number [4].

```
********STILL OPTIMAL********
PAUSE
```

The program determines whether or not the changes will cause a basis change.

```
     DO YOU WISH TO SOLVE THIS TABLEAU

          SELECT 'Y' OR 'N'
[Y]
```

If you do not wish to see the new tableau, you may return to the main menu by typing [N].

```
          FINAL TABLEAU - OPT1MAL
PAUSE
```

This banner announces that a final solution is available and that it is optimal. Other conditions (degenerate) would be shown if they existed.

```
    DO YOU WANT THE OUTPUT IN

          1)  E FORMAT

                 OR

          2)  F FORMAT

[2]
```

Output for the tableaus is available in either E or F format. Enter the number of your choice.

```
                    X( 1)      X( 2)      X( 3)      X( 4)      X( 5)
OBJ FUNCTION      2.67862     .00000     .00000    2.32143    2.61904
CN  NAME  VAR  ##################################################################
 1         2      .14286    1.00000     .00000    -.14286     .09524
 2         3      .96429     .00000    1.00000     .03571    -.19048
 3         8     -.10714     .00000     .00000     .10714     .09524

PAUSE
```

The final tableau is presented. The output is in 80 column format. To see the right 40 columns, type [CONTROL-A].

```
                     X( 6)         X( 7)         X( 8)          RHS
OBJ FUNCTION     297.67900     297.38100        .00000 =  -7785.71000
CN  NAME  VAR   #####################################################################
 1         2        .14286       -.09524        .00000 =    171.42900
 2         3       -.03571        .19048        .00000 =    157.14300
 3         8       -.10714       -.09524       1.00000 =     71.42860

PAUSE
```

The 80 column format allows 5 variables to be shown at one time (both sides), The display is continued until all data has been shown.

```
              X( 2) =     171.42900
              X( 3) =     157.14300
              X( 8) =      71.42860

                  Z =  -7785.71000

PAUSE
```

The final solution is presented separately. Following this display, the program returns to the main menu, just as it did after the results of options 1 and 2 were shown.

```
   THIS SEGMENT ALLOWS YOU TO ADD AN
   ADDITIONAL CONSTRAINT OR VARIABLE TO
   AN ALREADY SOLVED LINEAR PROGRAMMING
   PROBLEM

PAUSE
```

This caption is shown after data retrieval when option [4] was selected.

```
          DO YOU WISH TO ADD A:

              C)ONSTRAINT
                  OR
              V)ARIABLE

           SELECT 'C' OR 'V'

[V]
```

You can enter either one new constraint or one new variable. Select the letter of your choice.

```
    PLEASE ENTER THE COEFFICIENT FOR
        THE OBJECTIVE FUNCTION


          C(4) = [35]
```

If a problem originally had three variables, the new variable would be shown as number 4. The variables added during the previous solution are moved to the right.

```
PLEASE ENTER THE COEFFICIENT FOR
EACH CONSTRAINT


  A( 1, 4) = [7]
  A( 2, 4) = [5]
  A( 3, 4) = [1]
```

The constraint coefficients for the new variable are entered next.

```
FINAL TABLEAU - OPTIMAL
MULTIPLE OPTIMAL SOLUTIONS EXIST


PAUSE
```

If the user requests a full solution (as shown in option 3) the final conditions will be displayed. The full final tableau will be displayed after this statement as it was in option three.

TO ENTER THE LP DATABASE MODULE: TYPE X LP1:SYSTEM.STARTUP. PAUSE

If option [5] on the main menu is chosen, this instruction is presented. By typing [LP1:SYSTEM.STARTUP.] after the [PAUSE] and [X] (for execution), the program will return to the master menu in Module 1.

APPENDIX B

PROGRAMMERS' GUIDE

CONTENTS

I Introduction

The objective of the Programmers' Guide presented in this Appendix is to provide general information and guidance to those programmers and analysts who wish to modify and/or expand the linear programming package developed in this thesis. Information will be presented which will aid in the location of specific code, the interaction of this code with other units of code, and the specific purpose of each block of code. A section of this guide discusses the user-created disk files and the purpose of each file. Another section explains those procedures which are known to be peculiar to the Apple FORTRAN utilized in the LP package implementation. This section will be of specific interest to those who wish to translate all or a portion of this code for use on another computer, either micro or mainframe. The last three sections are devoted to the program code structure, variables, and the text listings as implemented in this thesis.

II Microcomputer Dependent Features

This linear programming package, which consists of four distinct main programs, has been written in Apple FORTRAN as supported by the Apple II and Apple II-plus microcomputers. This FORTRAN version uses the Apple Pascal Operating System which incorporates UCSD Pascal (Ref 1). Although the Apple FORTRAN language was created with the American National Standards Institute (ANSI) FORTRAN 77 subset as its primary reference, certain limitations and extensions do exist. The purpose of this section is to note those areas which do not conform to the ANSI 77 subset of FORTRAN. Those areas which are noted should be carefully examined prior to translation of these programs for implementation on other computer systems. Only the areas not conforming to the ANSI 77 subset need to be examined when translating to other systems which fully support the ANSI 77 language subset.

The first section discusses the areas in hich the Apple FORTRAN language does not conform to the ANSI 77 subset. Although the ANSI 77 subset specifies that integer and real data types will require the same amount of memory, Apple FORTRAN does not. Integers require two bytes while reals require four bytes (Ref 1:220). This specification places restrictions on the numerical range of both data types and may be different from that of a system which conforms tc the ANSI 77 subset.

Apple FORTRAN also supports some features which are included in the full FORTRAN language but not in the ANSI 77 subset. Several of these features are used in this program.

Subscript expressions: Apple FORTRAN and the full FORTRAN language support array elements as subscript expressions while the ANSI 77 subset does not (Ref 1:220). For example, if X(1)=3, Apple FORTRAN allows the (3,2) element of a Y matrix to be represented as Y[X(1),2]. Conformance to this standard may be accomplished by assigning to a temporary variable the value of the array element and then using this temporary variable as the subscript expression. For the above example, one must state Z=X(1) and then denote the (3,2) element as Y[Z,2]. This requires the designation of another integer variable, and therefore, more memory will be required.

Limits of a DO statement: Apple FORTRAN, as does the full language, places no restrictions on the integer expressions representing the limits of a DO statement, while the ANSI subset is somewhat restrictive (Ref 1:220). Violation of the ANSI 1977 FORTRAN subset standard may be avoided by designating a temporary variable to represent the limit expression at the cost of memory. An example which Apple FORTRAN allows but the subset does not would be:

```
DO 300 i=1,(A+B)
```

The equivalent statement for the subset would replace the expression (A+B) by a single variable.

Expressions in the input/output list of a WRITE statement: Again, the subset is the most restrictive and does not allow expressions as elements of a WRITE statement. Apple FORTRAN does support expressions in the I/O list, but the expression must not begin with a left parenthesis (Ref 1:221). This inconvenience may be overcome by using a leading addition operator symbol. This peculiarity may be removed in translation to another system through the use of a temporary variable. This may require a larger memory space. An example of this would be to replace the expression (A+B) in the input/output list with a single variable.

File structures: The file structure of Apple FORTRAN extends beyond the subset. The ANSI subset allows only unformatted, direct access files and formatted, sequential files. Apple FORTRAN supports both formatted and unformatted in either direct access or sequential files (Ref 1:221). Due to this difference, the OPEN and CLOSE statements referring to these files may not conform to the subset language. All data files of this software package are unformatted sequential files and, therefore, do not conform to the ANSI 77 subset. Consequently, the OPEN and CLOSE statements of these files do not conform to the ANSI subset. To translate these programs to a system whose FORTRAN conforms to the ANSI 77 subset, one could add format specifiers for each of the input/output elements of the READ and WRITE statements and change the OPEN and CLOSE statements accordingly. Specifically, these changes would be required in files which use units 3, 4, or 7 as the input/output units.

CHAR intrinsic function: Apple FORTRAN conforms to the full language but not to the ANSI 77 subset (Ref 1:220-221). The FORTRAN 77 subset does not specify a collating sequence for all possible characters but does specify general guidelines for such a sequence (Ref 3:193). This allows differences to be present among implementations. Apple FORTRAN uses the ASCII (American Standard Code for Information Interchange) in its CHAR intrinsic function implementation. If the new system does not use the ASCII collating sequence, appropriate changes must be made to the present programs prior to translation.

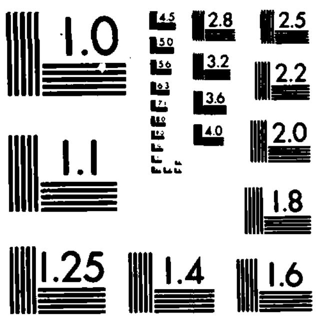
The following features are supported by the Apple FORTRAN language but are not in the subset or the full FORTRAN language (Ref 1:221). Compiler directives, annotated by a "$" in column one, have been used to allow the overlaying of compilation units. Without this feature, each program would have exceeded the memory capabilities of the Apple II-plus microcomputer and prevented the implementation of the software package. This is an area which must be considered very carefully prior to translation attempts. If the new target system is not substantially larger than the Apple II-plus (48K RAM plus a 16K language card) or does not support some type of overlay operation, translation of this software package may not be practical.

The edit control character "$" is a special Apple FORTRAN feature. This character prevents a line feed following a READ or WRITE statement (Ref 1:222). This feature has been used extensively in the WRITE statements which prompt user inputs. This has allowed user input to appear on the same line of the monitor as the prompt and aids greatly in legibility. This feature could be eliminated on translation with careful attention required for the tableau displays.

The previous discussion has noted those areas which are Apple FORTRAN specific. When translating these programs for implementation on another system, one must equally consider the corresponding machine-dependent features of the new target system. These features may coincide with those discussed above and therefore require minimal effort. However, features which are included in the ANSI FORTRAN 77 subset, but are not supported by the new system, must be carefully researched to insure the possibility of a successful translation.

AD-A124 804 FORTRAN BASED LINEAR PROGRAMMING FOR MICROCOMPUTERS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING T R FRALEY ET AL. DEC 82 3/5

UNCLASSIFIED AFIT/GOR/OS/82D-4 F/G 12/1 NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

III Disk File Structure

The object code files, which contain the compiled and linked FORTRAN source code files shown in Part VI of this Appendix, are placed on one of two disks. Each disk also contains a required data file. These two disks have been given the volume names of LP1: and LP2: and contain the following files:

```
LP1:
    SYSTEM.STARTUP              (Module 1)
    ED.CODE                     (Module 2)
    LPDATA                      (Data file)

LP2:
    TAB.CODE                    (Module 3)
    SEN.CODE                    (Module 4)
    LPDATAW                     (Data file)
```

The code files have their corresponding module numbers in parentheses to the right, and the two data files have been annotated for future reference.

The four code files can be placed on a single disk due to their combined size. This factor, combined with the fact that each module is a separate program, required the creation of the two data files named LPDATA and LPDATAW on disks LP1: and LP2:, respectively. Both data files are unformatted, sequential files which contain a character string of maximum length 10. These character strings represent the disk volume number or disk name and filename of a user created data file. These programmer-defined files contain the volume number or disk name and filename which the user has input as the storage location of either the data file of a model entered or the solution of a linear programming problem.

LPDATA contains the user defined data volume or diskname:filename created by Module 1 when the user either saved a LP model to disk or edited a model currently on disk. The file also is used as an information carrier (transfer file) to either Module 2 or 3, whichever is selected when leaving Module 1. When the user attempts to transfer from Module 1 (data base entry) to either of the problem solver modules (Modules 2 and 3), the user is prompted to input the name of the data file which will be studied in the problem solver module selected. The user inputs a volume or diskname:filename and this is written to LP1:LPDATA. When the user begins either of the problem solver modules, LP1:LPDATA is read. The program then directs the problem solver module to read the designated file contained in LP1:LPDATA.

The same logic is also present in the LP2:LPDATAW. This file contains the user-defined volume number or disk name:filename of the data file created by either Modules 2 or 3. The volume number or diskname:filename contained in LP2:LPDATAW is the file which contains the results of either problem solver module. When the user begins Module 4, LP2:LPDATAW is read to direct the sensitivity analysis module to read the designated file. LP2:LPDATAW may also be changed when transferring directly from Module 1 to Module 4.

The two files discussed above serve to link the various modules together by identifying the location and name of the needed data files. When a problem has been entered using Module 1, the user is prompted to save this data under a user-specified disk volume or disk name and filename. The same sequence also occurs upon completion of the problem solver modules. These files, whose volume number or diskname:filenames are placed in LPDATA or LPDATAW, may be saved to disk LP1:, LP2:, or any disk which the user designates in the volume name.

The data files which store the LP models and solutions of the model are unformatted, sequential files. Those files created by Module 1 contain the LP model and are configured in a manner so that Modules 2 and 3 may interpret them. Those data files created by Modules 2 and 3 contain the final results of an LP problem. These files are configured such that Module 4 is capable of interpreting them.

The disk files which have been provided on the two disks must remain as shown. If either LP1:LPDATA or LP2:LPDATAW is removed, an execution error will occur since the programs will attempt to open those files on their respective disks. Any changing of files must be done in conjunction with corresponding code changes to prevent execution errors.

IV <u>LP Package Structure</u>

The four main programs, designated as Modules 1 through 4, which form this LP package have each been compiled in separate units called compilation units. After the compilation of each unit in a module, the units were linked into four distinct object code files and stored on disk under a volume:filename. This process of separate compilation permits the use of the OVERLAY procedure which allows a compilation unit to be resident in memory only while in use. When the overlayed unit is no longer required for processing, resident memory is available for use by other overlay units. This OVERLAY procedure allows each program to be much larger than would have been possible if the entire object code of a module were stored in resident memory.

Listed below are the module numbers followed by the volume or diskname:filename where each module is stored on the two disks provided with this LP package. Next, for each module, the compilation unit names are shown. Below each of the compilation unit names are those text files which are present in each compilation unit. The order of listing of the unit and text file names are the same as shown in PART VI. This will aid the programmer in locating the desired text files.

Module 1

LP1:SYSTEM.STARTUP

```
UNIT10
  PROGRAM DATAB

UNIT11
  SUBROUTINE DATAS
  SUBROUTINE DATAN

UNIT12
  SUBROUTINE EDIT
  SUBROUTINE VNCH

UNIT13
  SUBROUTINE ADVAR
  SUBROUTINE OBJCH

UNIT14
  SUBROUTINE CNVA
  SUBROUTINE DELCON
  SUBROUTINE DELVAR

UNIT15
  SUBROUTINE ICNRCH
  SUBROUTINE ADCON
  SUBROUTINE DISPLY

UNIT16
  SUBROUTINE SAVE
  SUBROUTINE INIT
  SUBROUTINE DATAD
  SUBROUTINE HEADER
  SUBROUTINE MODUL(INEW)
  SUBROUTINE DBHED
  SUBROUTINE INTRO
  SUBROUTINE DBE
  SUBROUTINE DBM
  SUBROUTINE DEM

UNIT17
  SUBROUTINE CHECK(E,INVAL,RNEW)
  SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
  SUBROUTINE CHECK3(E,INVAL,INEW)
```

Module 2

```
LP1:ED

  UNIT20
    PROGRAM EDUC

  UNIT21
    SUBROUTINE OBMDU
    SUBROUTINE OPTION

  UNIT22
    SUBROUTINE READY
    SUBROUTINE CNMDU

  UNIT23
    SUBROUTINE OPT
    SUBROUTINE TCAL

  UNIT24
    SUBROUTINE PIVOT
    SUBROUTINE WORK
    SUBROUTINE OVER(RES)

  UNIT25
    SUBROUTINE HEADER
    SUBROUTINE ASKQ(ASK)
    SUBROUTINE QUESTN
    SUBROUTINE BIGM
    SUBROUTINE INDEX
    SUBROUTINE MODIFA
    SUBROUTINE INTRD

  UNIT26
    SUBROUTINE TDISPL
    SUBROUTINE BASDIS

  UNIT27
    SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
```

Module 3

LP2:TAB

```
UNIT30
  PROGRAM PROBS

UNIT32
  SUBROUTINE OPTN

UNIT33
  SUBROUTINE WORK
  SUBROUTINE OPTB

UNIT34
  SUBROUTINE CONVRT
  SUBROUTINE ACNCH
  SUBROUTINE INRD
  SUBROUTINE NFILE(N)

UNIT35
  SUBROUTINE PSHED
  SUBROUTINE ASKQ(ASK)
  SUBROUTINE BIGM
  SUBROUTINE INDEX
  SUBROUTINE MODIFP
  SUBROUTINE MODIFD

UNIT36
  SUBROUTINE TDISPL

UNIT37
  SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
```

MODULE 4

```
LP2:SEN

  UNIT40
    PROGRAM MAINSA
    SUBROUTINE SELECT

  UNIT41
    SUBROUTINE COMRHS

  UNIT42
    SUBROUTINE COEFFR

  UNIT43
    SUBROUTINE MULCNG

  UNIT44
    SUBROUTINE ADDCON

  UNIT45
    SUBROUTINE SOLVE
    SUBROUTINE OPTB
    SUBROUTINE WORK
    SUBROUTINE TDISPL

  UNIT47
    SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
    SUBROUTINE CHECK(E,INVAL,RNEW)

  UNIT48
    SUBROUTINE RETRIV

  COMVAR.TEXT
```

V Variable List

This section discusses all variables containe^d in this LP software package. It is divided into five subsections. The first subsection discusses those variables which are present in two or more of the four modules and are identically defined. Those variables which are present in two or more, but not all of the modules, have been identified by indicating the modules in which they are used. The next four subsections describe those variables which are specific to just one the four modules. Also listed in the individual module variable listings are those variables which may have different meanings or value ranges in other modules.

A person studying the text files and requiring the meaning of specific variables should first check the respective module variable listing. If the variable is not found there, it will be defined in the main variable listing. Also, if the dimension of an array has been specified by an asterik (*), the dimension of the array may not be the same in each use. This notation is used only when the array elements are assigned by a data statement each time the subroutine is called.

Main Variable Listing

A(20,60)	Real array which contains the models' constraint coefficients, including the surplus, slack, and artifical variable coefficients. (Modules 1, 2, 3, and 4)
ALLOW(*)	Character array with each element a maximum length of 1. The array contains the integer and symbolic characters which are allowed as user inputs. It is used as a reference to validate user input.
AO(20,20)	Real array which contains the original models' constraint coefficients prior to pivot or tableau modification. (Modules 2, 3, and 4)
ARTV(20)	Integer array which contains the constraint numbers of those constraints which contain artificial variables.
ASK	Integer flag which specifies whether or not another model is to be studied before exiting present module. (Modules 2 and 3 only) 0 = Exit module 1 = Remain in present module
B(20)	Real array which contains the original right hand-sides of the constraints. (Modules 1, 2, and 3)
BASIC	Integer variable which contains the current iteration number of basic solutions, both feasible and infeasible. (Modules 2 and 3 only)
BM	Real variable which contains the value used for M in application of the "Big M" Method. (Modules 2 and 3 only)
C(60)	Real array which contains the original objective function coefficients and also the (Z(J)-C(J)) values during subsequent pivots. (Modules 1, 2, and 3)
CB(20)	Integer array which contains the variable subscripts of the basic variables. (Modules 2, 3, and 4)

CN(20)	Character array with each element a maximum length of 6. The array contains the constraint names assigned by user.
D	Integer dummy argument which contains the maximum number of user input characters which will be verified.
E(10)	Character array with each element a maximum length of 1. The array is a dummy argument which is used by subroutines which validate user input.
FMT	Integer flag which denotes whether output is in E or F format. (Modules 2 and 3 only) 0 = E format 1 = F format
FN	Character variable with maximum length 10. It contains the disk name:filename of the file currently being studied.
FNO	Character variable with maximum length 10. It contains the disk name:filename of the file which has been modified by Module 2 or 3, while the new file (name presently in FN) is being created for further study with Module 4. (Modules 2 and 3 only)
GNEG	Real variable which contains the largest negative (Z(J)-C(J)) during the iterative process of determining the pivot column. (Modules 2, 3, and 4)
HOLD	Real variable which contains the tableau element of the pivot column and row currently being modified in the iterative step. (Modules 2, 3, and 4)
HVAL	Integer dummy argument which contains the largest integer value allowed as user input.
IBTAB	Integer variable which denotes the interval between displayed intermediate basic tableaus. (Modules 2 and 3 only) 0 = Do not display intermediate basic tableaus 1 = Display every intermediate basic tableau 2 = Display every second intermediate basic tableau, etc.

IFLAG(1)-(10)	Integer flag. See variable list preceding each module listing for specific meaning in each module.
IFTAB	Integer flag which denotes whether or not final tableau is displayed. (Modules 2 and 3 only) 1 = Display final tableau 2 = Do not display final tableau
INDEXE	Integer variable which specifies the variable subscripts of the artificial variables.
INDEXG	Integer variable which specifies the variable subscripts of the surplus variables.
INDEXL	Integer variable which specifies the variable subscripts of the slack variables.
INEQ(20)	Integer array which contains the type of inequality or equality of each constraint. 0 = Less-than or equal 1 = Greater-than or equal 2 = Equality
INEW	Integer variable which is used as both the actual and dummy arguments of the subroutines which validate user input.
INVAL	Integer flag which is used as both the actual and dummy arguments of the subroutines which validate user input. 0 = User input is valid 1 = User input is invalid
ITAB	Integer flag which denotes whether or not initial basic tableau is to be displayed. 1 = Display initial basic tableau 2 = Do not display initial basic tableau
K	Integer variable which contains the number of constraints in the model.
KFA	Integer variable which contains the column number of the first artificial variable. (Modules 2, 3, and 4)

KFS	Integer variable which contains the column number of the first slack variable. (Modules 2 and 3 only)
KFSA	Integer variable which contains the column number of the first surplus variable. (Modules 2 and 3 only)
KFSU	Integer variable which contains the column number of the last surplus variable. (Modules 2 and 3 only)
MM	Character variable of maximum length 3. It contains either "MAX" or "MIN" for maximization or minimization, respectively.
MXMN	Integer flag which denotes whether original problem was maximization or minimization. 1 = Maximization 2 = Minimization
NEC	Integer variable which contains the number of equality constraints.
NGC	Integer variable which contains the number of greater-than or equal constraints.
NLC	Integer variable which contains the number of less-than or equal constraints.
OBJN	Character variable of maximum length 10. It contains the name of the objective function.
OPTS	Integer flag which denotes whether or not last basic solution was optimal. (Modules 2, 3, and 4) 0 = Non-optimal 1 = Optimal
OUTP	Integer flag which denotes whether output is to be displayed on screen or printer. (Modules 2 and 3 only) 1 = Display on screen 2 = Display on printer
P(10)	Character array with each element a maximum length 1. All user inputs are read as characters. It is also used as actual arguments to subroutine calls which verify user inputs.

PELE	Real variable which contains the coefficient value of the pivot element designated by PK and PR. (Modules 2 and 3 only)
PINEQ(20)	Character array with each element a maximum length 1. It contains the symbolic representation of the equality or inequality for each constraint.
PK	Integer variable which contains the column selected for the current pivot. (Modules 2, 3, and 4)
PN	Character variable of maximum length 20. It contains the problem name supplied by user for current model.
PR	Integer variable which contains the row selected for the current pivot. (Modules 2, 3, and 4)
SPR	Real variable which contains the smallest ratio of the right-hand side/pivot column element for all constraints. (Modules 2, 3 and 4)
SUM	Real variable used as temporary sum of a summation process. (Modules 2 and 3 only)
T	Integer variable which contains the number of 80 column widths required to display tableau.
TIE	Integer flag which that denotes a tie exists for entering or leaving variable. (Modules 2 and 3 only) 0 = No tie 1 = Tie
V	Integer variable which contains the number of variables in the model excluding surplus, slack and artificial variables.
VN(20)	Character array with each element a maximum length of 6. It contains the variable names assigned by user.
VT	Integer variable which contains the total number of variables in the model, including surplus, slack and artificials. (Modules 2, 3, and 4)

XB(20)	Real array which contains the constraint right-hand sides of tableau.
Z	Real variable which contains the current objective function value.

Module 1 Variable Listing

CHAK	Integer variable which contains the column number which the user has selected to make coefficient corrections.
CHAN	Character variable of maximum length 6. It contains the new constraint name which user has defined prior to its assignment to CN(I).
CHARO	Integer variable which contains the number of the constraint which the user has selected to make corrections.
D	Integer dummy argument which contains the maximum number of user input digits.
DECIMA	Integer flag which denotes whether or not a decimal had been found during the process of user input validation. 0 = No decimal processed 1 = Decimal processed
IFLAG(1)	Not used
IFLAG(2)	Integer flag which denotes whether or not model has been saved to disk since entering of data or change of data 0 = Not saved to disk 1 = Saved to disk
IFLAG(3)	Integer flag which denotes whether or not tableau is in proper form for initial pivot. 0 = Not in proper form 1 = Form is correct
IFLAG(4)	Integer flag which denotes form of objective function. 0 = Form is Z=X 1 = Form is Z-X=0
IFLAG(5)	Integer flag which denotes whether or not model contains variable, constraint and objective function names. 0 = No name, subscripts only 1 = Names and subscripts

IFLAG(6) Integer flag which denotes the changes desired in a constraint.

- 0 = Change coefficient, inequality, and right hand side
- 1 = Change coefficient only
- 2 = Change inequality only
- 3 = Change right-hand side only
- 4 = Change constraint name only

IFLAG(7) Not used.

IFLAG(8) Integer flag which denotes the changes in the objective function.

- 0 = Change cost coefficient and maximization/minimization choice
- 1 = Change cost coefficient only
- 2 = Change maximization/minimization choice only

IFLAG(9) Integer flag which denotes whether to display objective function and constraints or objective function only.

- 0 = Display objective function and constraints
- 1 = Display constraints only

IFLAG(10) Not used.

M Real variable which is a multiplier to properly place the decimal in the verified user input.

NEGAT Integer flag which denotes whether user input was a positive or negative value.

- 0 = Positive input
- 1 = Negative input

RNEW Real actual and dummy argument to SUBROUTINE CHECK2(E,INVAL,RNEW)

Module 2 Variable Listing

CO	Character variable of maximum length 7. It contains a character string for display noting a correct response in SUBROUTINE OPT.
CO(20)	Real array which contains the absolute value of the constraint's original right-hand side. It is located in SUBROUTINE ASKQ(ASK).
D(10)	Character array with each element a maximum length of 1. Actual argument in subroutine call statements located in SUBROUTINE OPTION. User input is read into this array.
F	Character variable of maximum length 1. It is located in SUBROUTINE OPT for reading user responses.
IFLAG(1)	Used as a dummy storage area to prevent writing over other IFLAG(*) variables. Also an integer variable which denotes whether a basic solution is infeasible due to a negative right-hand side, and if so, which constraint contains the negative right-hand side. 0 = Solution not infeasible due to negative right-hand side I = Solution infeasible due to negative right-hand side in constraint I
IFLAG(2)	Integer flag which denotes whether or not model has been saved to disk since entering of data or change of data. 0 = Not saved to disk 1 = Saved to disk
IFLAG(3)	Integer flag which denotes that screen is to be cleared after display of tableau. 0 = Do not clear screen 1 = Clear screen
IFLAG(4)	Integer flag which denotes whether or not multiple optimal solutions exist SUBROUTINE OPT. Also used in SUBROUTINE PIVOT as an integer variable which contains the column selection of the algorithm. 0 = No multiple optimal solutions 1 = Multiple optimal solutions exist

IFLAG(5) — Integer flag which denotes whether or not model contains variable, constraint, and objective function names.

- 0 = No names, subscripts only
- 1 = Names and subscripts

IFLAG(6) — Integer variable which contains the basic variable subscript of the degenerate variable in SUBROUTINE OPT. Also used in SUBROUTINE PIVOT as an integer variable which contains the row selection of the algorithm.

- 0 = Solution not degenerate
- else = Basic variable subscript which is zero

IFLAG(7) — Integer flag which denotes whether or not current solution is unbounded or bounded

- 0 = Bounded
- 1 = Unbounded

IFLAG(8) — Integer flag utilized in SUBROUTINE OPT to determine if variable is a basic variable.

- 0 = Non basic variable
- 1 = Basic variable

IFLAG(9) — Integer variable which denotes whether to display current constraints only, current LP model without noting basic variables, or LP model with basic variables annotated.

- 0 = Current LP model without annotating basic variables
- 1 = Current constraints only
- 2 = Current LP model with basic variables annotated

IFLAG(10) — Integer flag which denotes whether or not further pivots are allowed or desired.

- 0 = Further pivots allowable and/or desired
- 5 = Further pivots not desired and/or allowed

INC — Character variable with maximum length of 9. It contains character string for display noting incorrect response in SUBROUTINE OPT.

INEQ(20) — Integer array which contains values designating the type of equality or inequality after constraints with negative right-hand sides have been multiplied by -1.
- 0 = Less-than or equal
- 1 = Greater-than or equal
- 2 = Equality

INF1 — Integer variable which denotes whether or not a basic solution is infeasible due to an artificial variable being negative, and if so, the constraint number of the negative artificial variable.
- 0 = Solution not infeasible due to negative artificial variable
- I = Solution infeasible due to negative artificial variable in constraint I

L — Integer flag which denotes whether or not primal pivots are permissible on the current tableau.
- 0 = Primal pivot not permissible
- 1 = Primal pivot is permissible

M — Integer flag which denotes whether or not dual pivots are permissible on the current tableau.
- 0 = Dual pivot not permissible
- 1 = Dual pivot is permissible

MOD — Integer flag which denotes whether the user or the algorithm should modify the initial tableau into the proper simplex form.
- 1 = User modification
- 2 = Algorithm modification

NEC — Integer variable which contains the number of equality constraints after constraints with negative right-hand sides have been multiplied by -1.

NGC — Integer variable which contains the number of greater-than or equal constraints after constraints with negative right-hand sides have been multiplied by -1.

NLC — Integer variable which contains the number of less-than or equal constraints after constraints with negative right-hand sides have been multiplied by -1.

NNU	Character variable of maximum length 14. It contains a character string for display noting whether solution was not optimal, nondegenerate, or unbounded.
ODB	Character variable of maximum length 10. It contains a character string for display noting whether solution was optimal, degenerate, or bounded.
OIU	Integer flag which denotes whether the user or the algorithm will identify optimal, infeasible, and unbounded solutions. 1 = User 2 = Algorithm
PES	Integer flag which denotes the method of pivot element selection. 1 = User selects, algorithm checks 2 = User selects, no algorithm check 3 = Algorithm selects, no user input
PKS	Integer variable which contains the user selected pivot column.
PRS	Integer variable which contains the user selected pivot row.
RES	Integer flag which denotes whether or not the user wishes to perform a pivot with a pivot element that is equal to or approximately zero.
RATIO	Real variable which contains the ratio of the right-hand side of row I/coefficient of row I, column PK element.
S	Integer variable which contains the proper response value to question asked of user.

Module 3 Variable Listing

C2(20)	Real array which contains the C(J)'s for the dual problem (the X(I)'s of the primal problem).
DUAL	Integer flag which denotes whether or not dual pivots are to be allowed in problem solution. 1 = Dual pivots are not to be used 2 = Dual pivots may be used
IE	Integer counter which contains the number of unconstrained variables added to the dual model.
IFLAG(1)	Used as a dummy storage area to prevent writing over other IFLAG(*) VALUES.
IFLAG(2)	Integer flag which denotes whether or not model has been saved to disk since entering of data or change of data. 0 = Not saved to disk 1 = Saved to disk
IFLAG(3)	Not used.
IFLAG(4)	Integer flag which denotes whether or not multiple optimal solutions exist. 0 = No multiple optimal solutions 1 = Multiple optimal solutions exist
IFLAG(5)	Integer flag which denotes whether or not model contains variable, constraint, and objective function names. 0 = No names, subscripts only 1 = Names and subscripts
IFLAG(6)	Integer flag which denotes whether the solution is degenerate or nondegenerate. 0 = Nondegenerate 1 = Degenerate
IFLAG(7)	Integer flag which denotes whether current solution is unbounded or bounded. 0 = Bounded 1 = Unbounded

IFLAG(8)	Integer flag utilized in SUBROUTINE OPTB to determine if variable a is basic variable. 0 = Non basic variable 1 = Basic variable
IFLAG(9)	Integer flag which denotes whether or not the current solution is to be displayed. 0 = Do not display current solution 1 = Display current solution
IFLAG(10)	Integer flag which denotes whether solution of model was performed by primal pivots or dual pivots. 0 = Primal pivots 1 = Dual pivots
IT	Integer flag which denotes a constraint is required to be added to insure that an initial pivot may be performed. 0 = No constraint added 1 = Constraint added
INFP	Integer flag which denotes whether solution is feasible or infeasible due to either a negative right-hand side or an artificial variable at a positive level. 0 = Feasible 1 = Infeasible
K2	Integer variable which contains the numbe constraints of the dual problem.
N	Integer actual and dummy argument which denotes whether variables or constraints have been added to the model. 1 = Variables 2 = Constraints
NEC	Integer variable which contains the number of equality constraints after constraints with negative right-hand sides have been multiplied by -1.
NGC	Integer variable which contains the number of greater-than or equal constraints after constraints with negative right-hand sides have been multiplied by -1.

NLC	Integer variable which contains the number of less-than or equal constraints after constraints with negative right-hand sides have been multiplied by -1.
PROBT	Integer flag which denotes whether problem to be solved is the primal or dual problem of the current model. 1 = Primal 2 = Dual
TN	Character variable of maximum length 11. It contains a string to be displayed annotating whether constraints or variables have been added to the model.
V2	Integer variable which contains the number of variables in the dual problem.
VN2(20)	Character array with each element a maximum length 6. It contains the variable names of the dual problem.

Module 4 Variable Listing

AF(20,60)	The matrix of real variables which are the final tableau values of the full matrix. These values are read from the datafile and are then modified by some sections of Module 4 in order to obtain a new final tableau.
ARTVAR	An integer variable used to count the number of artificial variables in the problem and to determine which column in B-inverse is associated with the given constraint.
BASIC(20)	An integer variable used to indicate whether or not a particular column in the A matrix is in the basis. 0 = not in basis 1 = in basis
BO(20)	The vector of original values of the right-hand side which were entered in Module 1.
BF(20)	The vector of final values for the right-hand side from Module 3 or, after modification, the new final values.
CO(20)	The vector of original objective function coefficients from Module 1.
CF(20)	The vector of objective function coefficient in the final tableau from Module 3 or, after modification, the new final tableau values.
CKILL1	A real variable used to check whether or not a lower bound will make the sensitivity analysis ill-conditioned.
CKILL2	A real variable used to check whether or not an upper bound will make the sensitivity analysis ill-conditioned.
CLOWER	A real variable used to compute the lower bound on each objective function coefficient.
COL	An integer variable used to denote the columns under consideration.

CONSTR	An integer variable used to denote the constraint under consideration.
CUPPER	A real variable used to compute the upper bound on each objective function coefficient.
DELADN	A real variable used to compute the minimum negative change to each element of the original A matrix which would cause a multiple optimal solution in the final tableau.
DELAUP	A real variable used to compute the minimum positive change to each element of the original A matrix which would cause a multiple optimal solution in the final tableau.
DELTAA(20)	A temporary vector of real variables which holds the values of the new column of an added variable while the original values are being multiplied by B-inverse.
DELTAA(20,20)	A temporary matrix of real variables which holds the change (delta) to each of the elements in the A matrix.
DELTAB(20)	A temporary vector of real variables which holds the change (delta) to each of the elements in the B vector.
DELTAC(20)	A temporary vector of real variables which holds the change (delta) to each of the elements in the C vector.
HEAD1	An integer variable which indicates whether or not a heading has been displayed on the screen. 0 = Heading has not been displayed 1 = Heading has been displayed
HEAD2	An integer variable which indicates whether or not a heading has been printed. 0 = Heading has not been printed 1 = Heading has been printed
IFLAG(4)	An integer variable used to indicate a multiple optimal solution. 0 = Not multiple optimal 1 = Multiple optimal

IFLAG(5)	An integer variable used to indicate whether or not named resources and variables are used. 0 = Names not used 1 = Names used
IFLAG(6)	An integer variable which indicates whether or not the problem is degenerate. 0 = Not degenerate 1 = Degenerate
IFLAG(7)	An integer variable which indicates whether or not the problem is unbounded. 0 = Not unbounded 1 = Unbounded
IFLAG(8)	An integer variable used as a temporary indicator when checking for multiple optimal solutions.
IFLAG(9)	An integer variable used as an indicator to prevent manipulation of possibly ill-conditioned matrices.
IFLAG(10)	An integer variable which shows when dual pivots have been used during the initial tableau solution. 0 = No dual pivots 1 = Dual pivots
ILL1	An integer variable which indicates whether or not the lower bound of an element in the A matrix may present an ill-conditioned problem when solved through sensitivity analysis.
ILL2	An integer variable which indicates whether or not the upper bound of an element in the A matrix may present an ill-conditioned problem when solved through sensitivity analysis.
J	An integer variable generally used in DO loops to denote columns 1 through V or VT.
LWBD(20,20)	A real matrtix used during right-hand-side ranging to compute the lower bound of a column of the A matrix associated with a particular constraint.

LINES	An integer variable used to count the number of lines which have been displayed on the screen.
LINEP	An integer variable used to count the number of lines which have been printed on a page.
LWBD	A real variable used to compute the lower bound of an element in the A matrix.
NEWA(20,20)	A real matrix used to hold the new (user input) values of the A elements.
NEWB(20)	A real vector used to hold the new values of the right-hand side. The first element of the vector (NEWB(1)) is occassionally used as a temporary holding variable during computations.
NEWCJ(20)	A real vector used to hold the new values of the objective function coefficients.
RMAX(20)	A real vector used to compute the minimum positive resource (right-hand-side) change which would force a change in the basis.
RMIN(20)	A real vector used to compute the minimum negative resource (right-hand-side) change which would force a change in the basis.
ROW	An integer variable generally used in DO loops to vary the constraints from 1 to K while working with a different constraint under the variable name CONSTR.
RSCH(20,20)	A real matrix which holds the ratio between the right-hand-side value and the A element of the column associated with the constraint under consideration. The minimum positive and negative ratios determine the maximum right-hand-side changes allowed for the constraint while maintaining the current basis.
RSLLIM(20)	A real vector which indicates the lower limit for each right-hand-side element.
RSUPLIM	A real vector which indicates the upper limit for each element of the right-hand side.

SELINP(10)	A character vector used during keyboard input of numbers. The "characters" are sent to a subroutine to be checked and then returned as numbers if no errors are detected.
SELSUB	A character variable used to direct the desired subroutine call.
SELOUT	A character variable used to direct output to the screen, printer, or both.
SELSOL	A character variable used to indicate whether or not a particular tableau will be solved.
SLACK	An integer variable used to count the number of slack variables.
TEMP	A real variable used to temporarily store values during computation.
TEMPA	A real variable used to temporarily store values during computation.
TEMPA(20,20)	A matrix of real variables which is used to hold the columns of the B-inverse matrix while the columns of the matrix are being realigned to the identity matrix order.
TEMPCJ(20)	A vector of real variables which holds the values of the objective function coefficients which are above the slack and artificial columns while the columns of the B-inverse matrix are being reordered.
UPBD	A real variable used to compute the upperbound of an element in the A matrix.
UPBD(20,20)	A real matrix used during right-hand-side ranging to compute the upper bound of a column of the A matrix associated with a particular constraint.
ZLP	A real variable which holds the Z lower bound value which will be printed.
ZUP	A real variable which holds the Z upper bound value which will be printed.

ZLS — A real variable which holds the Z lower bound value which will be displayed on the screen.

ZUS — A real variable which holds the Z upper bound value which will be displayed on the screen.

VI Program Listings

The following listings are the text files which have been compiled and linked to form the code files of this LP package. Preceding each program and subroutine listing are comments which may assist a programmer in efforts to modify, expand, or translate the Apple FORTRAN source code.

The comment blocks contain several items of importance in a standard format to assist future programmers. The first item listed in each comment block is the module number which the listed program or subroutine is a part. Immediately following is the compilation unit name which contains this program or subroutine. The next line is only present in the comment block of the first listing of each compilation unit. This line lists those compilation units which contain subroutines called by this compilation unit. Next, the name of the program or subroutine which this comment block preceeds is shown. The following section is a brief discussion of the program's or subroutine's purpose and any special items of interest. The program or subroutines which call this subroutine are listed following the discussion. Next, a listing of those subroutines or programs which may call this subroutine are shown. The last section of the comment block identifies those variables which either influence execution of the program/subroutine or are changed during execution. The first variables listed with the heading USED are those which may be utilized, but not changed,during execution. The second heading, MODIFIED, lists those variables which may change in value during the execution of the program or subroutine. Only those variables directly used or modified by the programs or subroutines have been shown. Therefore, if Program A calls Subroutine B which changes the value of variable C, only Subroutine B will list variable C as a modified variable. Also, note that all arrays in which the specific array element or elements used or modified may vary due to problem size are annotated with an asterik (*). Numeric subscripts are shown only where a specific element or elements are known to be either used or modified during the execution of the program or subroutine.

```
C ****************************************
C  MODULE 1 UNIT10                                             *
C    UNIT $USES: UNIT11 THRU UNIT17                            *
C                                                              *
C  PROGRAM DATAB                                               *
C  USE: MAIN PROGRAM OF MODULE 1 LP PACKAGE. PURPOSE OF MODULE IS THE *
C       ENTRY OF NEW AND EDITING OF EXISTING LP MODELS IN A FORM     *
C       ACCEPTABLE WITH MODULES 2 AND 3. MODULE 1 CONSISTS OF 8      *
C       SEPARATELY COMPILED UNITS (UNIT10 THRU UNIT17) WITH ALL UNITS *
C       EXCEPT UNIT10 BEING OVERLAY UNITS.                           *
C       PROGRAM DATAB ACTS AS AN OUTER COMMAND LEVEL WHICH SOLICITS  *
C       USER INPUT DESIGNATINS THE OPTION DESIRED.  THIS DESIGN      *
C       ALLOWS OVERLAY UNITS TO BE RELEASED FROM MEMORY PRIOR TO NEW *
C       UNITS BEING CALLED WHICH WOULD OVERLOAD MEMORY.              *
C  CALLED BY: NONE                                             *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)              *
C             SUBROUTINE DATAD                                 *
C             SUBROUTINE DATAN                                 *
C             SUBROUTINE DATAS                                 *
C             SUBROUTINE DBE                                   *
C             SUBROUTINE DBHED                                 *
C             SUBROUTINE DBM                                   *
C             SUBROUTINE DEM                                   *
C             SUBROUTINE DISPLY                                *
C             SUBROUTINE EDIT                                  *
C             SUBROUTINE GENIF                                 *
C             SUBROUTINE HEADER                                *
C             SUBROUTINE INIT                                  *
C             SUBROUTINE INTRO                                 *
C             SUBROUTINE MODUL(INEW)                           *
C  VARIABLES:                                                  *
C       USED: IFLAG(2),INVAL                                   *
C   MODIFIED: IFLAG(5),IFLAG(9),INEW,P(*)                      *
C ****************************************
$USES UCHECK IN UNIT17.CODE OVERLAY
$USES USAVE IN UNIT16.CODE OVERLAY
$USES UICNRCH IN UNIT15.CODE OVERLAY
$USES UCNVA IN UNIT14.CODE OVERLAY
$USES UADVAR IN UNIT13.CODE OVERLAY
$USES UEDIT IN UNIT12.CODE OVERLAY
$USES UDATAS IN UNIT11.CODE OVERLAY
      PROGRAM DATAB
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),FN,MM,FN,PINEQ(20),P(10),OBJN
      OPEN(1,FILE='CONSOLE:')
      OPEN(5,FILE='CONSOLE:')
      CALL HEADER
100   WRITE(1,'(/,3X,''ARE INTRODUCTORY REMARKS DESIRED?''/13X,''(Y/
     .N, RETURN) '',$)')
```

```
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        CALL INTRO
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,110)
110     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 100
      ENDIF
      INEW=0
C     USER SELECTS DESIRED MODULE
      CALL MODUL(INEW)
      CALL DBMED
C     DATA BASE ENTRY OPTIONS DISPLAYED
120   CALL DBE
130   WRITE(1,'(/13X,''WHICH OPTION?  '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,5,INVAL,INEW)
      IF(INVAL .EQ.1)THEN
        WRITE(1,110)
        GO TO 120
      ENDIF
      GO TO(140,140,140,150,160)INEW
C     ALL VALUES INITIALIZED TO ZERO
140   CALL INIT
      WRITE(1,'(A)')CHAR(12)
      IF(INEW .EQ. 1)THEN
C       USER HAS SELECTED TO ENTER MODEL WITH SUBSCRIPTS
        IFLAG(5)=0
        CALL GENIF
        CALL DATAS
        GO TO 200
      ELSEIF(INEW .EQ. 2)THEN
C       USER HAS SELECTED TO ENTER MODEL WITH NAMES
        IFLAG(5)=1
        CALL GENIF
        CALL DATAN
        GO TO 200
      ELSE
C       USER HAS SELECTED TO READ MODEL FROM DISK
        CALL DATAD
        GO TO 200
      ENDIF
150   CALL INTRO
C     USER HAS SELECTED TO REVIEW INTRODUCTORY REMARKS
      GO TO 120
160   STOP
C     DATA BASE MANAGEMENT OPTIONS DISPLAYED
200   CALL DBM
210   WRITE(1,'(/13X,''WHICH OPTION?  '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,5,INVAL,INEW)
      IF(INVAL .EQ.1)THEN
```

```
        WRITE(1,110)
        GO TO 210
      ENDIF
      WRITE(1,'(A)')CHAR(12)
      GO TO(220,230,240,250,250,270)INEW
220   IFLAG(9)=0
C     INPUT MODEL IS DISPLAYED
      CALL DISPLY
      GO TO 200
C     CONTROL PASSED TO EDITING SUBROUTINE
230   CALL EDIT
      GO TO 200
C     INPUT MODEL IS SAVED TO DISK
240   CALL SAVE
      GO TO 200
C     CHECKS TO INSURE MODEL SAVED TO DISK PRIOR TO TERMINATION
250   IF(IFLAG(2) .EQ. 0)THEN
        WRITE(1,260)
260     FORMAT(10(/),15X,'WARNING!!' /6X,'CURRENT FILE WILL BE LOST!'//
     .13X,'CONTINUE? (Y/N) ',$)
        READ(5,'(A)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          IF(INEW .EQ. 4)THEN
C           USER HAS CHOSEN TO RETURN TO MANAGEMENT MENU
            GO TO 120
          ELSE
            GO TO 280
          ENDIF
        ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
          GO TO 200
        ELSE
          WRITE(1,110)
          GO TO 250
        ENDIF
      ENDIF
      IF(INEW .EQ. 4)THEN
        GO TO 120
      ELSE
        GO TO 280
      ENDIF
270   WRITE(1,260)
      READ(5,'(A)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        STOP
      ELSE
        GO TO 200
      ENDIF
C     EXECUTION MANAGEMENT OPTIONS DISPLAYED
200   CALL DEM
290   WRITE(1,'(/13X,''WHICH OPTION?  '',$)')
      READ(5,'(A)')P(1)
      CALL CHECK2(P,1.5,INVAL,INEW)
```

```
      IF(INVAL .EQ. 1)THEN
        WRITE(1,110)
        GO TO 290
      ENDIF
      WRITE(1,'(A)')CHAR(12)
      GOTO(300,300,300,200,160)INEW
C     USER SELECTS NEXT MODULE DESIRED
300   CALL MODUL(INEW)
      STOP
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT11                                                             *
C    UNIT $USES: UNIT12 THRU UNIT17                                            *
C                                                                              *
C  SUBROUTINE DATAS                                                            *
C  USE: SOLICITS INPUT OF OBJECTIVE FUNCTION AND CONSTRAINT                    *
C       COEFFICIENTS, CONSTRAINT INEQUALITIES AND RHS'S FOR LP MODELS          *
C       WHICH VARIABLES ARE DESIGNATED BY SUBSCRIPT ONLY.  USED ONLY           *
C       FOR INPUT OF NEW MODELS.                                               *
C  CALLED BY: NONE                                                             *
C  CALLS    : SUBROUTINE CHECK(P,INVAL,RNEW)                                   *
C             SUBROUTINE CHECK3(P,INVAL,INEW)                                  *
C             SUBROUTINE ICNRCH                                                *
C             SUBROUTINE OBJCH                                                 *
C  VARIABLES:                                                                  *
C       USED: INEW,INVAL,K,MM,PN,RNEW,V                                        *
C   MODIFIED: A(*,*),B(*),C(*),IFLAG(2),IFLAG(3),IFLAG(4),IFLAG(8),            *
C             INEQ(*),NEC,NGC,NLC,P(*),PINEQ(*)                                *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK IN UNIT17.CODE OVERLAY
$USES USAVE IN UNIT16.CODE OVERLAY
$USES UICNRCH IN UNIT15.CODE OVERLAY
$USES UCNVA IN UNIT14.CODE OVERLAY
$USES UADVAR IN UNIT13.CODE OVERLAY
$USES UEDIT IN UNIT12.CODE OVERLAY
      SUBROUTINE DATAS
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      WRITE(1,100)CHAR(12)
100   FORMAT(A)
      WRITE(1,'(/8X,''OBJECTIVE FUNCTION INPUT''///1X,''INPUT THE FUNCTI
     .ON AS IF IT WERE IN THE''/13X,''FOLLOWING FORM''/5X,''Z = X(1) + X
     .(2) + X(3) + ETC.''/)')
      WRITE(1,'(1X,''A MAXIMUM OF 10 ENTRIES PER COEFFICIENT''
     ./1X,''INCLUDING DECIMAL AND SIGN ARE ALLOWED.''//1X,''IF COEFFICIE
     .NT IS ZERO, HIT "RETURN"''/10X,''WITHOUT DIGIT ENTRY.'',6(/))')
      PAUSE
110   WRITE(1,100)CHAR(12)
      WRITE(1,120)PN,MM
120   FORMAT(5X,'PROBLEM ID: ',A20,/8X,'OBJECTIVE FUNCTION INPUT'/14X,A3
     .,'IMIZATION',/)
C     OBJECTIVE COEFFICIENTS INPUT BY USER
      DO 160 J=1,V
130     WRITE(1,140)J
140     FORMAT(8X,'C(',I2,') = ',$)
        READ(5,'(10A1)')(P(I),I=1,10)
        CALL CHECK(P,INVAL,RNEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,150)
```

```
150       FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
          GO TO 130
        ELSE
          C(J)=RNEW
        ENDIF
160   CONTINUE
      WRITE(1,'(///)')
      PAUSE
C     MODEL IS IN Z=X FORM
      IFLAG(4)=0
170   WRITE(1,100)CHAR(12)
      WRITE(1,120)PN,MM
      WRITE(1,180)
180   FORMAT(10(/),7X,'ARE CORRECTIONS NEEDED?  ',$)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        IFLAG(8)=0
C       CONTROL PASSED TO OBJ FUNCTION EDITING ROUTINE
        CALL OBJCH
        GO TO 190
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,150)
        GO TO 170
      ENDIF
190   WRITE(1,100)CHAR(12)
      WRITE(1,'(12X,''CONSTRAINT INPUT''//''INPUT CONSTRAINT VARIABLE CO
     .EFFICIENTS''/''AS IF THE CONSTRAINT WAS IN THE''/''FOLLOWING FORM
     .''/6X,''X(1) + X(2) + X(3)  <=> RHS''/)')
      WRITE(1,'(''THE VARIABLE COEFFICIENTS ARE A MAXIMUM''
     ./''OF 10 CHARACTERS.''//''IF COEFFICIENT IS ZERO, ENTER 0 OR HIT''
     ./''"RETURN" WITHOUT ENTRY.'')')
      WRITE(1,'(/''THE LESS-THAN (<) REPRESENTS A LESS-THAN''/''OR EQUAL
     . INEQUALITY.''//''THE GREATER-THAN (>) REPRESENTS A''/''GREATER-TH
     .AN OR EQUAL INEQUALITY.''//''NEGATIVE RHS IS PERMITTED.'',/)')
      PAUSE
      WRITE(1,100)CHAR(12)
C     CONSTRAINT COEFFICIENTS INPUT BY USER
      DO 270 I=1,K
200     WRITE(1,210)PN,I
210     FORMAT(5X,'PROBLEM ID: ',A20,/13X,'CONSTRAINT ',I2,/)
        DO 240 J=1,V
220       WRITE(1,230)J
230       FORMAT(11X,'X(',I2,') = ',$)
          READ(5,'(10A1)')(P(L),L=1,10)
          CALL CHECK(P,INVAL,RNEW)
          IF(INVAL .EQ. 1)THEN
            WRITE(1,150)
            GO TO 220
          ELSE
            A(I,J)=RNEW
          ENDIF
240     CONTINUE
```

```
C       CONSTRAINT INEQUALITY INPUT BY USER
250     WRITE(1,'(6X,''INEQUALITY    '',$)')
        READ(5,'(A1)')P(1)
        CALL CHECK3(P,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,150)
          GO TO 250
        ELSE
          INEQ(I)=INEW
        ENDIF
C       COUNT OF EACH TYPE INEQUALITY PERFORMED
        IF(INEW .EQ. 0)THEN
          NLC=NLC + 1
          PINEQ(I)='<'
        ELSEIF(INEW .EQ.1)THEN
          NGC=NGC + 1
          PINEQ(I)='>'
        ELSE
          NEC=NEC + 1
          PINEQ(I)='='
        ENDIF
C       CONSTRAINT RHS INPUT BY USER
 260    WRITE(1,'(13X,''RHS = '',$)')
        READ(5,'(10A1)')(P(L),L=1,10)
        CALL CHECK(P,INVAL,RNEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,150)
          GO TO 260
        ELSE
          B(I)=RNEW
        ENDIF
        WRITE(1,100)CHAR(12)
270   CONTINUE
280   WRITE(1,100)CHAR(12)
      WRITE(1,180)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
C       CONTROL PASSED TO CONSTRAINT CHANGE ROUTINE
        CALL ICNPCH
        GO TO 290
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,150)
        GO TO 280
      ENDIF
290   IFLAG(3)=0
      IFLAG(2)=0
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT11                                                     *
C                                                                      *
C  SUBROUTINE DATAN                                                    *
C  USE: SOLICITS INPUT OF OBJECTIVE FUNCTION AND CONSTRAINT            *
C       COEFFICIENTS, CONSTRAINT INEQUALITIES AND RHS'S FOR LP MODEL   *
C       WHICH VARIABLES ARE DESIGNATED BY NAMED VARIABLES,             *
C       CONSTRAINTS, AND OBJECTIVE FUNCTION.  USED ONLY FOR INPUT OF   *
C       NEW MODELS.                                                    *
C  CALLED BY: PROGRAM DATAB                                            *
C  CALLS    : SUBROUTINE CHECK(P,INVAL,RNEW)                           *
C             SUBROUTINE CHECK3(P,INVAL,INEW)                          *
C             SUBROUTINE ICNRCH                                        *
C             SUBROUTINE OBJCH                                         *
C             SUBROUTINE VNCH                                          *
C  VARIABLES:                                                          *
C       USED: INEW,INVAL,K,MM,OBJN,PN,RNEW,V                           *
C   MODIFIED: A(**),B(*),C(*),CN(*),IFLAG(2),IFLAG(3),IFLAG(4),        *
C             IFLAG(6),IFLAG(8),INEQ(*),NEC,NGC,NLC,P(*),PINEQ(*),     *
C             VN(*)                                                    *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE DATAN
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      WRITE(1,100)CHAR(12)
100   FORMAT(A)
      WRITE(1,'(7(/),10X,''VARIABLE NAME INPUT''//1X,''ENTER VARIABLE NA
     .MES WHICH CORRESPOND''/3X,''TO THE '',I2,'' VARIABLES THAT AFFECT
     .''/15X,A10//''NAMES ARE TO BE 6 CHARACTERS OR LESS.'',5(/))') V,
     .OBJN
      PAUSE
      WRITE(1,100)CHAR(12)
      WRITE(1,120)PN
120   FORMAT(5X,'PROBLEM ID: ',A20)
      WRITE(1,130)
130   FORMAT(10X,'VARIABLE NAME INPUT'/)
C     VARIABLE NAMES INPUT BY USER
      DO 140 J=1,V
        WRITE(1,'(13X,''X('',I2,'') = '',$)')J
        READ(5,'(A6)')VN(J)
140   CONTINUE
150   WRITE(1,100)CHAR(12)
      WRITE(1,120)PN
      WRITE(1,130)
      WRITE(1,160)
160   FORMAT(11(/),7X,'ARE CORRECTIONS NEEDED?  ',$)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
C       CONTROL PASSED TO VARIABLE NAME EDITING ROUTINE
```

```
        CALL VNCH
       ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,170)
170     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        PAUSE
        GO TO 130
       ENDIF
180    WRITE(1,100)CHAR(12)
       WRITE(1,'(7(/),9X,''CONSTRAINT NAME INPUT''//''ENTER CONSTRAINT NA
      .MES WHICH CORRESPOND''/''TO THE '',I2,'' CONSTRAINTS WHICH AFFECT'
      .'/A10//''NAMES ARE TO BE 6 CHARACTERS OR LESS.'',4(/))')K,OBJN
       PAUSE
       WRITE(1,100)CHAR(12)
       WRITE(1,120)PN
       WRITE(1,190)
190    FORMAT(9X,'CONSTRAINT NAME INPUT'/)
C      CONSTRAINT NAMES INPUT BY USER
       DO 210 I=1,K
        WRITE(1,'(9X,''CONSTRAINT '',I2,'' = '',$)')I
        READ(5,'(A6)')CN(I)
210    CONTINUE
220    WRITE(1,100)CHAR(12)
       WRITE(1,120)PN
       WRITE(1,190)
       WRITE(1,160)
       READ(5,'(A1)')P(1)
       IF(ICHAR(P(1)) .EQ. 89)THEN
C       FLAG ALLOWS ONLY CONSTRAINT NAME CHANGE TO BE PERFORMED IN
C         EDITING ROUTINE
        IFLAG(6)=4
        CALL ICNHCH
        IFLAG(6)=0
       ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,170)
        PAUSE
        GO TO 220
       ENDIF
230    WRITE(1,100)CHAR(12)
       WRITE(1,'(/8X,''OBJECTIVE FUNCTION INPUT''//1X,''INPUT THE FUNCTIO
      .N AS IF IT WERE IN THE''/13X,''FOLLOWING FORM''/5X,''Z = X(1) + X(
      .2) + X(3) + ETC.''//1X,''A MAXIMUM OF 10 ENTRIES PER COEFFICIENT''
      .)')
       WRITE(1,'(1X,''INCLUDING DECIMAL AND SIGN ARE ALLOWED.''//1X,''IF
      .COEFFICIENT IS ZERO, HIT "RETURN"''/10X,''WITHOUT DIGIT ENTRY.'',
      .4(/))')
       PAUSE
       WRITE(1,100)CHAR(12)
       WRITE(1,240)PN,OBJN,MM
240    FORMAT(5X,'PROBLEM ID: ',A20,/8X,'OBJECTIVE FUNCTION INPUT'/7X,A10
      .,2X,A3,'IMIZATION',/)
       DO 280 J=1,V
C      OBJECTIVE COEFFICIENTS INPUT BY USER
```

```
250     WRITE(1,260)J,VN(J)
260     FORMAT(7X,'C(',I2,') = ',A6,' = ',$)
        READ(5,'(10A1)')(P(I),I=1,10)
        CALL CHECK(P,INVAL,RNEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,170)
          GO TO 250
        ELSE
          C(J)=RNEW
        ENDIF
280   CONTINUE
      PAUSE
C     MODEL IS IN Z=X FORM
      IFLAG(4)=0
290   WRITE(1,100)CHAR(12)
      WRITE(1,240)PN,OBJN,MM
      WRITE(1,160)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        IFLAG(8)=0
C       CONTROL PASSED TO OBJECTIVE FUNCTION EDITING ROUTINE
        CALL OBJCH
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,170)
        PAUSE
        GO TO 290
      ENDIF
300   WRITE(1,100)CHAR(12)
      WRITE(1,'(12X,''CONSTRAINT INPUT''//''INPUT CONSTRAINT VARIABLE CO
     .EFFICIENTS''/''AS IF THE CONSTRAINT WAS IN THE''/''FOLLOWING FORM
     .''/6X,''X(1) + X(2) + X(3)  (=) RHS'')')
      WRITE(1,'(/''THE VARIABLE COEFFICIENTS ARE A MAXIMUM''
     ./''OF 10 CHARACTERS''//''IF COEFFICIENT IS ZERO, ENTER 0 OR HIT''
     ./''"RETURN" WITHOUT ENTRY.'')')
      WRITE(1,'(/''THE LESS-THAN (<) REPRESENTS A LESS-THAN''/''OR EQUAL
     . INEQUALITY.''//''THE GREATER-THAN (>) REPRESENTS A''/''GREATER-TH
     .AN OR EQUAL INEQUALITY.''//''NEGATIVE RHS IS PERMITTED.'',/)')
      PAUSE
      WRITE(1,100)CHAR(12)
C     CONSTRAINT COEFFICIENTS INPUT BY USER
      DO 400 I=1,K
310     WRITE(1,120)PN
320     WRITE(1,'(9X,''CONSTRAINT '',I2,'' = '',A6,/)')I,CN(I)
        DO 350 J=1,V
330       WRITE(1,340)J,VN(J)
340       FORMAT(7X,'X(',I2,') = ',A6,' = ',$)
          READ(5,'(10A1)')(P(L),L=1,10)
          CALL CHECK(P,INVAL,RNEW)
          IF(INVAL .EQ. 1)THEN
            WRITE(1,170)
            GO TO 330
          ELSE
```

```
          A(I,J)=RNEW
        ENDIF
350   CONTINUE
C     CONSTRAINT INEQUALITY INPUT BY USER
360   WRITE(1,'(7X,''INEQUALITY'',5X,$)')
      READ(5,'(A1)')P(1)
      CALL CHECK3(P,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,170)
        GO TO 360
      ELSE
        INEQ(I)=INEW
      ENDIF
C     COUNT OF EACH TYPE INEQUALITY PERFORMED
      IF(INEW .EQ. 0)THEN
        NLC=NLC + 1
        PINEQ(I)='<'
      ELSEIF(INEW .EQ. 1)THEN
        NGC=NGC + 1
        PINEQ(I)='>'
      ELSE
        NEC=NEC + 1
        PINEQ(I)='='
      ENDIF
C     CONSTRAINT RHS INPUT BY USER
370   WRITE(1,'(7X,''RHS'',12X,''= '',$)')
      READ(5,'(10A1)')(P(L),L=1,10)
      CALL CHECK(P,INVAL,RNEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,170)
        GO TO 370
      ELSE
        B(I)=RNEW
      ENDIF
      WRITE(1,100)CHAR(12)
400   CONTINUE
410   WRITE(1,100)CHAR(12)
      WRITE(1,160)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
C       CONTROL PASSED TO CONSTRAINT CHANGE ROUTINE
        CALL ICNRCH
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,170)
        PAUSE
        GO TO 410
      ENDIF
430   IFLAG(3)=0
      IFLAG(2)=0
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT12                                                         *
C    UNIT $USES: UNIT13 THRU UNIT17                                        *
C                                                                          *
C  SUBROUTINE EDIT                                                         *
C  USE: SOLICITS USER INPUT OF THE TYPE CHANGE REQUIRED TO MODEL,          *
C       SETS FLAGS AND CALLS PROPER SUBROUTINE TO PERFORM INPUTED          *
C       TYPE CHANGE.  FLAGS CAUSE ONLY CHANGES REQUESTED TO BE             *
C       INITIALLY ACCESSIBLE TO USER.  USED TO CORRECT MOST RECENT         *
C       MODEL INPUT OR EDIT MODEL READ FROM DISK.                          *
C  CALLED BY: PROGRAM DATAB                                                *
C  CALLS    : SUBROUTINE ADCON                                             *
C             SUBROUTINE ADVAR                                             *
C             SUBROUTINE CNVA                                              *
C             SUBROUTINE DELCON                                            *
C             SUBROUTINE DELVAR                                            *
C             SUBROUTINE ICNRCH                                            *
C             SUBROUTINE OBJCH                                             *
C             SUBROUTINE VNCH                                              *
C  VARIABLES:                                                              *
C       USED: INEW,INVAL                                                   *
C   MODIFIED: IFLAG(2),IFLAG(6),IFLAG(8),P(*)                              *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK IN UNIT17.CODE OVERLAY
$USES USAVE IN UNIT16.CODE OVERLAY
$USES UICNRCH IN UNIT15.CODE OVERLAY
$USES UCNVA IN UNIT14.CODE OVERLAY
$USES UADVAR IN UNIT13.CODE OVERLAY
      SUBROUTINE EDIT
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(12X,''DATA BASE EDITOR''//''YOU MAY EDIT THE CURRENT MOD
     .EL IN ANY OF''/''THE FOLLOWING MANNERS:''/''1.  ADD A VARIABLE''/
     .''2.  ADD A CONSTRAINT''/''3.  DELETE A VARIABLE''/''4.  DELETE A
     . CONSTRAINT'')')
      WRITE(1,'(''5.  CHANGE COEFFICIENT BY CONSTRAINT''/''6.  CHANGE CO
     .EFFICIENTS BY VARIABLE'')')
      WRITE(1,'(''7.  CHANGE RHS OF CONSTRAINT''/''8.  CHANGE CONSTRAINT
     . INEQUALITY''/''9.  CHANGE OBJECTIVE FUNCTION COST''/4X,''COEFFICI
     .ENTS''/''10. CHANGE MAXIMIZATION/MINIMIZATION'')')
      WRITE(1,'(4X,''CHOICE''/''11. CHANGE VARIABLE NAMES''/''12. CHANGE
     . CONSTRAINT NAMES''/''13. RETURN TO LAST MENU''/4X,''(DATA BASE MA
     .NAGEMENT)'')')
C     USER INPUTS THE TYPE CHANGE DESIRED IN MODEL
120   WRITE(1,'(/13X,''WHICH OPTION?  '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,13,INVAL,INEW)
```

```
      IF(INVAL .EQ. 1)THEN
        WRITE(1,'(5X,''INVALID ENTRY, PLEASE REENTER'')')
        GO TO 120
      ENDIF
      GOTO(210,220,230,240,250,260,270,280,290,300,310,320,330)INEW
C     CONTROL IS PASSED TO APPROPRIATE ROUTINE WITH FLAGS DESIGNATING
C       TYPE CHANGE ALLOWED TO BE PERFORMED
210   CALL ADVAR
      GO TO 340
220   CALL ADCON
      GO TO 340
230   CALL DELVAR
      GO TO 340
240   CALL DELCON
      GO TO 340
250   IFLAG(6)=1
      CALL ICNRCH
      GO TO 340
260   CALL CNVA
      GO TO 340
270   IFLAG(6)=3
      CALL ICNRCH
      GO TO 340
280   IFLAG(6)=2
      CALL ICNRCH
      GO TO 340
290   IFLAG(8)=1
      CALL OBJCH
      GO TO 340
300   IFLAG(8)=2
      CALL OBJCH
      GO TO 340
310   CALL VNCH
      GO TO 340
320   IFLAG(6)=4
      CALL ICNRCH
      GO TO 340
330   RETURN
C     FLAGS RESET TO DEFAULT VALUES
340   IFLAG(2)=0
      IFLAG(6)=0
      IFLAG(8)=0
      GO TO 100
      END
```

```
C ***************************************
C  MODULE 1 UNIT12                      *
C                                       *
C  SUBROUTINE VNCH                      *
C  USE: PERFORMS USER DESIRED CHANGES TO VARIABLE NAMES OF A MODEL    *
C       WHICH HAS BEEN DESIGNATED INITIALLY AS A MODEL CONTAINING    *
C       NAMED VARIABLES.                *
C  CALLED BY: SUBROUTINE DATAN          *
C             SUBROUTINE EDIT           *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)   *
C  VARIABLES:                           *
C       USED: IFLAG(5),INEW,INVAL,V     *
C   MODIFIED: IFLAG(2),P(*),RES,VN(*)   *
C ***************************************
      SUBROUTINE VNCH
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10,RES*6
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      WRITE(1,100)CHAR(12)
100   FORMAT(A)
      IF(IFLAG(5) .EQ. 0)THEN
        WRITE(1,'(7(/),16X,''MISTAKE!''//''THE MODEL BEING EDITED DOES N
     .OT INCLUDE''/''VARIABLE NAMES, ONLY SUBSCRIPTS.''//''YOU ARE BEING
     . RETURNED TO THE DATA BASE''/''EDITOR''//)')
        PAUSE
        RETURN
      ENDIF
110   WRITE(1,100)CHAR(12)
      WRITE(1,'(11(/),''DO YOU WANT PRESENT NAMES DISPLAYED? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,100)CHAR(12)
        WRITE(1,'(13X,''VARIABLE NAMES''/)')
        DO 140 J=1,V
          WRITE(1,120)J,VN(J)
120       FORMAT(13X,'X(',I2,') = ',A6)
140     CONTINUE
        PAUSE
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,160)
160     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 110
      ENDIF
180   WRITE(1,100)CHAR(12)
190   WRITE(1,'( 6(/),''WHICH VARIABLE NAME IS TO BE CHANGED?''/6X,''PLE
     .ASE ENTER SUBSCRIPT VALUE.'')')
200   WRITE(1,'(/10X,''VARIABLE X(?)  = '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,V,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
```

```
          WRITE(1,160)
          GO TO 200
        ENDIF
        WRITE(1,'(//10X,''PRESENT'',7X,''DESIRED''/11X,''NAME'',10X,''NAME
     .''/)')
        WRITE(1,'(10X,A6,8X,$)')VN(INEW)
        READ(5,'(A6)')RES
C       USER GIVEN OPTION TO DELETE REQUESTED CHANGE
220     WRITE(1,'(/8X,''IS CHANGE STILL DESIRED? '',$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          VN(INEW)=RES
          WRITE(1,'(/11X,''1 CHANGE COMPLETED'')')
          PAUSE
        ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
          WRITE(1,'(/11X,''NO CHANGES PERFORMED'')')
          PAUSE
        ELSE
          WRITE(1,160)
          GO TO 220
        ENDIF
230     WRITE(1,100)CHAR(12)
        WRITE(1,'(11(/),1X,''FURTHER VARIABLE NAME CHANGES NEEDED?''/19X,
     .$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          WRITE(1,100)CHAR(12)
          GO TO 110
        ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
          IFLAG(2)=0
          RETURN
        ELSE
          WRITE(1,160)
          GO TO 230
        ENDIF
        END
```

```
C ****************************************
C  MODULE 1 UNIT13                                                      *
C     UNIT $USES: UNIT14 AND UNIT17                                     *
C                                                                       *
C  SUBROUTINE ADVAR                                                     *
C  USE: PERFORMS THE ADDITION OF A VARIABLE TO THE MODEL BY             *
C       SOLICITING USER INPUTS FOR VARIABLE COEFFICIENTS IN ALL         *
C       CONSTRAINTS.  MODIFIES NECESSARY VARIABLES TO REFLECT           *
C       ADDITION OF VARIABLE TO MODEL AND REORGANIZES DATA IN ARRAYS.   *
C  CALLED BY: SUBROUTINE EDIT                                           *
C  CALLS    : SUBROUTINE CHECK(P,INVAL,RNEW)                            *
C             SUBROUTINE CHECK2(P,N,M,INVAL,INEW,)                      *
C             SUBROUTINE CNVA                                           *
C             SUBROUTINE OBJCH                                          *
C  VARIABLES:                                                           *
C       USED: CN(*),IFLAG(5),INEW,INVAL,X,RNEW                          *
C   MODIFIED: A(*,*),C(*),IFLAG(2),P(*),V,VN(*)                         *
C ****************************************
$USES UCHECK IN UNIT17.CODE OVERLAY
$USES UCNVA IN UNIT14.CODE OVERLAY
      SUBROUTINE ADVAR
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
C     NUMBER OF VARIABLES INCREASED BY 1
      V=V+1
      WRITE(1,130)PN
130   FORMAT(5X,'PROBLEM ID: ',A20)
C     DETERMINES IF MODEL CONTAINS NAMES
      IF(IFLAG(5) .EQ. 0)THEN
C       MODEL DOES CONTAIN NAMES
        WRITE(1,'(10X,''VARIABLE NAME INPUT''//''ENTER VARIABLE NAME WHI
     .CH CORRESPONDS''/''TO VARIABLE X(''.I2,'').''//''NAMES ARE TO BE 6
     . CHARACTERS OR LESS.'')')V
        WRITE(1,'(10X,''VARIABLE X('',I2,'') = '',$)')V
        READ(5,'(A6)')VN(V)
        WRITE(1,110)CHAR(12)
        WRITE(1,130)PN
        WRITE(1,'(7X,''VARIABLE COEFFICIENT INPUT''/8X,''VARIABLE X('',I
     .2,'') = '',A6,/)')V,VN(V)
C       VARIABLE COEFFICIENT INPUT FOR EACH CONSTRAINT
        DO 200 I=1,K
140       WRITE(1,150)I,CN(I)
150       FORMAT(1X,'CONSTRAINT #',I2,') = ',A6,$)
          READ(5,'(10A1)')(P(L),L=1,10)
          CALL CHECK(P,INVAL,RNEW)
          IF(INVAL .EQ. 1)THEN
            WRITE(1,170)
```

```
170         FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
            GO TO 140
          ELSE
            A(I,V)=RNEW
          ENDIF
200     CONTINUE
        PAUSE
      ELSE
C       MODEL DOES NOT CONTAIN NAMES
        WRITE(1,'(7X,''VARIABLE COEFFICIENT INPUT''/13X,''VARIABLE X('',
     .I2,'')''/)')V
        DO 250 I=1,K
210       WRITE(1,220)I
220       FORMAT(6X,'CONSTRAINT #',I2,' = ',$)
          READ(5,'(10A1)')(P(L),L=1,10)
          CALL CHECK(P,INVAL,RNEW)
          IF(INVAL .EQ. 1)THEN
            WRITE(1,170)
            GO TO 210
          ELSE
            A(I,V)=RNEW
          ENDIF
250     CONTINUE
        PAUSE
      ENDIF
260   WRITE(1,110)CHAR(12)
      WRITE(1,130)PN
C     VARIABLE COST COEFFICIENT INPUT BY USER
270   IF(IFLAG(5) .EQ. 1)THEN
        WRITE(1,'(7X,''OBJECTIVE COEFFICIENT INPUT''/8X,''VARIABLE X('',
     .I2,'') = '',A6)')V,VN(V)
        WRITE(1,280)V
280     FORMAT(//11X,'C(',I2,') = ',$)
        READ(5,'(10A1)')(P(L),L=1,10)
      ELSE
        WRITE(1,'(7X,''OBJECTIVE COEFFICIENT INPUT''/13X,''VARIABLE X(''
     .,I2,'')'')')V
        WRITE(1,280)V
        READ(5,'(10A1)')(P(L),L=1,10)
      ENDIF
      CALL CHECK(P,INVAL,RNEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,170)
        GO TO 270
      ENDIF
      C(V)=RNEW
290   WRITE(1,110)CHAR(12)
      WRITE(1,300)
300   FORMAT(11(/),7X,'ARE CORRECTIONS NEEDED ON THIS VARIABLE?'/19X,$)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,110)CHAR(12)
```

```
      WRITE(1,'(8(/),10X,''CHANGES REQUIRED IN:''//5X,''1. OBJECTIVE C
     .OST COEFFICIENT''/19X,''OR''/5X,''2. CONSTRAINT COEFFICIENT'')')
320     WRITE(1,'(/13X,''WHICH OPTION? '',$)')
        READ(5,'(A1)')P(1)
        CALL CHECK2(P,1,2,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,170)
          GO TO 320
        ENDIF
        IF(INEW .EQ. 1)THEN
C         USER ELECTS TO CHANGE COST COEFFICIENTS
          CALL OBJCH
        ELSE
C         USER ELECTS TO CHANGE CONSSTRAINT COEFFICIENT
          CALL CNVA
        ENDIF
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,170)
        GO TO 290
      ENDIF
330   WRITE(1,110)CHAR(12)
340   WRITE(1,'(11(/),8X,''ADD ANOTHER VARIABLE? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
C       USER HAS SELECTED TO ADD ANOTHER VARIABLE
        GO TO 100
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,170)
        GO TO 330
      ENDIF
      IFLAG(2)=0
      RETURN
      END
```

```
C ****************************************
C  MODULE 1 UNIT13                                               *
C                                                                *
C  SUBROUTINE OBJCH                                              *
C  USE: PERFORMS USER INPUT CHANGES TO OBJECTIVE FUNCTION        *
C       COEFFICIENTS AND MAXIMIZATION/MINIMIZATION CHOICE OF THE *
C       CURRENT MODEL.  USED IN CORRECTION OF MOST RECENT MODEL INPUT *
C       OR MODEL FROM DISK BEING EDITED.                         *
C  CALLED BY: SUBROUTINE ADVAR                                   *
C             SUBROUTINE DATAN                                   *
C             SUBROUTINE DATAS                                   *
C             SUBROUTINE EDIT                                    *
C  CALLS    : SUBROUTINE CHECK(P,INVAL,RNEW)                     *
C             SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                *
C  VARIABLES:                                                    *
C       USED: IFLAG(5),INEW,INVAL,OBJN,PN,RNEW,V,VN(*)           *
C   MODIFIED: C(*),IFLAG(2),IFLAG(8),MM,MXMN,F(*)                *
C ****************************************
      SUBROUTINE OBJCH
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      IF(IFLAG(8) - 1) 120,200,400
C     ACCESS TO CHANGE ALL OF OBJ FUNCTION
120   IF(IFLAG(5) .EQ. 0)THEN
        WRITE(1,130)PN,MM
130     FORMAT(5X,'PROBLEM ID: ',A20/7X,'OBJECTIVE FUNCTION CHANGE'/14X,
     .A3,'IMIZATION'/)
      ELSE
        WRITE(1,140)PN,OBJN,MM
140     FORMAT(5X,'PROBLEM ID: ',A20/7X,'OBJECTIVE FUNCTION CHANGE'/7X,A
     ,10,2X,A3,'IMIZATION'/)
      ENDIF
      WRITE(1,'(//2X,''WHICH OF THE BELOW REQUIRES CHANGES?''//4X,''1. C
     .OST COEFFICIENTS''/19X,''OR''/4X,''2. MAXIMIZATION/MINIMIZATION CH
     .OICE'')')
150   WRITE(1,'(/13X,''WHICH OPTION?  '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,2,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,160)
160     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 150
      ENDIF
      WRITE(1,110)CHAR(12)
      GOTO(200,400)INEW
C     ACCESS TO CHANGE OBJ FUNCTION COEFFIECIENTS ONLY
200   IF(IFLAG(5) .EQ.  0)THEN
```

```
        WRITE(1,130)PN,MM
      ELSE
        WRITE(1,140)PN,OBJN,MM
      ENDIF
210   WRITE(1,'(////2X,''DISPLAY THE PRESENT COEFFICIENTS?  '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,110)CHAR(12)
        IF(IFLAG(5) .EQ. 0)THEN
          WRITE(1,130)PN,MM
        ELSE
          WRITE(1,140)PN,OBJN,MM
        ENDIF
        DO 230 J=1,V
          IF(IFLAG(5) .EQ. 0)THEN
            WRITE(1,'(10X,''C('',I2,'') = '',1PE12.5)')J,C(J)
          ELSE
            WRITE(1,'(5X,''C('',I2,'') = '',A6,'' = '',1PE12.5)')J,VN(J)
     .,C(J)
          ENDIF
230     CONTINUE
        PAUSE
        WRITE(1,110)CHAR(12)
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,160)
        GO TO 210
      ENDIF
      WRITE(1,110)CHAR(12)
      IF(IFLAG(5) .EQ. 0)THEN
        WRITE(1,130)PN,MM
      ELSE
        WRITE(1,140)PN,OBJN,MM
      ENDIF
250   WRITE(1,'(/3X,''WHICH COEFFICIENT IS TO BE CHANGED?''/6X,''PLEASE
     . ENTER SUBSCRIPT VALUE.'')')
260   WRITE(1,'(/7X,''COST COEFFICIENT C(?)  = '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,V,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,160)
        GO TO 260
      ENDIF
270   WRITE(1,'(/6X,''PRESENT''.14X,''DESIRED''/7X.''C('',I2,'')'',16X,
     .''C('',I2,'')'')')INEW,INEW
      WRITE(1,'(3X,1PE12.5,9X.$)')C(INEW)
      READ(5,'(10A1)')(P(L),L=1,10)
      CALL CHECK(P,INVAL,RNEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,160)
        GO TO 270
      ENDIF
280   WRITE(1,'(/8X,''IS CHANGE STILL DESIRED? '',$)')
```

```
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        C(INEW)=PNEW
        WRITE(1,290)
290     FORMAT(/11X,'1 CHANGE COMPLETED')
        PAUSE
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        WRITE(1,300)
300     FORMAT(/10X,'NO CHANGES PERFORMED')
        PAUSE
      ELSE
        WRITE(1,160)
        GO TO 280
      ENDIF
310   WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),''FURTHER COST COEFFICIENT CHANGES?  '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,110)CHAR(12)
        GO TO 250
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,160)
        GO TO 310
      ENDIF
330   WRITE(1,110)CHAR(12)
340   WRITE(1,350)
350   FORMAT(11(/),'FURTHER OBJECTIVE FUNCTION CHANGES?  ',$)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,110)CHAR(12)
        GO TO 120
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,160)
        GO TO 330
      ENDIF
      GO TO 500
C     ACCESS TO CHANGE MAX/MIN CHOICE
400   WRITE(1,110)CHAR(12)
      IF(IFLAG(5) .EQ. 0)THEN
        WRITE(1,130)PN,MM
      ELSE
        WRITE(1,140)PN,OBJN,MM
      ENDIF
      WRITE(1,'(//5X,''PRESENT CHOICE IS '',A3,''IMIZATION.'')')MM
420   WRITE(1,'(/5X,''WOULD YOU LIKE THIS CHANGED? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ.  89)THEN
        IF(MXMN .EQ.1)THEN
          MXMN=0
          MM='MIN'
        ELSE
          MXMN=1
```

```
        MM='MAX'
      ENDIF
      WRITE(1,290)
    ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
      WRITE(1,300)
    ELSE
      WRITE(1,160)
      GO TO 420
    ENDIF
    WRITE(1,'(4(/))')
    PAUSE
    GO TO 330
500 IFLAG(2)=0
    IFLAG(8)=0
    RETURN
    END
```

```
C ****************************************
C  MODULE 1 UNIT13                                              *
C                                                               *
C  SUBROUTINE GENIF                                             *
C  USE: SOLICITS INPUT OF THE PROBLEM NAME, MAXIMIZATION/MINIMIZATION *
C       CHOICE, OBJECTIVE FUNCTION NAME, NUMBER OF CONSTRAINTS AND    *
C       VARIABLES FROM USER.  USED ONLY FOR INPUT OF NEW MODELS.      *
C  CALLED BY: PROGRAM DATAB                                     *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)               *
C  VARIABLES:                                                   *
C       USED: IFLAG(5),INEW,INVAL                               *
C   MODIFIED: K,MM,MXMN,OBJN,P(*),PN,V                          *
C ****************************************
      SUBROUTINE GENIF
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NEC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
C     PROBLEM NAME ENTERED FOR NEW MODEL
      WRITE(1,'(///7X,''ENTER A PROBLEM IDENTIFIER''/7X,''(MAXIMUM OF 20
     . CHARACTERS)''//3X,''PROBLEM ID = '',$)')
      READ(5,'(A20)')PN
100   WRITE(1,'(/40(''*''))')
C     MAX/MIN CHOICE ENTERED
      WRITE(1,'(4(/),1X,''IS PROBLEMS OBJECTIVE FUNCTION TO BE:''// 14X,
     .''1. MAXIMIZED''/19X,''OR''/14X,''2. MINIMIZED'')')
110   WRITE(1,'(/13X,''WHICH OPTION?  '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,2,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,120)
120     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 110
      ENDIF
      MXMN=INEW
C     CHARACTER STRING ESTABLISHED REPRESENTING MAX/MIN CHOICE
      IF(MXMN .EQ. 1)THEN
        MM='MAX'
      ELSE
        MM='MIN'
      ENDIF
      IF(IFLAG(5) .EQ. 1)THEN
C       MODEL INCLUDES NAMES SO OBJ FUNCTION NAME INPUT
        WRITE(1,'(A)')CHAR(12)
        WRITE(1,'(7(/),1X,''WHAT IS THE NAME OF THE OBJECTIVE YOU''/1X,
     .11X''WANT TO '',A3,''IMIZE?''//1X,''(FOR EXAMPLE, COST, MANPOWER,
     . ETC.)''//1X,''MAXIMUM OF 10 CHARACTERS ALLOWED''//6X,''OBJECTIVE
     . NAME = '',$)')MM
        READ(5,'(A10)')OBJN
      ENDIF
      WRITE(1,'(A)')CHAR(12)
```

```
C     NUMBER OF CONSTRAINTS ENTERED
      WRITE(1,'(///1X,''ENTER NUMBER OF CONSTRAINTS IN PROBLEM''/13X,''(
     .MAXIMUM OF 20)'')')
130   WRITE(1,'(//9X,''NUMBER OF CONSTRAINTS = '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,20,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,120)
        GO TO 130
      ENDIF
      K=INEW
      WRITE(1,'(///40(''*''))')
C     NUMBER OF VARIABLES ENTERED
      WRITE(1,'(///2X,''ENTER NUMBER OF VARIABLES IN PROBLEM''/13X,''(MA
     .XIMUM OF 20)'')')
140   WRITE(1,'(//9X,''NUMBER OF VARIABLES = '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,20,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,120)
        GO TO 140
      ENDIF
      V=INEW
      WRITE(1,'(A)')CHAR(12)
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT14                                                      *
C    UNIT $USES: UNIT17                                                 *
C                                                                       *
C  SUBROUTINE CNVA                                                      *
C  USE: ALLOWS MODEL COEFFICIENTS TO BE CHANGED BY VARIABLE.            *
C       SOLICITS INPUT OF VARIABLE COEFFICIENT TO BE CHANGED BY         *
C       INPUTING VARIABLE NUMBER.  VARIABLE COEFFICIENT OF SPECIFIED    *
C       VARIABLE MAY BE CHANGED BY DESIGNATING CONSTRAINT NUMBER AND    *
C       THE DESIRED COEFFICIENT.                                        *
C  CALLED BY: SUBROUTINE ADVAR                                          *
C             SUBROUTINE EDIT                                           *
C  CALLS    : SUBROUTINE CHECK(P,INVAL,RNEW)                            *
C             SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                       *
C  VARIABLES:                                                           *
C       USED: A(*,*),CN(*),IFLAG(5),INEW,INVAL,K,RNEW,V,VN(*)           *
C   MODIFIED: CHAK,CHARO,IFLAG(2),P(*)                                  *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK IN UNIT17.CODE OVERLAY
      SUBROUTINE CNVA
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V,CHARO,CHAK
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(''WHICH VARIABLE COEFFICIENTS REQUIRE''/16X''CHANGES?''/
     ./6X,''PLEASE ENTER SUBSCRIPT VALUE.'')')
130   WRITE(1,'(/10X,''VARIABLE X(?)  = '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,V,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,140)
140     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 130
      ENDIF
      CHAK=INEW
      WRITE(1,110)CHAR(12)
      WRITE(1,'(2X,''VARIABLE COEFFICIENT BY CONSTRAINT'')')
      IF(IFLAG(5) .EQ. 1)THEN
        WRITE(1,'(29X,A6/29X,''X('',I2,'')'')')VN(CHAK),CHAK
        DO 160 I=1,K
          WRITE(1,'(2X,''CONSTRAINT #'',I2,2X,A6,2X,1P,E12.6)')I,CN(I),
     .A(I,CHAK)
160     CONTINUE
        PAUSE
      ELSE
        WRITE(1,'(29X,''X('',I2,'')'')')CHAK
        DO 180 I=1,K
          WRITE(1,'(2X,''CONSTRAINT #'',I2,10X,E12.6)')I,A(I,CHAK)
180     CONTINUE
```

```
      PAUSE
     ENDIF
190  WRITE(1,110)CHAR(12)
     WRITE(1,'(10(/),''WHICH CONSTRAINT REQUIRES A CHANGE IN''/9X,''THE
    . X('',I2,'') COEFFICIENT?''/4X,''PLEASE ENTER CONSTRAINT NUMBER.''
    .)')CHAK
200  WRITE(1,'(/8X,''CONSTRAINT NUMBER = '',$)')
     READ(5,'(2A1)')P(1),P(2)
     CALL CHECK2(P,2,K,INVAL,INEW)
     IF(INVAL .EQ. 1)THEN
       WRITE(1,140)
       GO TO 200
     ENDIF
     CHARO=INEW
210  WRITE(1,'(/6X,''PRESENT'',14X,''DESIRED''/7X,''X('',I2,'')'',16X,
    .''X('',I2,'')'')')CHAK,CHAK
     WRITE(1,'(3X,1PE12.5,9X,$)')A(CHARO,CHAK)
     READ(5,'(10A1)')(P(L),L=1,10)
     CALL CHECK(P,INVAL,RNEW)
     IF(INVAL .EQ. 1)THEN
       WRITE(1,140)
       GO TO 210
     ENDIF
220  WRITE(1,'(/6X,''IS CHANGE STILL DESIRED? '',$)')
     READ(5,'(A1)')P(1)
     IF(ICHAR(P(1)) .EQ. 89)THEN
       A(CHARO,CHAK)=RNEW
       WRITE(1,'(/11X,''1 CHANGE COMPLETED'')')
       PAUSE
     ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
       WRITE(1,'(/10X,''NO CHANGES PERFORMED'')')
       PAUSE
     ELSE
       WRITE(1,140)
       GO TO 220
     ENDIF
230  WRITE(1,110)CHAR(12)
     WRITE(1,'(10(/),''FURTHER COEFFICIENT CHANGES OF SAME''/''VARIABL
    .E IN DIFFERENT CONSTRAINT?''/19X,$)')
     READ(5,'(A1)')P(1)
     IF(ICHAR(P(1)) .EQ. 89)THEN
       GO TO 190
     ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
       IFLAG(2)=0
       RETURN
     ELSE
       WRITE(1,140)
       GO TO 230
     ENDIF
240  WRITE(1,110)CHAR(12)
     WRITE(1,'(10(/),''FURTHER COEFFICIENT CHANGES OF DIFFERENT''/''VAR
    .IABLE?''/19X,$)')
```

```
READ(5,'(A1)')P(1)
IF(ICHAR(P(1)) .EQ. 89)THEN
  GO TO 100
ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
  IFLAG(2)=0
  RETURN
ELSE
  WRITE(1,140)
  GO TO 240
ENDIF
END
```

```
C ***************************************
C  MODULE 1 UNIT14                                                    *
C                                                                     *
C  SUBROUTINE DELCON                                                  *
C  USE: PERFORMS THE DELETION OF A CONSTRAINT FROM MODEL BY           *
C       SOLICITING USER INPUT OF CONSTRAINT NUMBER.  MODIFIES         *
C       NECESSARY VARIABLES TO REFLECT DELETION OF CONSTRAINT AND     *
C       REORGANIZES DATA IN ARRAYS.                                   *
C  CALLED BY: SUBROUTINE EDIT                                         *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                     *
C  VARIABLES:                                                         *
C       USED: IFLAG(5),INEW,INVAL,V                                   *
C   MODIFIED: A(*,*),B(*),CN(*),IFLAG(2),INEQ(*),K,NEC,NGC,NLC,P(*),  *
C             PINEQ(*)                                                *
C ***************************************
      SUBROUTINE DELCON
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      IF(IFLAG(5) .EQ. 1)THEN
120     WRITE(1,'(11(/),''NEED TO SEE CONSTRAINT NAME LIST? '',$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          WRITE(1,110)CHAR(12)
          WRITE(1,'(8X,''CONSTRAINT NAME LISTING''/)')
          DO 150 I=1,K
140         WRITE(1,'(7X,''CONSTRAINT # '',I2,'' = '',A6)')I,CN(I)
150       CONTINUE
          PAUSE
        ELSEIF(ICHAR(P(1)) .NE. 78)THEN
          WRITE(1,160)
160       FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
          GO TO 100
        ENDIF
      ENDIF
200   WRITE(1,110)CHAR(12)
      WRITE(1,'(10(/),''WHICH CONSTRAINT DO YOU WISH TO DELETE?''/5X,''P
     .LEASE ENTER CONSTRAINT NUMBER.''/)')
220   WRITE(1,'(10X,''DELETE CONSTRAINT # ''.$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,K,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,160)
        GO TO 220
      ENDIF
      B(INEW)=0.
      DO 250 J=1,V
        A(INEW,J)=0.
```

```
250   CONTINUE
C     COUNT BY TYPE OF INEQUALITY UPDATED SINCE 1 LESS CONSTRAINT
      IF(INEQ(INEW) .EQ. 0)THEN
        NLC=NLC - 1
      ELSEIF(INEQ(INEW) .EQ. 1)THEN
        NGC=NGC - 1
      ELSE
        NEC=NEC - 1
      ENDIF
C     IF CONSTRAINT NOT LAST CONSTRAINT, ALL ROWS MOVED UP 1
      IF(INEW .LT. K)THEN
        DO 300 I=INEW,K-1
          B(I)=B(I+1)
          INEQ(I)=INEQ(I+1)
          PINEQ(I)=PINEQ(I+1)
          CN(I)=CN(I+1)
          DO 290 J=1,V
            A(I,J)=A(I+1,J)
290       CONTINUE
300     CONTINUE
      ENDIF
C     NUMBER OF CONSTRAINT DECREASED BY 1
      K=K-1
      IFLAG(2)=0
      RETURN
      END
```

```
C ***************************************
C  MODULE 1 UNIT14                                                   *
C                                                                    *
C  SUBROUTINE DELVAR                                                 *
C  USE: PERFORMS THE DELETION OF A VARIABLE FROM MODEL BY SOLICITING *
C       USER INPUT OF VARIABLE NUMBER.  MODIFIES NECESSARY VARIABLES *
C       TO REFLECT VARIABLE DELETION AND REORGANIZES DATA IN ARRAYS. *
C  CALLED BY: SUBROUTINE EDIT                                        *
C  CALLS     : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                   *
C  VARIABLES:                                                        *
C       USED: IFLAG(5),INVAL,INEW,K                                  *
C   MODIFIED: A(*,*),C(*).IFLAG(2),P(*),V,VN(*)                      *
C ***************************************
      SUBROUTINE DELVAR
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,F*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      IF(IFLAG(5) .EQ. 1)THEN
C       MODEL CONTAINS NAMES
120     WRITE(1,'(11(/),''NEED TO SEE VARIABLE NAME LISTING? '',$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          WRITE(1,110)CHAR(12)
          WRITE(1,'(9X,''VARIABLE NAME LISTING''/)')
          DO 150 J=1,V
            WRITE(1,'(8X,''VARIABLE X('',I2,'')  = '',A6)')J,VN(J)
150       CONTINUE
          PAUSE
        ELSEIF(ICHAR(P(1)) .NE. 78)THEN
          WRITE(1,160)
160       FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
          GO TO 100
        ENDIF
      ENDIF
200   WRITE(1,110)CHAR(12)
      WRITE(1,'(10(/),''WHICH VARIABLE DO YOU WISH TO DELETE?''/5X,''PLE
     .ASE ENTER SUBSCRIPT VALUE.''/)')
220   WRITE(1,'(9X,''DELETE VARIABLE X(?) = '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,V,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,160)
        GO TO 220
      ENDIF
C     COST COEFFICIENT FOR DELETED VARIABLE ZEROED
      C(INEW)=0.
C     CONSTRAINT COEFFICIENTS FOR DELETED VARIABLE ZEROED
      DO 250 I=1,K
```

```
      A(K,INEW)=0.
250   CONTINUE
C     IF VARIABLE NOT LAST VARIABLE, ALL COLUMNS MOVED LEFT 1
      IF(INEW .LT. V)THEN
        DO 300 J=INEW,V-1
          C(J)=C(J+1)
          IF(IFLAG(5) .EQ. 1)THEN
            VN(J)=VN(J+1)
          ENDIF
          DO 290 I=1,K
            A(I,J)=A(I,J+1)
290       CONTINUE
300     CONTINUE
      ENDIF
C     NUMBER OF VARIABLES DECREASED BY 1
      V=V-1
320   WRITE(1,110)CHAR(12)
      WRITE(1,'(10(/),8X,''DELETE ANOTHER VARIABLE? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 100
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,160)
        GO TO 320
      ENDIF
      IFLAG(2)=0
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT 15                                                        *
C    UNIT $USES: UNIT17                                                    *
C                                                                          *
C  SUBROUTINE ICNRCH                                                       *
C  USE: SOLICITS INPUT OF CONSTRAINT NUMBER AND TYPE OF CHANGE             *
C       DESIRED IN CONSTRAINT.  ALLOWS USER TO CHANGE CONSTRAINT           *
C       COEFFICIENT, INEQUALITY, RHS OR NAME.  PERFORMS DESIRED            *
C       CHANGE AND MODIFIES THOSE VARIABLES WHICH THE CHANGE               *
C       NECESSITATES.                                                      *
C  CALLED BY: SUBROUTINE ADCON                                             *
C             SUBROUTINE DATAN                                             *
C             SUBROUTINE DATAS                                             *
C             SUBROUTINE EDIT                                              *
C  CALLS    : SUBROUTINE CHECK(P,INVAL,RNEW)                               *
C             SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                          *
C             SUBROUTINE CHECK3(P,INVAL,INEW)                              *
C             SUBROUTINE DISPLY                                            *
C  VARIABLES:                                                              *
C       USED: IFLAG(5),INEW,INVAL,K,RNEW,V                                 *
C   MODIFIED: A(*,*),B(*),CHAK,CHAN,CHARO,CN(*),IFLAG(2),IFLAG(6),         *
C             IFLAG(9),INEQ(*),NEC,NGC,NLC,P(*),PINEQ(*)                   *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK IN UNIT17.CODE OVERLAY
      SUBROUTINE ICNRCH
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10,CHAN*6
      INTEGER V,CHAK,CHARO
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,120)
120   FORMAT(11X,'CONSTRAINT CHANGE')
C     CHECKS IF ONLY CONSTRAINT NAME IS TO BE CHANGED
      IF(IFLAG(6) .EQ. 4)THEN
        GO TO 140
      ENDIF
      WRITE(1,'(10(/),1X,''DISPLAY THE PRESENT CONSTRAINTS? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 78)THEN
        GO TO 140
      ELSEIF(ICHAR(P(1)) .NE. 89)THEN
        WRITE(1,130)
130     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 100
      ENDIF
C     FLAG ALLOWS ONLY CONSTRAINTS TO BE DISPLAYED
      IFLAG(9)=1
      CALL DISPLY
      WRITE(1,110)CHAR(12)
140   WRITE(1,'( 4(/),''WHICH CONSTRAINT DO YOU WISH TO CHANGE?''/5X,''P
```

```
      .LEASE ENTER CONSTRAINT NUMBER.'')')
150   WRITE(1,'(/9X,''CHANGE CONSTRAINT #'',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,K,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 150
      ENDIF
      CHARO=INEW
190   GOTO(200,350,500,650,800)(IFLAG(6)+1)
C     ACCESS TO CHANGE ANY PART OF CONSTRAINT ALLOWED
200   WRITE(1,110)CHAR(12)
      WRITE(1,210)
210   FORMAT(11X,'CONSTRAINT CHANGE')
      WRITE(1,'(//2X,''CHANGE DESIRED IN CONSTRAINT #'',I2,'' IS:''//8X,
     .''1. VARIABLE COEFFICIENT''/8X,''2. INEQUALITY''/8X,''3. RHS''/5X,
     .''4. NO CHANGES'')')CHARO
220   WRITE(1,'(/13X,''WHICH OPTION? '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,4,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 220
      ENDIF
      GOTO(350,500,650,1300)INEW
C     ONLY VARIABLE COEFFICIENTS ALLOWED TO BE CHANGED
350   WRITE(1,110)CHAR(12)
      WRITE(1,'(5X,''WHICH VARIABLE COEFFICIENT OF''/4X,''CONSTRAINT '',
     .I2,'' REQUIRES CHANGES?'')')CHARO
360   WRITE(1,'(/10X,''VARIABLE X(?)  = '',$,2(/))')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,V,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 360
      ENDIF
      CHAK=INEW
370   WRITE(1,'(/6X,''PRESENT'',14X,''DESIRED''/7X,''X('',I2,'')'',16X,
     .''X('',I2,'')'')')CHAK,CHAK
      WRITE(1,'(3X,1PE12.5,9X,$)')A(CHARO,CHAK)
      READ(5,'(10A1)')(P(L),L=1,10)
      CALL CHECK(P,INVAL,RNEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 370
      ENDIF
380   WRITE(1,390)
390   FORMAT(/8X,'IS CHANGE STILL DESIRED? ',$)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        A(CHARO,CHAK)=RNEW
        WRITE(1,400)
```

```
400     FORMAT(/11X,'1 CHANGE COMPLETED')
        PAUSE
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        WRITE(1,410)
410     FORMAT(/10X,'NO CHANGE PERFORMED')
        PAUSE
      ELSE
        WRITE(1,130)
        GO TO 380
      ENDIF
420   WRITE(1,110)CHAR(12)
      WRITE(1,430)
430   FORMAT(10(/),'FURTHER COEFFICIENT CHANGES TO THIS'/13X,'CONSTRAINT
     .? ',$)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 200
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,130)
        GO TO 420
      ENDIF
      GO TO 1200
C     ONLY INEQUALITY CHANGES ALLOWED
500   WRITE(1,'(/6X,''CONSTRAINT INEQUALITY CHANGE'')')
510   WRITE(1,'(//6X,''PRESENT'',14X,''DESIRED'')')
      WRITE(1,'(9X,A1,20X,$)')PINEQ(CHARO)
      READ(5,'(A1)')P(1)
      CALL CHECK3(P,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 510
      ENDIF
520   WRITE(1,390)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
C       COUNT BY TYPE OF INEQUALITY UPDATED
        WRITE(1,400)
        IF(INEQ(CHARO) .EQ. 0)THEN
          NLC=NLC-1
        ELSEIF(INEQ(CHARO) .EQ. 1)THEN
          NGC=NGC-1
        ELSE
          NEC=NEC-1
        ENDIF
        INEQ(CHARO)=INEW
        IF(INEQ(CHARO) .EQ. 0)THEN
          NLC=NLC+1
          PINEQ(CHARO)='<'
        ELSEIF(INEQ(CHARO) .EQ. 1)THEN
          NGC=NGC+1
          PINEQ(CHARO)='>'
        ELSE
```

```
          NEC=NEC+1
          FINEQ(CHARO)='='
          ENDIF
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        WRITE(1,410)
      ELSE
        WRITE(1,130)
        GO TO 520
      ENDIF
      PAUSE
      GO TO 1200
C     ONLY RHS CHANGES ALLOWED
650   WRITE(1,'(/9X,''CONSTRAINT RHS CHANGE'')')
660   WRITE(1,'(6X,''PRESENT'',14X,''DESIRED''/7X,''B('',I2,'')'',16X,''
     .B('',I2,'')'')')CHARO,CHARO
      WRITE(1,'(3X,1PE12.5,9X,$)')B(CHARO)
      READ(5,'(10A1)')(P(L),L=1,10)
      CALL CHECK(P,INVAL,RNEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 660
      ENDIF
680   WRITE(1,390)
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        B(CHARO)=RNEW
        WRITE(1,400)
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        WRITE(1,410)
      ELSE
        WRITE(1,130)
        GO TO 680
      ENDIF
      PAUSE
      GO TO 1200
C     ONLY CONSTRAINT NAMES CHANGES ALLOWED
800   WRITE(1,110)CHAR(12)
      IF(IFLAG(5) .EQ. 0)THEN
C       MODEL DOES NOT INCLUDE NAMES
        WRITE(1,'(7(/),16X,''MISTAKE!''//''THE MODEL BEING EDITED DOES N
     .OT INCLUDE''/''CONSTRAINT NAMES, ONLY SUBSCRIPTS.''//''YOU ARE BEI
     .NG RETURNED TO DATA BASE''/''EDITOR''//)')
        PAUSE
        RETURN
      ENDIF
      WRITE(1,'(9X,''CONSTRAINT NAME CHANGE'')')
      WRITE(1,'(//''PRESENT NAME FOR CONSTRAINT #'',I2,'' = '',A6)')CHAR
     .O,CN(CHARO)
810   WRITE(1,'(//''DESIRED NAME FOR CONSTRAINT #'',I2,'' = '',$)')CHARO
      READ(5,'(A6)')CHAN
      WRITE(1,390)
      READ(5,'(A1)')P(1)
```

```
      IF(ICHAR(P(1)) .EQ. 89)THEN
        CN(CHARD)=CHAN
        WRITE(1,400)
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        WRITE(1,410)
      ELSE
        WRITE(1,130)
        GO TO 810
      ENDIF
      PAUSE
1200  WRITE(1,110)CHAR(12)
      IF(IFLAG(6) .EQ. 4)THEN
        GO TO 1220
      ENDIF
1210  WRITE(1,110)CHAR(12)
      WRITE(1,'(10(/),''FURTHER CHANGES TO THIS CONSTRAINT? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 200
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,130)
        GO TO 1210
      ENDIF
1220  WRITE(1,110)CHAR(12)
      WRITE(1,'(10(/),''FURTHER CHANGES TO ANY CONSTRAINT? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 100
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,130)
        GO TO 1220
      ENDIF
      IFLAG(2)=0
      IFLAG(6)=0
1300  RETURN
      END
```

```
C *************************************
C  MODULE 1 UNIT15                                               *
C                                                                 *
C  SUBROUTINE ADCON                                               *
C  USE: SOLICITS INPUT OF COEFFICIENTS, INEQUALITY, RHS,  AND NAME OF *
C       ADDED CONSTRAINT.  MODIFIES NECESSARY VARIABLES TO REFLECT    *
C       ADDITION OF CONSTRAINT TO MODEL.                          *
C  CALLED BY: SUBROUTINE EDIT                                     *
C  CALLS    : SUBROUTINE CHECK(P,INVAL,RNEW)                      *
C             SUBROUTINE CHECK3(P,INVAL,INEW)                     *
C             SUBROUTINE ICNRCH                                   *
C  VARIABLES:                                                     *
C       USED: IFLAG(5),INEW,INVAL,PN.RNEW.V,VN(*)                 *
C   MODIFIED: A(*,*),B(*),CN(*),IFLAG(2),INEQ(*),K,NEC,NGC,NLC,P(*), *
C             PINEQ(*)                                            *
C *************************************
      SUBROUTINE ADCON
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(12X,''CONSTRAINT INPUT''//''INPUT CONSTRAINT VARIABLE CO
     .EFFICIENTS''/''AS IF THE CONSTRAINT WAS IN THE''/13X,''FOLLOWING F
     .FORM'')')
      WRITE(1,'(6X,''X(1) + X(2) + X(3)  <=> RHS''//''THE VARIABLE COEFF
     .ICIENTS ARE A MAXIMUM''/''OF 10 CHARACTERS.''//''IF COEFFICIENT IS
     . ZERO, ENTER 0 OR HIT''/''"RETURN" WITHOUT ENTRY''//)')
      PAUSE
120   WRITE(1,110)CHAR(12)
C     NUMBER OF CONSTRAINTS UPDATED
      K=K + 1
      WRITE(1,130)PN
130   FORMAT(5X,'PROBLEM ID: ',A20)
      IF(IFLAG(5) .EQ. 1)THEN
C        MODEL INCLUDES NAMES SO CONSTRAINT NAME INPUT BY USER
         WRITE(1,'(/9X,''CONSTRAINT NAME INPUT''//''ENTER CONSTRAINT NAME
     . WHICH CORRESPONDS''/''TO CONSTRAINT #'',I2,''.''//''NAMES ARE TO
     . BE 6 CHARACTERS OR LESS.'')')K
         WRITE(1,'(/9X,''CONSTRAINT '',I2,'' = ''.$)')K
         READ(5,'(A6)')CN(K)
         WRITE(1,110)CHAR(12)
         WRITE(1,130)PN
         WRITE(1,'(''CONSTRAINT VARIABLE COEEFICIENT INPUT'')')
         WRITE(1,'(9X,''CONSTRAINT '',I2,'' = '',A6,/)')K,CN(K)
C        ADDED CONSTRAINT COEFFICIENTS INPUT BY USER
         DO 200 J=1,V
140        WRITE(1,150)J,VN(J)
150        FORMAT(7X,'X(',I2,') =',A6,' = ',$)
           READ(5,'(10A1)')(P(L),L=1,10)
```

```
            CALL CHECK(P,INVAL,RNEW)
            IF(INVAL .EQ. 1)THEN
              WRITE(1,160)
160           FORMAT(/5X,'INVALID,ENTRY, PLEASE REENTER')
              GO TO 140
            ELSE
              A(K,J)=RNEW
            ENDIF
200       CONTINUE
        ELSE
          WRITE(1,'(13X,''CONSTRAINT '',I2)')K
          DO 250 J=1,V
210         WRITE(1,220)J
220          FORMAT(6X,'X(',I2,') = ',$)
            READ(5,'(10A1)')(P(L),L=1,10)
            CALL CHECK(P,INVAL,RNEW)
            IF(INVAL .EQ. 1)THEN
              WRITE(1,160)
              GO TO 210
            ELSE
              A(K,J)=RNEW
            ENDIF
250       CONTINUE
        ENDIF
C       ADDED CONSTRAINT INEQUALITY INPUT BY USER
260     IF(IFLAG(5) .EQ. 0)THEN
          WRITE(1,'(1X,''INEQUALITY    '',$)')
        ELSE
          WRITE(1,'(1X,''INEQUALITY'',12X,$)')
        ENDIF
        READ(5,'(A1)')P(1)
        CALL CHECK3(P,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,160)
          GO TO 260
        ELSE
          INEQ(K)=INEW
        ENDIF
C       COUNT BY INEQUALITY TYPE UPDATED
        IF(INEW .EQ. 0)THEN
          NLC=NLC + 1
          PINEQ(K)='<'
        ELSEIF(INEW .EQ. 1)THEN
          NGC=NGC + 1
          PINEQ(K)='>'
        ELSE
          NEC=NEC + 1
          PINEQ(K)='='
        ENDIF
C       ADDED CONSTRAINT RHS INPUT BY USER
280     IF(IFLAG(5) .EQ. 0)THEN
          WRITE(1,'(8X,''RHS  = '',$)')
```

```
      ELSE
        WRITE(1,'(8X,''RHS'',10X,''= '',$)')
      ENDIF
      READ(5,'(10A1)')(P(L),L=1,10)
      CALL CHECK(P,INVAL,RNEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,160)
        GO TO 280
      ELSE
        B(K)=RNEW
      ENDIF
      PAUSE
290   WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),7X,''ARE CORRECTIONS NEEDED?  '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        CALL ICNRCH
        GO TO 300
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        GO TO 300
      ELSE
        WRITE(1,160)
        GO TO 290
      ENDIF
300   WRITE(1,110)CHAR(12)
310   WRITE(1,'(11(/),8X,''ADD ANOTHER CONSTRAINT? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 120
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        IFLAG(2)=0
        RETURN
      ELSE
        WRITE(1,160)
        GO TO 300
      ENDIF
      END
```

```
C*************************************
C  MODULE 1 UNIT15                                                  *
C                                                                   *
C  SUBROUTINE DISPLY                                                *
C  USE: FORMATS TABLEAU OUTPUT TO BOTH SCREEN AND PRINTER.  OUTPUTS *
C       EITHER THE CONSTRAINTS ONLY OR THE COMPLETE TABLEAU AS INPUT *
C       BY USER.  OPENS AND CLOSES OUTPUT UNIT DESIGNATED BY USER.  *
C  CALLED BY: PROGRAM DATAB                                         *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                   *
C  VARIABLES:                                                       *
C       USED: A(*,*),B(*),C(*),CN(*),IFLAG(5),INEW,INVAL,K,MM,OBJN, *
C             PINEQ(*),PN,V,VN(*)                                   *
C   MODIFIED: IFLAG(9),N,P(*),T                                     *
C*************************************
      SUBROUTINE DISPLY
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V,T
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(8(/),7X,''WOULD YOU LIKE DISPLAY ON:''//15X,''1. SCREEN'
     .'/20X,''OR''/15X,''2. PRINTER'')')
120   WRITE(1,'(/13X,''WHICH OPTION? '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,2,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
130     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 120
      ENDIF
C     PROPER FILE FOR SELECTED OUTPUT DEVICE OPENED
      IF(INEW .EQ. 1)THEN
        OPEN(2,FILE='CONSOLE:')
      ELSE
        OPEN(2,FILE='PRINTER:')
      ENDIF
      WRITE(1,110)CHAR(12)
      IF(IFLAG(9) .EQ. 1)THEN
C       ONLY CONSTRAINTS ARE DISPLAYED
        WRITE(2,220)PN
220     FORMAT(10X,A20/10X,'CURRENT CONSTRAINTS')
      ELSE
C       OBJ FUNCTION AND CONSTRAINTS DISPLAYED
        WRITE(2,230)PN,MM
230     FORMAT(10X,A20/7X,'CURRENT LP MODEL: ',A3,'IMIZE ',$)
        IF(IFLAG(5) .EQ. 1)THEN
          WRITE(2,240)OBJN
240       FORMAT(A10)
        ELSE
          WRITE(2,250)
```

```
250       FORMAT(' ')
        ENDIF
      ENDIF
C     NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED
      T=(V/5)+1
      DO 470 N=1,T
        IF(IFLAG(5) .EQ. 1)THEN
C         VARIABLE NAMES PRINTED AS COLUMN HEADERS
          WRITE(2,'(13X,$)')
          DO 270 J=(N*5)-4,N*5
            IF(J .GT. V)THEN
              GO TO 270
            ENDIF
            WRITE(2,260)VN(J)
260         FORMAT(4X,A6,3X,$)
270       CONTINUE
          WRITE(2,'('' '')')
        ENDIF
        WRITE(2,'(13X,$)')
        DO 290 J=(N*5)-4,N*5
          IF(J .GT. V)THEN
            GO TO 290
          ENDIF
          WRITE(2,280)J
280       FORMAT(4X,'X(',I2,')',4X,$)
290     CONTINUE
C       IF LAST 80 COLUMN DISPLAY, DISPLAY RHS
        IF(T .EQ. 1 .OR. N .EQ. T)THEN
          WRITE(2,300)
300       FORMAT(7X,'RHS')
        ELSE
          WRITE(2,'('' '')')
        ENDIF
        IF(IFLAG(9) .EQ. 0)THEN
          WRITE(2,'(''OBJ FUNCTION'',1X,$)')
          DO 320 J=(N*5)-4,N*5
            IF(J .GT. V)THEN
              GO TO 320
            ENDIF
            WRITE(2,310)C(J)
310         FORMAT(1X,1PE12.5,$)
320       CONTINUE
          IF(T .EQ.1 .OR. N .EQ. T)THEN
            WRITE(2,330)
330         FORMAT(1X,'=     Z')
          ELSE
            WRITE(2,'('' '')')
          ENDIF
          WRITE(2,'(''CONST  NAME'',2X,67(''*''))')
        ELSE
          WRITE(2,'(''CONST  NAME'')')
        ENDIF
```

```
C       CONSTRAINT NUMBER, NAME, BASIC VARIABLE, COEFFICIENTS,
C         INEQUALITY, AND RHS DISPLAYED
        DO 400 L=1,K
          IF(L .GT. K)THEN
            GO TO 400
          ENDIF
          WRITE(2,340)L
340       FORMAT('CN#',I2,$)
          IF(IFLAG(5) .EQ. 1)THEN
            WRITE(2,350)CN(L)
350           FORMAT(1X,A6,1X,$)
          ELSE
            WRITE(2,'(8X,$)')
          ENDIF
          DO 370 J=(N*5)-4,N*5
            IF(J .GT. V)THEN
              GO TO 370
            ENDIF
            WRITE(2,360)A(L,J)
360         FORMAT(1X,1PE12.5,$)
370       CONTINUE
          IF(T .EQ. 1 .OR. N .EQ. T)THEN
            WRITE(2,380)PINEQ(L),B(L)
380         FORMAT(1X,A1,1X,1PE12.5)
          ELSE
            WRITE(2,'('' '')')
          ENDIF
400     CONTINUE
C       SPACING APPROPRIATE FOR SELECTED OUTPUT DEVICE IMPLEMENTED
        IF(INEW .EQ. 1)THEN
          PAUSE
          WRITE(2,110)CHAR(12)
        ELSE
          WRITE(2,'(2(/))')
        ENDIF
470   CONTINUE
      IFLAG(9)=0
C     OUTPUT DEVICE FILE CLOSED
      CLOSE(2)
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT16                                                         *
C    UNIT $USES: UNIT17                                                    *
C                                                                          *
C  SUBROUTINE SAVE                                                         *
C  USE: SOLICITS VOLUME:FILENAME OF DISK FILE TO STORE NEW OR EDITED       *
C       MODEL.  ALSO SOLICITS INPUT WHICH IDENTIFIES INPUTTED FILE AS      *
C       A NEW FILE OR AN UPDATE OF FILE.  SAVES DATA IN DISK FILE          *
C       (FN) AND ALSO WRITES (FN) TO DISKFILE LP1:LPDATA FOR TRANSFER      *
C       TO MODULES 2 AND 3.                                                *
C  CALLED BY: PROGRAM DATAB                                                *
C  CALLS    : NONE                                                         *
C  VARIABLES:                                                              *
C       USED: A(*,*),B(*),C(*),CN(*),IFLAG(1),IFLAG(3),IFLAG(4),           *
C             IFLAG(5),IFLAG(6),IFLAG(7),IFLAG(8),IFLAG(9),IFLAG(10),      *
C             INEQ(*),K,MM,MXMN,NEC,NGC,NLC,OBJN,PINEQ(*),PN,V,VN(*)       *
C   MODIFIED: FN,IFLAG(2),P(*)                                             *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK IN UNIT17.CODE OVERLAY
      SUBROUTINE SAVE
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10,RES*6
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
C     USER INPUTS FILE NAME WHICH MODEL IS TO BE SAVED
      WRITE(1,'(//9X,''SAVE LP MODEL TO DISK''///2X,''ENTER THE DISK DRI
     .VE NUMBER AND FILE''/6X,''NAME WHICH YOU WANT PROBLEM''/10X,A20,/
     .14X,''SAVED UNDER.'')')PN
      WRITE(1,'(/8X,''ENTER EXACTLY AS FOLLOWS''/10X,''DISK DRIVE:FILENA
     .ME''//12X,''EG.   #4:FILENAM''//''THE DRIVE:FILENAME MUST BE 10 CH
     .ARACTERS''//16X,''OR LESS''//''IF THE ABOVE IS ENTERED INCORRECTLY
     .,''/7X,''YOUR MODEL WILL BE LOST!!'')')
      WRITE(1,'(/7X,''DISK:FILENAME = '',$)')
      READ(5,'(A10)')FN
120   WRITE(1,'(/7X,''ARE CORRECTIONS NEEDED? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 100
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,200)
        PAUSE
        GO TO 120
      ENDIF
      WRITE(1,110)CHAR(12)
C     USER PROMPTED TO INSERT DISK TO WHICH FILE IS TO BE SAVED
      WRITE(1,'(9(/),2X,''INSURE THE DISK TO CONTAIN THE FILE''//15X,A10
     .//13X,''IS AVAILABLE.'',6(/))')FN
      PAUSE
      WRITE(1,110)CHAR(12)
```

```
      WRITE(1,'(9(/),''HAS THIS DISK:FILENAME COMBINATION BEEN''/12X,''U
     .SED PREVIOUSLY?''//''(ARE YOU UPDATING A CURRENTLY EXISTING''/17X,
     .''FILE?)'')')
150   WRITE(1,'(/16X,''(Y/N) '',$)')
      READ(5,'(A1)')P(1)
C     PROPER STATUS OF FILE DETERMINED AND OPENED
      IF(ICHAR(P(1)) .EQ. 89)THEN
        OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        OPEN(3,FILE=FN,STATUS='NEW',FORM='UNFORMATTED')
      ELSE
        WRITE(1,200)
200     FORMAT(/5X,'INVALID ENTRY. PLEASE REENTER')
        GO TO 150
      ENDIF
C     MODEL WRITTEN TO DISK
      WRITE(3)PN,MXMN,MM,K,V,NEC,NGC,NLC
      DO 250 I=1,10
        WRITE(3)IFLAG(I)
250   CONTINUE
      DO 300 I=1,K
        WRITE(3)INEQ(I),PINEQ(I),B(I)
        DO 290 J=1,V
          WRITE(3)A(I,J)
290     CONTINUE
300   CONTINUE
      DO 350 J=1,V
        WRITE(3)C(J)
350   CONTINUE
      IF(IFLAG(5).EQ. 1)THEN
        DO 380 I=1,K
          WRITE(3)CN(I)
380     CONTINUE
        DO 400 J=1,V
          WRITE(3)VN(J)
400     CONTINUE
        WRITE(3)OBJN
      ENDIF
      IFLAG(2)=1
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/))')
      PAUSE
C     NAME OF MODEL LAST SAVED WRITTEN TO TRANSFER FILE
      OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
      WRITE(3)FN
      CLOSE(3,STATUS='KEEP')
      RETURN
      END
```

```
C ****************************************
C  MODULE 1 UNIT16                                                    *
C                                                                     *
C  SUBROUTINE INIT                                                    *
C  USE: INITIALIZES ALL VARIABLES TO ZERO EXCEPT CHARACTER VARIABLES. *
C  CALLED BY: PROGRAM DATAB                                           *
C  CALLS    : NONE                                                    *
C  VARIABLES:                                                         *
C       USED: NONE                                                    *
C   MODIFIED: A(*,*),B(*),C(*),IFLAG(1) THRU IFLAG(10),INEQ(*),K,NEC, *
C             NGC,NLC,V                                               *
C ****************************************
      SUBROUTINE INIT
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MYMN
      DO 200 I=1,20
        B(I)=0.
        INEQ(I)=0
        DO 100 J=1,60
          A(I,J)=0.
100     CONTINUE
200   CONTINUE
      DO 300 J=1,60
        C(J)=0.
300   CONTINUE
      DO 400 I=1,10
        IFLAG(I)=0
400   CONTINUE
      NEC=0
      NGC=0
      NLC=0
      RETURN
      END
```

```
C ***************************************
C  MODULE 1 UNIT16                                              *
C                                                               *
C  SUBROUTINE DATAD                                             *
C  USE: SOLICITS VOLUME:FILENAME FROM USER OF FILE TO BE READ FROM    *
C       DISK.  PROMPTS USER TO INSERT CORRECT DISK AND READS MODEL    *
C       REQUESTED FROM DISK INTO MEMORY FOR FUTURE EDITING OR         *
C       DISPLAY.                                                      *
C  CALLED BY: PROGRAM DATAB                                           *
C  CALLS    : NONE                                                    *
C  VARIABLES:                                                         *
C       USED: NONE                                                    *
C   MODIFIED: A(*,*),B(*),C(*),CN(*),FN,IFLAG(1) THRU IFLAG(10),      *
C             INEQ(*),K,MM,MXMN,NEC,NGC,NLC,OBJN,P(*),PINEQ(*),V,     *
C             VN(*)                                                   *
C ***************************************
      SUBROUTINE DATAD
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER V
      COMMON/C1/A(20,60),B(20),C(60),INEQ(20),IFLAG(10),NEC,NGC,NLC,K,V,
     .MXMN
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
C     USER INPUTS FILE NAME OF MODEL TO BE READ
      WRITE(1,'(5(/),8X,''READ LP MODEL FROM DISK''///''ENTER THE DISK D
     .RIVE NUMBER AND FILE''/''NAME WHICH HOLDS THE MODEL DESIRED.''//8X
     .,''ENTER EXACTLY AS FOLLOWS''/10X,''DISK DRIVE:FILENAME''//12X,''E
     .G.   #4:FILENAM'')')
      WRITE(1,'(/7X,''DISK:FILENAME = '',$)')
      READ(5,'(A10)')FN
      WRITE(1,110)CHAR(12)
C     USER PROMPTED TO INSERT DISK CONTAINING DESIGNATED FILE
      WRITE(1,'(9(/),5X,''INSURE THE DISK CONTAINING THE''//15X,A10//10X
     .,''MODEL IS AVAILABLE.'',7(/))')FN
      PAUSE
C     DESIGNATED FILE OPENED AND READ TO MEMORY
      OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
      READ(3)PN,MXMN,MM,K,V,NEC,NGC,NLC
      DO 180 I=1,10
        READ(3)IFLAG(I)
180   CONTINUE
      DO 250 I=1,K
        READ(3)INEQ(I),PINEQ(I),B(I)
        DO 240 J=1,V
          READ(3)A(I,J)
240     CONTINUE
250   CONTINUE
      DO 270 J=1,V
        READ(3)C(J)
270   CONTINUE
      IF(IFLAG(5) .EQ. 1)THEN
```

```
        DO 280 I=1,K
          READ(3)CN(I)
 280    CONTINUE
        DO 300 J=1,V
          READ(3)VN(J)
 300    CONTINUE
        READ(3)OBJN
      ENDIF
      IFLAG(2)=1
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/))')
      PAUSE
C     LAST READ FILE NAME IS WRITTEN TO TRANSFER FILE
      OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
      WRITE(3)FN
      CLOSE(3,STATUS='KEEP')
      RETURN
      END
```

```
C ****************************************
C  MODULE 1 UNIT 16                       *
C                                         *
C  SUBROUTINE HEADER                      *
C  USE: FORMATS AND DISPLAYS TITLE/AUTHOR PAGE.   *
C  CALLED BY: PROGRAM DATAB               *
C  CALLS    : NONE                        *
C  VARIABLES: NONE                        *
C ****************************************
      SUBROUTINE HEADER
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(9X,22(''*'')/9X,''*'',3X,''FORTRAN BASED'',4X,''*''
     ./9X,''*'',20X,''*''/9X,''* LINEAR PROGRAMMING *''/9X,''*'',20X,''*
     .''/9X,''*'',8X,''FOR'',9X,''*''/9X,''*'',20X,''*''/9X,''*'',3X,''M
     .ICROCOMPUTERS'',3X,''*''/9X,22(''*''))')
      WRITE(1,'(//19X,''BY''//9X,''THEODORE R. E. FRALEY''//18X,''AND''
     .//14X,''DALE A. KEM'')')
      RETURN
      END
```

```
C *************************************
C  MODULE 1 UNIT16                                              *
C                                                               *
C  SUBROUTINE MODUL(INEW)                                       *
C  USE: UPON INITIAL ENTRY INTO MODULE, SOLICITS INPUT AS TO WHICH   *
C       MODULE OF THE SOFTWARE PACKAGE IS DESIRED.  PROVIDES     *
C       INSTRUCTIONS AS TO WHAT PRIOR ACTIONS ARE REQUIRED TO ENTER  *
C       EACH MODULE AND THE COMMANDS REQUIRED TO GAIN ACCESS TO THESE *
C       MODULES.  IF USER SELECTS TO ENTER ANOTHER MODULE, MODULE 1  *
C       PROGRAM IS TERMINATED WITH INSTRUCTIONS ON SCREEN SHOWING THE *
C       COMMANDS REQUIRED TO ENTER SELECTED MODULE.  OTHERWISE, USER  *
C       ENTERS NEW MODEL OR EDITS MODEL WITH THIS MODULE.        *
C  CALLED BY: PROGRAM DATAR                                     *
C  CALLS    : SUBROUTINE CHECK2(P,N.M,INVAL,INEW)               *
C  VARIABLES:                                                   *
C       USED: INVAL                                             *
C   MODIFIED: FN,INEW,P(*)                                      *
C *************************************
      SUBROUTINE MODUL(INEW)
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      COMMON/C2/VN(60),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
C     DETERMINES IF MODULE DESIRED ALREADY SELECTED
      IF(INEW . NE. 0)THEN
        INEW=INEW+1
        GO TO 200
      ENDIF
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
C     DISPLAYS MODULE OPTIONS
      WRITE(1,'(/12X,''MODULE SELECTION''//''THE FOLLOWING OPTIONS ARE A
     .VAILABLE:''//''1. DATA BASE ENTRY (ENTER DATA BASE OR''/3X,''EDIT
     . CURRENT DATA BASE)''//''2. LP INSTRUCTIONAL MODULE''//''3. LP PRO
     .BLEM SOLVER MODULE'')')
      WRITE(1,'(/''4. LP SENSITIVITY ANALYSIS MODULE''//''(NOTE: OPTIONS
     . 2, OR 3 REQUIRE THAT A''/''DATA BASE BE CURRENTLY STORED ON DISK)
     .'')')
      WRITE(1,'(/,''(NOTE: OPTION 4 REQUIRES THAT A DATA''/''FILE HAVE B
     .EEN SAVED UPON LEAVING THE''/''OPTION 2 OR 3 MODULES ABOVE.)'')')
120   WRITE(1,130)
130   FORMAT(/13X,'WHICH OPTION? ',$)
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,4,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,140)
140     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 120
      ENDIF
      IF(INEW .EQ. 1)THEN
C       USER ELECTS TO ENTER MODEL
        RETURN
      ENDIF
200   WRITE(1,110)CHAR(12)
```

```
      IF(INEW .EQ. 4)THEN
C       USER ELECTS TO PERFORM SENSITIVITY ANALYSIS AND PROMPTED TO
C         INSERT DISK LP2 TO WRITE FILE NAME IN TRANSFER FILE
        WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/)
     .)')
        PAUSE
C       TRANSFER FILE OPENED
        OPEN(3,FILE='LP2:LPDATAW',STATUS='OLD',FORM='UNFORMATTED')
      ELSE
C       USER HAS SELECTED OTHER THAN SENSITIVITY ANALYSIS AND PROMPTED
C         TO INSERT DISK LP1
        WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/)
     .)')
        PAUSE
C       TRANSFER FILE OPENED
        OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
      ENDIF
      WRITE(1,110)CHAR(12)
      IF(INEW .EQ. 2)THEN
C       USER ELECTS EDUCATIONAL MODULE
        WRITE(1,'(8X,''LP INSTRUCTIONAL MODULE'')')
      ELSEIF(INEW .EQ. 3)THEN
C       USER ELECTS PROBLEM SOLVER MODULE
        WRITE(1,'(8X,''LP PROBLEM SOLVER MODULE'')')
      ELSE
        WRITE(1,'(5X,''LP SENSITIVITY ANALYSIS MODULE'')')
      ENDIF
C     MODEL FILE NAME IN TRANSFER FILE READ
      READ(3)FN
      CLOSE(3,STATUS='KEEP')
      WRITE(1,'(/,''TO USE THIS MODULE, A DATA BASE MUST''/''HAVE BEEN P
     .REVIOUSLY CREATED USING THE''/''DATA BASE ENTRY (MODULE 1) AND SAV
     .ED TO''/''DISK.'')')
      WRITE(1,'(/''THE DATA BASE WHICH IS CURRENTLY''/''IDENTIFIED AS TH
     .E PROBLEM TO BE STUDIED''/''IS:''/15X,A10)')FN
C     USER DETERMINES IF PROPER FILE NAME IN TRANSFER FILE
230   WRITE(1,'(//''IS THIS THE MODEL YOU WISH TO STUDY? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 300
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
C       USER ENTERS DESIRED FILE NAME TO BE ENTERED IN TRANSFER FILE
        WRITE(1,'(/''PLEASE ENTER THE DISK DRIVE NUMBER AND''/''FILENAME
     . OF THE FILE YOU WISH TO STUDY.''/''(INSURE THIS IS ENTERED EXACTL
     .Y AS IT'')')
        WRITE(1,'(''WAS SAVED PREVIOUSLY AND ALSO THAT THE''/''PROPER DI
     .SK IS IN THE PROPER DRIVE.)'')')
        WRITE(1,'(/4X,''MODEL TO BE STUDIED = '',$)')
        READ(5,'(A10)')FN
250     WRITE(1,'(/,7X,''ARE CORRECTIONS NEEDED? '',$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
```

```
        GO TO 200
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,140)
        GO TO 250
      ENDIF
      WRITE(1,110)CHAR(12)
      IF(INEW .EQ. 4)THEN
C       USER PROMPTED TO INSERT DISK LP2 AND FILE NAME IS WRITTEN
        WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7
     .(/))')
        PAUSE
        OPEN(3,FILE='LP2:LPDATAN',STATUS='OLD',FORM='UNFORMATTED')
      ELSE
C       MODEL FILE NAME WRITTEN TO TRANSFER FILE FOR OTHER THAN
C         SENSITIVITY ANALYSIS
        WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7
     .(/))')
        PAUSE
        OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
      ENDIF
      WRITE(3)FN
      CLOSE(3,STATUS='KEEP')
    ELSE
      WRITE(1,140)
      GO TO 230
    ENDIF
    WRITE(1,110)CHAR(12)
    WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/))')
    PAUSE
300 WRITE(1,110)CHAR(12)
C   INSTRUCTIONS TO ENTER OTHER MODULES DISPLAYED
    IF(INEW .EQ. 2)THEN
      WRITE(1,'(8(/),1X,''TO ENTER THE LP INSTRUCTIONAL MODULE:''//17X
     .,''TYPE''//19X,''X''/17X,''LP1:ED'',3(/))')
      STOP
    ELSEIF(INEW .EQ. 3)THEN
      WRITE(1,'(8(/),1X,''TO ENTER THE LP PROBLEM SOLVER MODULE:''//17
     .X,''TYPE''//19X,''X''/16X,''LP2:TAB'',3(/))')
      STOP
    ELSE
      WRITE(1,'(8(/),2X,''TO ENTER THE LP SENSITIVITY ANALYSIS''/16X,
     .''MODULE:''//17X,''TYPE''/19X,''X''/16X,''LP2:SEN'',3(/))')
      STOP
    ENDIF
    RETURN
    END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT 16                                                            *
C                                                                              *
C  SUBROUTINE DBHED                                                            *
C  USE: DISPLAYS TITLE PAGE OF MODULE 1, DATA BASE ENTRY MODULE.               *
C  CALLED BY: PROGRAM DATAB                                                    *
C  CALLS    : NONE                                                             *
C  VARIABLES: NONE                                                             *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE DBHED
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(4(/),9X,22(''*'')/9X,''*'',20X,''*''/9X,''*'',8X,''DATA'
     .',8X,''*''/9X,''*'',20X,''*''/9X,''*'',8X,''BASE'',8X,''*''/9X,''*
     .'',20X,''*''/9X,''*'',8X,''ENTRY'',7X,''*''/9X,''*'',20X,''*'')')
      WRITE(1,'(9X,''*'',7X,''MODULE'',7X,''*'',2(/9X,''*'',20X,''*''),/
     .9X,''*'',6X,''MODULE 1'',6X,''*''/9X,''*'',20X,''*''/9X,22(''*''),
     .3(/))')
      PAUSE
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT16                                                            *
C                                                                             *
C  SUBROUTINE INTRO                                                           *
C  USE: PROVIDES OVERVIEW OF LP PACKAGE AND PROVIDES BRIEF REFERENCE          *
C       MARKERS TO DOCUMENTATION.                                             *
C  CALLED BY: PROGRAM DATAB                                                   *
C  CALLS    : NONE                                                            *
C  VARIABLES: NONE                                                            *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE INTRO
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(2X,''LINEAR PROGRAMMING SOFTWARE PACKAGE''//''THIS PACKA
     .GE IS DESIGNED TO ALLOW''/''STUDENTS TO IMPROVE THEIR UNDERSTANDIN
     .G''/''OF THE SIMPLEX ALGORITHM AND ALSO''/''PROVIDE THE MANAGERS A
     .ND ANALYSTS WITH A''/''PROBLEM SOLVING TOOL.'')')
      WRITE(1,'(/,''THE PACKAGE CONSISTS OF FOUR DISTINCT''/''PROGRAMS (
     .ANNOTATED AS MODULES) WHOSE''/''FUNCTIONS ARE AS FOLLOWS:''//5X,''
     .MODULE 1: DATA BASE ENTRY''/5X,''MODULE 2: LP INSTRUCTION''/5X,''M
     .ODULE 3: LP PROBLEM SOLVER''/5X,''MODULE 4: SENSITIVITY ANALYSIS''
     .)')
      WRITE(1,'(/,''ALL LP PROBLEMS MUST BE ENTERED INTO A''/''DATABASE
     . USING MODULE 1.  MODULES 2 OR''/''3 MAY BE USED TO DETERMINE A SO
     .LUTION TO''/''A PROBLEM AND THIS MUST OCCUR PRIOR TO''/''ENTERING
     . MODULE 4.'')')
      PAUSE
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(/,''INSTRUCTIONS ON HOW TO ENTER EACH MODULE''/''WILL BE
     .PRESENTED WHEN APPROPRIATE.''//''ANSWERS TO SPECIFIC QUESTIONS CON
     .CERNING''/''ANY MODULE WILL BE FOUND IN THE USERS''/''GUIDE (APPEN
     .DIX A) OF THE THESIS''/''DOCUMENTATION.'')')
      WRITE(1,'(/,''ALL RESPONSES REQUIRE A [RETURN] TO NOTE''/''THE COM
     .PLETION OF INPUT.''//''ALSO, ALL YES/NO INPUTS MAY BE ENTERED''/''
     .BY [Y] OR [N], RESPECTFULLY.'')')
      PAUSE
      RETURN
      END
```

```
C ****************************************
C  MODULE 1 UNIT16                        *
C                                         *
C  SUBROUTINE DBE                         *
C  USE: DISPLAYS MENU OF DATA BASE ENTRY OPTIONS (NEW MODEL, READ  *
C       EXISTING MODEL,QUIT,INTRODUCTORY REMARKS).                 *
C  CALLED BY: PROGRAM DATAB               *
C  CALLS    : NONE                        *
C  VARIABLES: NONE                        *
C ****************************************
      SUBROUTINE DBE
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(12X,''DATA BASE ENTRY''//6X,''TO ENTER LP MODEL DATA BAS
     .E''/4X,''YOU HAVE THE FOLLOWING OPTIONS:''///''1. CREATE MODEL INT
     .ERACTIVELY:SUBSCRIPTS''/3X,''(VARIABLES ANNOTATED BY SUBSCRIPTS,''
     ./3X,''CONSTRAINTS ANNOTATED BY NUMBER ONLY)'')')
      WRITE(1,'(/''2. CREATE MODEL INTERACTIVELY:NAMED''/3X,''VARIABLES
     . AND CONSTRAINTS ARE''/3X,''ASSIGNED NAMES)''//''3. READ FROM DISK
     .''/3X,''(PREVIOUSLY CREATED BASE)'')')
      WRITE(1,'(/''4. DISPLAY INTRODUCTORY REMARKS''//''5. QUIT PROGRAM'
     .')')
      RETURN
      END
```

```
C ****************************************
C  MODULE 1 UNIT16                        *
C                                         *
C  SUBROUTINE DBM                         *
C  USE: DISPLAYS MENU OF DATA BASE MANAGEMENT OPTIONS (DISPLAY,EDIT,  *
C       SAVE,ENTER NEW MODEL,SOLVE,QUIT)  *
C  CALLED BY: PROGRAM DATAB               *
C  CALLS    : NONE                        *
C  VARIABLES: NONE                        *
C ****************************************
      SUBROUTINE DBM
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(10X,''DATA BASE MANAGEMENT''//''THE FOLLOWING OPTIONS AR
     .E AVAILABLE:''/''1. DISPLAY CURRENT LP MODEL''/3X,''(SCREEN OR PRI
     .NTER)''//''2. EDIT CURRENT LP MODEL''/3X,''(CHANGE ANY PARAMETER)
     .''//''3. SAVE CURRENT MODEL TO DISK FILE''/3X,''(MAY THEN EDIT TO
     . ANOTHER MODEL)'')')
      WRITE(1,'(/''4. ENTER NEW MODEL''/3X,''(CURRENT MODEL LOST IF NOT
     . ON DISK)''//''5. SOLVE PROBLEM''/3X,''(INCLUDES EDUCATIONAL, PROB
     .LEM SOLVER,''/4X,''AND SENSITIVITY ANALYSIS)''//''6. QUIT:UNSAVED
     . FILES DESTROYED!'')')
      RETURN
      END
```

```
C ****************************************
C  MODULE 1 UNIT16                                               *
C                                                                *
C  SUBROUTINE DEM                                                *
C  USE: DISPLAYS MENU OF OPTIONS REGARDING WHICH MODULE IS DESIRED TO *
C       BE EXECUTED NEXT.  ALSO OPTIONS OF RETURNING TO DATA BASE    *
C       MANAGEMENT OR TO QUIT THE LP PACKAGE ARE DISPLAYED.          *
C  CALLED BY: PROGRAM DATAB                                      *
C  CALLS    : NONE                                               *
C  VARIABLES: NONE                                               *
C ****************************************
      SUBROUTINE DEM
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(10X,''EXECUTION MANAGEMENT''//''THE FOLLOWING OPTIONS AR
     .E AVAILABLE:''//''1. LP INSTRUCTIONAL MODULE''/3X,''(EACH TABLEAU
     . MAY BE DISPLAYED)''//''2. PROBLEM SOLVER MODULE''/3X,''(NO USER I
     .NTERACTION)''//''3. SENSITIVITY ANALYSIS MODULE '')')
      WRITE(1,'(/''4. RETURN TO DATA BASE MANAGEMENT MENU''//''5. QUIT:U
     .NSAVED FILES WILL BE LOST!'')')
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 1 UNIT17                                                        *
C    UNIT $USES: NONE                                                     *
C                                                                         *
C  SUBROUTINE CHECK(E,INVAL,RENEW)                                        *
C  USE: VERIFIES USER INPUT OF REAL NUMBERS WHICH HAVE BEEN READ INTO     *
C       SINGLE ELEMENT CHARACTER STRINGS.  CHECKS ELEMENT BY ELEMENT      *
C       THAT EACH CHARACTER STRING IS A NUMERIC, A VALID OPERATOR, OR     *
C       DECIMAL POINT.  IF ALL ARE VALID, TRANSFORMS CHARACTER            *
C       STRING REPRESENTATION INTO NUMERIC REAL.  IF INVALID              *
C       CHARACTER IS FOUND, FLAG SET WHICH CALLING ROUTINE CHECKS TO      *
C       SIGNAL USER TO REINPUT NUMBER.                                    *
C  CALLED BY: SUBROUTINE ADCON                                            *
C             SUBROUTINE ADVAR                                            *
C             SUBROUTINE CNVA                                             *
C             SUBROUTINE DATAN                                            *
C             SUBROUTINE DATAS                                            *
C             SUBROUTINE ICNRCH                                           *
C             SUBROUTINE OBJCH                                            *
C  CALLS    : NONE                                                        *
C  VARIABLES:                                                             *
C       USED: ALLOW(*),E(*),M                                             *
C   MODIFIED: DECIMA,INVAL,NEGAT,RNEW                                     *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE CHECK(E,INVAL,RNEW)
      CHARACTER ALLOW*1,E*1
      DIMENSION E(10),ALLOW(14)
      REAL M
      INTEGER DECIMA
      DATA ALLOW/'1','2','3','4','5','6','7','8','9','0','+','-','.',
     .' '/
      RNEW=0.0
      M=.1
      INVAL=0
      DECIMA=0
      NEGAT=0
      DO 400 I=1,10
C       CHECKS FIRST FOR BLANK CHARACTERS
        IF (E(I) .EQ. ALLOW(14)) THEN
          GO TO 400
        ENDIF
C       CHECKS EACH CHARACTER TO INSURE ACCEPTABILITY
        DO 200 J=1,13
          IF (E(I) .EQ. ALLOW(J)) THEN
            IF (DECIMA .EQ. 1) THEN
              GO TO 150
            ELSEIF (E(I) .EQ. '-') THEN
              NEGAT=1
              GO TO 400
            ELSEIF(E(I) .EQ. '.') THEN
              DECIMA=1
              GO TO 400
```

```
          ELSE
            RNEW=10*RNEW + (ICHAR(E(I)) -48)
            GO TO 400
          ENDIF
150       RNEW=RNEW +(ICHAR(E(I))-48)*M
          M=M*.1
          GO TO 400
        ELSEIF (J .EQ. 13) THEN
          INVAL=1
          RNEW=0.0
          RETURN
        ENDIF
200     CONTINUE
400   CONTINUE
      IF (NEGAT .EQ. 1) THEN
        RNEW= -1. * RNEW
      ENDIF
      RETURN
      END
```

```
C ****************************************
C  MODULE 1 UNIT17                                                   *
C                                                                    *
C  SUBROUTINE CHECK2(E,D.HVAL,INVAL,INEW)                            *
C  USE: VERIFIES USER INPUT OF INTEGER NUMBERS WHICH HAVE BEEN READ  *
C       INTO SINGLE ELEMENT CHARACTER STRINGS.  CHECKS ELEMENT BY    *
C       ELEMENT THAT EACH CHARACTER STRING IS NUMERIC OR BLANK.  IF  *
C       ALL ARE VALID, TRANSFORMS CHARACTER STRING REPRESENTATION    *
C       INTO NUMERIC INTEGER.  IF INVALID CHARACTER FOUND, FLAG SET  *
C       WHICH CALLING ROUTINE CHECKS TO SIGNAL USER TO REINPUT       *
C       NUMBER.                                                      *
C  CALLED BY: PROGRAM DATAB                                          *
C             SUBROUTINE ADVAR                                       *
C             SUBROUTINE CNVA                                        *
C             SUBROUTINE DELCON                                      *
C             SUBROUTINE DELVAR                                      *
C             SUBROUTINE DISPLY                                      *
C             SUBROUTINE GENIF                                       *
C             SUBROUTINE ICNRCH                                      *
C             SUBROUTINE MODUL(INEW)                                 *
C             SUBROUTINE OBJCH                                       *
C             SUBROUTINE VNCH                                        *
C  CALLS    : NONE                                                   *
C  VARIABLES:                                                        *
C       USED: ALLOW(*),D,E(*),HVAL                                   *
C   MODIFIED: INEW,INVAL                                             *
C ****************************************
      SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
      CHARACTER ALLOW*1,E*1
      DIMENSION E(10),ALLOW(11)
      INTEGER D,HVAL
      DATA ALLOW/'1','2','3','4','5','6','7','8','9','0',' '/
      INEW=0
      INVAL=0
      DO 300 I=1,D
        DO 200 J=1,10
C         CHECKS FIRST FOR BLANK CHARACTERS
          IF(E(I) .EQ. ALLOW(11))THEN
            GO TO 300
          ELSEIF(E(I) .EQ. ALLOW(J))THEN
            INEW=INEW*10 + (ICHAR(E(I))-48)
            GO TO 300
          ELSEIF(J .EQ. 10)THEN
            INVAL=1
            INEW=0
            RETURN
          ENDIF
200     CONTINUE
300   CONTINUE
      IF(INEW .EQ. 0 .OR. INEW .GT. HVAL)THEN
        INVAL=1
        INEW=0
```

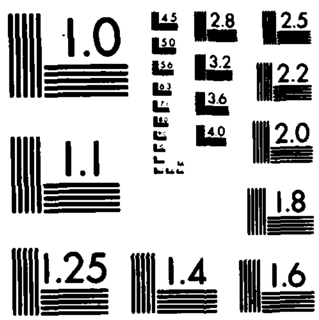
```
  RETURN
ENDIF
RETURN
END
```

```
C ****************************************
C  MODULE 1 UNIT17                                                  *
C                                                                   *
C  SUBROUTINE CHECK3(E,INVAL,INEW)                                  *
C  USE: VERIFIES USER INPUT OF INEQUALITY AND EQUALITY SYMBOLS.     *
C       IF ELEMENT IS INVALID, SETS FLAG WHICH CALLING ROUTINE CHECKS *
C       TO SIGNAL USER TO REINPUT SYMBOL.                           *
C  CALLED BY: SUBROUTINE ADCON                                      *
C             SUBROUTINE DATAN                                      *
C             SUBROUTINE DATAS                                      *
C             SUBROUTINE ICNRCH                                     *
C  CALLS    : NONE                                                  *
C  VARIABLES:                                                       *
C       USED: ALLOW(*),E(*)                                         *
C   MODIFIED: INEW,INVAL                                            *
C ****************************************
      SUBROUTINE CHECK3(E,INVAL,INEW)
      CHARACTER ALLOW*1,E*1
      DIMENSION E(10),ALLOW(3)
      DATA ALLOW/'<','=','>'/
      INEW=0
      INVAL=0
      IF(ICHAR(E(1)) .EQ. 60)THEN
        INEW=0
      ELSEIF(ICHAR(E(1)) .EQ. 62)THEN
        INEW=1
      ELSEIF(ICHAR(E(1)) .EQ. 61)THEN
        INEW=2
      ELSE
        INVAL=1
        INEW=0
      ENDIF
      RETURN
      END
```

```
C****************************************************************
C  MODULE 2 UNIT20                                              *
C    UNITS $USES: UNIT21 THRU UNIT27                            *
C                                                               *
C  PROGRAM EDUC                                                 *
C  USE: MAIN PROGRAM OF MODULE 2, LINEAR PROGRAMMING PACKAGE.   *
C       PURPOSE OF MODULE IS TO PROVIDE A TUTORIAL WHICH PROVIDES *
C       GUIDANCE IN THE SEQUENCE OF STEPS OF TRANSFORMING A GIVEN *
C       PROBLEM INTO THE TABULAR FORM WHICH THE SIMPLEX METHOD IS *
C       APPLIED.  ALSO THE MATHEMATICAL OPERATIONS ARE EMPHASIZED BY *
C       SOLICITING RESPONSES FROM THE USER UPON THE VIEWING OF EITHER *
C       NUMERICAL MANIPULATION OPTIONS OR OBJECTIVE SELECTION OPTIONS.*
C       USER IS GIVEN IMMEDIATE FEEDBACK ON CORRECTNESS OF OPTION *
C       SELECTION WITH A BRIEF INSTRUCTIONAL NOTE FOLLOWING INCORRECT *
C       RESPONSES.  MODULE 2 CONSISTS OF 6 SEPARATELY COMPILED UNITS *
C       (UNIT 20 THRU UNIT 27) WITH ALL UNITS EXCEPT UNIT20 BEING *
C       OVERLAY UNITS.                                          *
C       PROGRAM EDUC ACTS AS A MEMORY RELEASE LOCATION.  WHENEVER THE *
C       PROGRAM CONTROL RETURNS TO THIS UNIT, ALL OVERLAY UNITS ARE *
C       RELEASED FROM MEMORY PRIOR TO NEW UNITS BEING SUMMONED. *
C  CALLED BY: NONE                                              *
C  CALLS    : SUBROUTINE ASKQ(ASK)                              *
C             SUBROUTINE BIGM                                   *
C             SUBROUTINE CNMDU                                  *
C             SUBROUTINE HEADER                                 *
C             SUBROUTINE INDEX                                  *
C             SUBROUTINE INTRO                                  *
C             SUBROUTINE MODIFA                                 *
C             SUBROUTINE OBMDU                                  *
C             SUBROUTINE OPT                                    *
C             SUBROUTINE OPTION                                 *
C             SUBROUTINE PIVOT                                  *
C             SUBROUTINE QUESTN                                 *
C             SUBROUTINE READY                                  *
C             SUBROUTINE TDISPL                                 *
C  VARIABLES:                                                   *
C       USED: ASK,BIGM,IBTAB,IFLAG(1),IFLAG(7),IFLAG(10),ITAB,KFA, *
C             MOD,NEC,NGC,OPTS,OUTP,PN                          *
C   MODIFIED: BASIC,C(*),DUAL,IFLAG(2),P(*),PES                 *
C****************************************************************
$USES UCHECK2 IN UNIT27.CODE OVERLAY
$USES UTDISPL IN UNIT26.CODE OVERLAY
$USES UHEADER IN UNIT25.CODE OVERLAY
$USES UPIVOT IN UNIT24.CODE OVERLAY
$USES UOPT IN UNIT23.CODE OVERLAY
$USES UREADY IN UNIT22.CODE OVERLAY
$USES UOBMDU IN UNIT21.CODE OVERLAY
      PROGRAM EDUC
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT,ASK
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
```

```
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      OPEN(1,FILE='CONSOLE:')
      OPEN(5,FILE='CONSOLE:')
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      CALL HEADER
C     ROUTINE WHICH ALLOWS USER TO CHANGE DEFAULTS CALLED
130   CALL OPTION
C     ROUTINE WHICH INITIALIZES VARIABLES AND READS MODEL CALLED
      CALL INTRD
C     DETERMINES WHETHER USER OR ALGORITHM TO PLACE IN TABULAR FORM
      IF(MOD .EQ. 1)THEN
C       USER TO PLACE IN TABULAR FORM
        CALL OBMDU
        CALL CNMDU
      ELSE
C       ALGORITHM TO PLACE IN TABULAR FORM
        CALL MODIFA
      ENDIF
C     ROUTINE WHICH ADDS SLACK, SURPLUS, AND ARTIFICAL VARIABLES CALLED
      CALL INDEX
      IF(MOD .EQ. 1)THEN
C       ROUTINE WHICH QUESTIONS USER ON TABLEAU FORM CALLED
        CALL READY
      ELSE
        DO 200 J=KFA,VT
          C(J)=-BM
200     CONTINUE
      ENDIF
C     CHECKS IF ARTIFICAL VARIABLES HAVE BEEN ADDED
      IF((NEC+NGC) .NE. 0)THEN
        CALL BIGM
      ENDIF
      IF(ITAB .EQ. 2)THEN
        BASIC=BASIC+1
        GO TO 300
      ENDIF
C     IF USER HAS SELECTED SCREEN OUTPUT AND ALSO TO SELECT PIVOT
C       ELEMENTS, ROUTINE WARNS USER TO STUDY TABLEAUS CAREFULLY
      IF(PES .NE. 3 .AND. OUTP .EQ. 1)THEN
        CALL QUESTN
      ENDIF
      WRITE(1,110)CHAR(12)
      IF(PES .EQ. 3)THEN
        IF((NEC+NGC) .NE. 0)THEN
          WRITE(1,220)
220       FORMAT(11(/),8X,'INITIAL BASIC SOLUTION',7(/))
        ELSE
          WRITE(1,240)
```

AD-A124 804 FORTRAN BASED LINEAR PROGRAMMING FOR MICROCOMPUTERS(U) 4/5
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING T R FRALEY ET AL. DEC 82
UNCLASSIFIED AFIT/GOR/OS/82D-4 F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
240     FORMAT(11(/),5X,'INITIAL BASIC FEASIBLE SOLUTION',7(/))
       ENDIF
       PAUSE
      ENDIF
C     USER SELECTED OUTPUT DEVICE FILE OPENED
      IF(OUTP .EQ. 1)THEN
       OPEN(2,FILE='CONSOLE:')
       WRITE(2,110)CHAR(12)
      ELSE
       OPEN(2,FILE='PRINTER:')
      ENDIF
C     TABLEAU HEADER PRINTED IF USER HAS ALLOWED ALGORITHM TO SOLVE
C       WITHOUT USER INTERFACE
      IF(PES .EQ. 3)THEN
       IF((NEC+NGC) .NE. 0)THEN
        WRITE(2,'(10X,A20/10X,''INITIAL BASIC SOLUTION''/)')PN
       ELSE
        WRITE(2,'(10X,A20/4X,''INITIAL BASIC FEASIBLE SOLUTION''/)')PN
       ENDIF
      ELSE
       I=1
       WRITE(2,290)PN,I
      ENDIF
C     BASIC SOLUTION COUNTER INCREMENTED
280   BASIC=BASIC+1
      IF(BASIC .NE. 1)THEN
       WRITE(2,290)PN,BASIC
290    FORMAT(10X,A20/10X,'BASIC SOLUTION #',I2,/)PN,BASIC
      ENDIF
C     FLAG DENOTES TABLEAU WITH BASIC VARIABLES ANNOTATED BE DISPLAYED
      IFLAG(9)=2
      CALL TDISPL
C     ROUTINE CALLED WHICH DETERMINES EXISTENCE OF OPTIMALITY
300   CALL OPT
      IF(OPTS .EQ. 1 .OR. IFLAG(7) .EQ. 1 .OR. IFLAG(1) .NE. 0)THEN
C       TABLEAU IS EITHER OPTIMAL, UNBOUNDED, OR INFEASIBLE
320    WRITE(1,110)CHAR(12)
       WRITE(1,'(10(/),''WOULD YOU LIKE TO PERFORM FURTHER PIVOTS''/11X
     .,''ON THIS TABLEAU? '',$)')
       READ(5,'(A1)')P(1)
       IF(ICHAR(P(1)) .EQ. 78)THEN
C         ROUTINE ALLOWS EXIT FROM MODULE
        CALL ASKQ(ASK)
        IF(ASK .EQ. 1)THEN
         GO TO 130
        ENDIF
       ELSEIF(ICHAR(P(1)) .EQ. 89)THEN
        IF(DUAL .EQ. 1)THEN
         PES=2
         DUAL=2
         WRITE(1,110)CHAR(12)
         WRITE(1,'(11(/),''THE OPTION OF PERFORMING DUAL PIVOTS HAS''
```

```
     ./6X,''BEEN ACTIVATED AT THIS TIME.'',9(/))')
         ENDIF
       ELSE
         WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER''//)')
         PAUSE
         GO TO 320
       ENDIF
     ENDIF
     CALL PIVOT
     IF(IFLAG(10) .EQ. 5)THEN
C       FLAG INDICATES FURTHER PIVOTS NOT DESIRED OR ALLOWED
       CALL ASKQ(ASK)
       IF(ASK .EQ. 1)THEN
         GO TO 130
       ENDIF
     ENDIF
     WRITE(1,110)CHAR(12)
C     DETERMINES IF INTERMEDIATE TABLEAU TO BE DISPLAYED
     IF(IBTAB .NE. 0)THEN
       IF(((FLOAT(BASIC-1))/(FLOAT(IBTAB))) .EQ. FLOAT((BASIC-1)/IBTAB)
     .)THEN
         IF(OUTP .EQ. 1)THEN
           OPEN(2,FILE='CONSOLE:')
         ELSE
           OPEN(2,FILE='PRINTER:')
         ENDIF
         GO TO 280
       ELSE
         BASIC=BASIC+1
         GO TO 300
       ENDIF
     ELSE
       BASIC=BASIC+1
       GO TO 300
     ENDIF
     STOP
     END
```

```
C ****************************************
C  MODULE 2 UNIT21                                                  *
C    UNIT $USES: UNIT27                                             *
C                                                                   *
C  SUBROUTINE OBMDU                                                 *
C  USE: DISPLAYS OBJECTIVE FUNCTION AS INPUT IN MODULE 1, SOLICITS  *
C       RESPONSE TO MENU OF OPTIONS OF OPERATIONS WHICH MAY BE      *
C       APPLIED TO PLACE OBJECTIVE FUNCTION IN MAXIMIZATION AND Z-X=0 *
C       FORM.  PROVIDES IMMEDIATE FEEDBACK AND DISPLAYS OBJECTIVE   *
C       FUNCTION AFTER PROPER MODIFICATION.  ONLY UTILIZED IF USER  *
C       ELECTS TO PERFORM OBJECTIVE FUNCTION MODIFICATION.          *
C  CALLED BY: PROGRAM EDUC                                          *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                   *
C  VARIABLES:                                                       *
C       USED: IFLAG(5),INEW,INVAL,MM,MXMN,V,VN(*)                   *
C   MODIFIED: C(*),P(*),T                                           *
C ****************************************
$USES UCHECK2 IN UNIT27.CODE OVERLAY
      SUBROUTINE OBMDU
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT,T
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
C     OBJECTIVE FUNCTION DISPLAYED AS INPUT BY USER
      WRITE(1,'(4X,''OBJECTIVE FUNCTION MODIFICATION'',8(/),''THE OBJECT
     .IVE FUNCTION, AS ENTERED, WILL''/''BE DISPLAYED NEXT.  AFTER THE D
     .ISPLAY,''/''YOU WILL BE ASKED TO SELECT THE OPTION'')')
      WRITE(1,'(''WHICH WILL TRANSFORM THE OBJECTIVE''/''FUNCTION INTO T
     .HE PROPER TABLEAU FORM''/''FOR THE SIMPLEX ALGORITHM.'',7(/))')
      PAUSE
      WRITE(1,110)CHAR(12)
      WRITE(1,120)
120   FORMAT(4X,'OBJECTIVE FUNCTION MODIFICATION'/7X,'PRESENT OBJECTIVE
     .FUNCTION'/)
C     NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED
      T=((V-1)/5)+1
      DO 300 N=1,T
        IF(IFLAG(5) .EQ. 1)THEN
C         VARIABLE NAMES PRINTED AS COLUMN HEADERS
          WRITE(1,'(10X,$)')
          DO 180 J=(N*5)-4,N*5
            IF(J .GT. V)THEN
              GO TO 180
            ENDIF
            WRITE(1,130)VN(J)
130         FORMAT(7X,A6,1X,$)
```

```
180       CONTINUE
          WRITE(1,'('' '')')
        ENDIF
        WRITE(1,'(10X,$)')
        DO 250 J=(N*5)-4,N*5
          IF(J .GT. V)THEN
            GO TO 250
          ENDIF
          WRITE(1,200)J
200       FORMAT(7X,'X(',I2,')',2X,$)
250     CONTINUE
        WRITE(1,'('' '')')
        IF(N .EQ. 1)THEN
          WRITE(1,'(2X,A3,1X,''Z  ='',$)')MM
        ELSE
          WRITE(1,'(10X,$)')
        ENDIF
C       OBJECTIVE FUNCTION COEFFICIENTS DISPLAYED
        DO 280 J=(N*5)-4,N*5
          IF(J .GT. V)THEN
            GO TO 280
          ENDIF
          IF(FMT .EQ. 0)THEN
            WRITE(1,'(1X,''+'',1X,1PE11.4,$)')C(J)
          ELSE
            WRITE(1,'(1X,''+'',1X,F11.4,$)')C(J)
          ENDIF
280     CONTINUE
        WRITE(1,'('' '',//)')
300   CONTINUE
      PAUSE
      WRITE(1,110)CHAR(12)
      WRITE(1,320)
C     OPTIONS DISPLAYED WHICH USER SELECTS THE ONE WHICH PROPERLY
C       MODIFIES OBJECTIVE FUNCTION
320   FORMAT(4X,'OBJECTIVE FUNCTION MODIFICATION')
      WRITE(1,'(//,''TO PLACE THE OBJECTIVE FUNCTION IN THE''/''PROPER F
     .ORMAT FOR THE SIMPLEX ALGORITHM''/''WHICH OF THE FOLLOWING SHOULD
     . BE DONE?''/)')
      WRITE(1,'(''1. ADD -C(J) TO BOTH SIDES OF EQUATION.''//''2. MULTIP
     .LY EQUATION BY -1 AND THEN ADD''/3X,''-C(J) TO BOTH SIDES OF EQUAT
     .ION.''//''3. NO CHANGES ARE NECESSARY.'')')
330   WRITE(1,340)
340   FORMAT(//7X,'WHICH OPTION IS CORRECT? ',$)
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,3,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,360)
360     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 330
      ENDIF
C     OBJ FUNCTION PROPERLY MODIFIED AND USER RECEIVES FEEDBACK
```

```
      IF(MXMN .EQ. 1)THEN
        DO 370 J=1,V
          C(J)=-C(J)
370     CONTINUE
        IF(INEW .EQ. 1)THEN
          WRITE(1,380)INEW
380       FORMAT(/10X,'OPTION #',I1,' IS CORRECT.'//)
          PAUSE
          GO TO 500
        ELSE
          WRITE(1,400)INEW
400       FORMAT(/9X,'OPTION #',I1,' IS INCORRECT.'//4X,'THE PROPER RESP
     .ONSE WAS OPTION #1.'//)
          PAUSE
          GO TO 500
        ENDIF
      ELSEIF(MXMN .EQ. 2)THEN
        IF(INEW .EQ. 2)THEN
          WRITE(1,380)INEW
          PAUSE
          GO TO 500
        ELSE
          WRITE(1,420)INEW
420       FORMAT(/9X,'OPTION #',I1,' IS INCORRECT.'//4X,'THE PROPER RESP
     .ONSE WAS OPTION #2.'//)
          PAUSE
        ENDIF
        MM='MAX'
      ENDIF
500   WRITE(1,110)CHAR(12)
C     PROPERLY MODIFIED OBJ FUNCTION DISPLAYED
      WRITE(1,'(2X,''AFTER THE PROPER MODIFICATION, THE''/6X,''OBJECTIVE
     . FUNCTION FORM IS:''/)')
      T=(V/5)+1
      DO 700 N=1,T
        IF(IFLAG(5) .EQ. 1)THEN
          WRITE(1,'(10X,$)')
          DO 550 J=(N*5)-4,N*5
            IF(J .GT. V)THEN
              GO TO 550
            ENDIF
            WRITE(1,520)VN(J)
520         FORMAT(7X,A6,1X,$)
550       CONTINUE
          WRITE(1,'('' '')')
        ENDIF
        WRITE(1,'(10X,$)')
        DO 600 J=(N*5)-4,N*5
          IF(J .GT. V)THEN
            GO TO 600
          ENDIF
          WRITE(1,570)J
```

```
570       FORMAT(7X,'X(',I2,')',2X,$)
600     CONTINUE
        WRITE(1,'('' '')')
        IF(N .EQ. 1)THEN
          IF(MXMN .EQ. 1)THEN
            WRITE(1,'(2X,''MAX  Z  '',$)')
          ELSE
            WRITE(1,'(2X,''MAX (-Z)'',$)')
          ENDIF
        ELSE
          WRITE(1,'(10X,$)')
        ENDIF
        DO 650 J=(N*5)-4,N*5
          IF(J .GT. V)THEN
            GO TO 650
          ENDIF
          IF(FMT .EQ. 0)THEN
            WRITE(1,'(1X,''+'',1X,1PE11.4,$)')C(J)
          ELSE
            WRITE(1,'(1X,''+'',1X,F11.4,$)')C(J)
          ENDIF
650     CONTINUE
        IF(N .EQ. T)THEN
          WRITE(1,'('' =0'')')
        ELSE
          WRITE(1,'('' '')')
        ENDIF
700   CONTINUE
      PAUSE
      RETURN
      END
```

```
C ****************************************
C  MODULE 2 UNIT21                                              *
C                                                               *
C  SUBROUTINE OPTION                                            *
C  USE: DISPLAYS DEFAULT OPTION VALUES AND SOLICITS RESPONSE TO *
C       CHANGE THESE DEFAULTS.  IF OPTION IS SELECTED TO BE CHANGED, *
C       USER REVIEWS MENU AND SELECTS DESIRED METHOD AND IS RETURNED *
C       TO DEFAULT OPTION DISPLAY.  SOME OPTIONS ARE CHANGED UPON    *
C       SELECTION DUE TO ONLY TWO METHOD BEING POSSIBLE.  OPTIONS ARE *
C       RESET TO PROGRAMMER SPECIFIED DEFAULT UPON EACH CALL TO THIS *
C       SUBROUTINE.                                             *
C  CALLED BY: PROGRAM EDUC                                      *
C  CALLS    : SUBROUTINE CHECK2(D,N,M,INVAL,INEW)               *
C  VARIABLES:                                                   *
C       USED: INEW, INVAL                                       *
C   MODIFIED: D,DUAL,FMT,IBTAB,IFTAB,ITAB,MOD,OIU,OUTP,PES      *
C ****************************************
      SUBROUTINE OPTION
      CHARACTER D(10)*1
      INTEGER PES,OIU,DUAL,OUTP,TIE,FMT
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      MOD=1
      PES=1
      DUAL=1
      OIU=1
      ITAB=1
      IBTAB=1
      IFTAB=1
      OUTP=1
      FMT=1
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
C     DEFAULT OPTIONS DISPLAYED
      WRITE(1,'(12X,''DEFAULT OPTIONS''/5X,''ENTER OPTION NUMBER TO CHAN
     .GE'')')
      WRITE(1,'(''1.TABLEAU FORMATION'',11X,$)')
      IF(MOD .EQ. 1)THEN
        WRITE(1,'(6X,''USER'')')
      ELSE
        WRITE(1,'(1X,''ALGORITHM'')')
      ENDIF
      WRITE(1,'(''2.PIVOT ELEMENT SELECTION'',5X,$)')
      IF(PES .EQ. 1)THEN
        WRITE(1,'(1X,''USER SEL''/31X,''ALGOR CHK'')')
      ELSEIF(PES .EQ. 2)THEN
        WRITE(1,'(1X,''USER SEL''/31X,''NO CHECK'')')
      ELSE
        WRITE(1,'(1X,''ALGORITHM''/31X,''SELECTS'')')
      ENDIF
      WRITE(1,'(/,''3.DUAL PIVOTS'',26X,$)')
      IF(DUAL .EQ. 1)THEN
        WRITE(1,'(''N'')')
```

```
      ELSE
        WRITE(1,'(''Y'')')
      ENDIF
      WRITE(1,'(''4.INFEASIBLE,UNBOUNDED,OPTIMAL''/4X,''SELECTION IDENTI
     .FICATION'',2X,$)')
      IF(OIU .EQ. 1)THEN
        WRITE(1,'(6X,''USER'')')
      ELSE
        WRITE(1,'(1X,''ALGORITHM'')')
      ENDIF
      WRITE(1,'(/,''5.TABLEAUS TO BE DISPLAYED''/4X,''INITIAL'',19X,$)')
      IF(ITAB .EQ. 1)THEN
        WRITE(1,'(9X,''Y'')')
      ELSE
        WRITE(1,'(9X,''N'')')
      ENDIF
      WRITE(1,'(4X,''INTERMEDIATE'',18X,''N = '',I2)')IBTAB
      WRITE(1,'(4X,''FINAL'',21X,$)')
      IF(IFTAB .EQ. 1)THEN
        WRITE(1,'(9X,''Y'')')
      ELSE
        WRITE(1,'(9X,''N'')')
      ENDIF
      WRITE(1,'(''6.OUTPUT LOCATION'',13X,$)')
      IF(OUTP .EQ. 1)THEN
        WRITE(1,'(4X,''SCREEN'')')
      ELSE
        WRITE(1,'(3X,''PRINTER'')')
      ENDIF
      WRITE(1,'(/,''7.OUTPUT FORMAT'',15X,$)')
      IF(FMT .EQ. 0)THEN
        WRITE(1,'(2X,''E FORMAT'')')
      ELSE
        WRITE(1,'(2X,''F FORMAT'')')
      ENDIF
      WRITE(1,'(''8.NO CHANGES''//''*SEE DOCUMENTATION FOR EXPLAINATION
     .''//7X,''WHICH OPTION (ENTER 1-8)? '',$)')
      READ(5,'(A1)')D(1)
      CALL CHECK2(D,1,8,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,'(5X,''INVALID ENTRY PLEASE REENTER'')')
        GO TO 100
      ENDIF
      GOTO(130,220,270,300,400,500,530,560)INEW
130   WRITE(1,110)CHAR(12)
C     MODIFICATION OPTIONS DISPLAYED
      WRITE(1,140)
140   FORMAT(2X,'EDUCATIONAL MODULE OPTION SELECTION'/)
      WRITE(1,'(''IN ORDER TO PLACE THE LP MODEL INTO THE''/''PROPER FOR
     .M FOR THE SIMPLEX ALGORITHM''/''(OBJECTIVE FUNCTION CHANGES, ADDIT
     .ION OF''/''SLACK OR ARTIFICAL VARIABLES), WHICH''/''METHOD IS DESI
     .RED?''//)')
```

```
      WRITE(1,'(''1. USER SELECTS MODIFICATION AND''/3X,''ALGORITHM CHEC
     .KS''//18X,''OR''//''2. ALGORITHM PERFORMS MODIFICATIONS.''/3X''(NO
     . USER INPUT)'')')
160   WRITE(1,170)
170   FORMAT(/13X,'WHICH OPTION? ',$)
      READ(5,'(A1)')D(1)
      CALL CHECK2(D,1,2,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,180)
180     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 160
      ELSE
        MOD=INEW
      ENDIF
      GO TO 100
220   WRITE(1,110)CHAR(12)
C     PIVOT ELEMENT SELECTION OPTIONS DISPLAYED
      WRITE(1,140)
      WRITE(1,'(''IN SELECTION OF PIVOT ELEMENTS FOR THE''/''SIMPLEX ALG
     .ORITHM, WHICH METHOD WOULD''/''YOU LIKE?'')')
      WRITE(1,'(2(/),''1. USER SELECTS, ALGORITHM CHECKS.''/3X,''(MAY CH
     .ANGE SELECTION AFTER CHECK)''//''2. USER SELECTS, NO ALGORITHM CHE
     .CK.''//''3. ALGORITHM SELECTS, NO USER INPUT.'')')
240   WRITE(1,170)
      READ(5,'(A1)')D(1)
      CALL CHECK2(D,1,3,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,180)
        GO TO 240
      ELSE
        PES=INEW
      ENDIF
      GO TO 100
270   WRITE(1,110)CHAR(12)
C     DUAL PIVOT OPTIONS DISPLAYED
280   WRITE(1,140)
      WRITE(1,'(8(/)''WOULD YOU LIKE TO BE ABLE TO PERFORM''/14X,''DUAL
     . PIVOTS? '')')
      WRITE(1,'(/,''(DUAL PIVOTS ARE ALLOWED ONLY IF USER''/''SELECTS PI
     .VOT ROW AND COLUMN ELEMENTS)''//13X,''(Y/N,RETURN) '',$)')
      READ(5,'(A1)')D(1)
      IF(ICHAR(D(1)) .EQ. 89)THEN
        DUAL=2
      ELSEIF(ICHAR(D(1)) .EQ. 78)THEN
        DUAL=1
      ELSE
        WRITE(1,180)
        GO TO 270
      ENDIF
      GO TO 100
300   WRITE(1,110)CHAR(12)
C     IDENTIFICATION OF FINAL TABLEAU OPTIONS DISPLAYED
```

```
      WRITE(1,140)
      WRITE(1,'(''AS OPTIMAL, INFEASIBLE, OR UNBOUNDED''/''SOLUTIONS OCC
     .UR, WHICH METHOD WOULD YOU''/''LIKE?'')')
      WRITE(1,'(2(/),''1. USER ATTEMPTS TO IDENTIFY, ALGORITHM''/3X,''CH
     .ECKS.''//''2. SYSTEM IDENTIFIES AND REPORTS AS''/3X,''OCCURS.'')')
320   WRITE(1,170)
      READ(5,'(A1)')D(1)
      CALL CHECK2(D,1,2,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,180)
        GO TO 320
      ELSE
        OIU=INEW
      ENDIF
      GO TO 100
400   WRITE(1,110)CHAR(12)
C     TABLEAU DISPLAY OPTIONS SHOWN
      WRITE(1,140)
      WRITE(1,'(''WHICH TABLEAUS WOULD YOU LIKE DISPLAYED?'')')
420   WRITE(1,'(/5X,''INITIAL TABLEAU? (Y/N) '',$)')
      READ(5,'(A1)')D(1)
      IF(ICHAR(D(1)) .EQ. 89)THEN
        ITAB=1
      ELSEIF(ICHAR(D(1)) .EQ. 78)THEN
        ITAB=2
      ELSE
        WRITE(1,180)
        GO TO 420
      ENDIF
440   WRITE(1,'(/5X,''INTERMEDIATE TABLEAUS? (Y/N) '',$)')
      READ(5,'(A1)')D(1)
      IF(ICHAR(D(1)) .EQ. 89)THEN
        WRITE(1,'(/''EVERY N(TH) INTERMEDIATE TABLEAU WILL BE''/15X,''DI
     .SPLAYED.'')')
450     WRITE(1,'(/4X,''WHAT VALUE DO YOU DESIRE FOR N?''//17X,''N = '',
     .$)')
        READ(5,'(2A1)')D(1),D(2)
        CALL CHECK2(D,2,99,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,180)
          PAUSE
          GO TO 450
        ENDIF
        IBTAB=INEW
      ELSEIF(ICHAR(D(1)) .EQ. 78)THEN
        IBTAB=0
      ELSE
        WRITE(1,180)
        GO TO 440
      ENDIF
460   WRITE(1,'(/5X,''FINAL TABLEAU? (Y/N) '',$)')
      READ(5,'(A1)')D(1)
```

```
      IF(ICHAR(D(1)) .EQ. 89)THEN
        IFTAB=1
      ELSEIF(ICHAR(D(1)) .EQ. 78)THEN
        IFTAB=2
      ELSE
        WRITE(1,180)
        GO TO 460
      ENDIF
      GO TO 100
C     OUTPUT LOCATION OPTION CHANGED
500   IF(OUTP .EQ. 1)THEN
        OUTP=2
      ELSE
        OUTP=1
      ENDIF
      GO TO 100
C     OUTPUT FORMAT CHANGED
530   IF(FMT .EQ. 1)THEN
        FMT=0
      ELSE
        FMT=1
      ENDIF
      GO TO 100
560   RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 2 UNIT22                                                            *
C    UNIT $USES: UNIT26 AND UNIT27                                            *
C                                                                             *
C  SUBROUTINE READY                                                           *
C  USE: DISPLAYS QUESTIONS CONCERNING WHETHER OR NOT THE TABLEAU              *
C       WHICH IS DISPLAYED IS AN INITIAL BASIC SOLUTION AND READY FOR         *
C       THE INITIAL PIVOT OF THE SIMPLEX ALGORITHM.  IF ARTIFICIAL            *
C       VARIABLES HAVE BEEN ADDED, THE "BIG M" METHOD MUST HAVE BEEN          *
C       APPLIED TO THE VARIABLES SPECIFIED BY USER.  USER IS GIVEN            *
C       IMMEDIATE FEEDBACK AND INSTRUCTIONAL COMMENTS WITH BOTH               *
C       CORRECT AND INCORRECT RESPONSES.  ROUTINE MODIFIES TABLEAU            *
C       CORRECTLY REGARDLESS OF USER INPUT.                                   *
C  CALLED BY: PROGRAM EDUC                                                    *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                             *
C             SUBROUTINE TDISPL                                               *
C  VARIABLES:                                                                 *
C       USED: BM,INEW,INVAL,KFA,NEC,NGC,VT                                    *
C   MODIFIED: C(*),IFLAG(3),IFLAG(9),P(*)                                     *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK2 IN UNIT27.CODE OVERLAY
$USES UTDISPL IN UNIT26.CODE OVERLAY
      SUBROUTINE READY
      CHARACTER VN*6,CN*6,FN*20,MM*3,FN*10,PINEG*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOC,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(2  N(20),PN,MM,FN,PINEQ(20),P(10),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(4(/),''THE TABLEAU AS MODIFIED PREVIOUSLY,''/''WILL BE D
     .ISPLAYED.''//''YOU WILL THEN BE ASKED IF THE TABLEAU IS''/''IN THE
     . CORRECT FORM FOR THE SIMPLEX''/''ALGORITHM.''//)')
      PAUSE
      WRITE(1,110)CHAR(12)
C     FLAGS ALLOW DISPLAY OF TABLEAU WITHOUT BASIC VARIABLES ANNOTATED
      IFLAG(3)=1
      IFLAG(9)=2
      OPEN(2,FILE='CONSOLE:')
      CALL TDISPL
      WRITE(1,110)CHAR(12)
      WRITE(1,'(6(/),1X,''IS THE TABLEAU IN THE PROPER FORM FOR''/10X,''
     .THE INITIAL PIVOT? '',$)')
      READ(5,'(A1)')P(1)
      IF((NEC+NGC) .EQ. 0)THEN
        IF(ICHAR(P(1)) .EQ. 89)THEN
          WRITE(1,130)
130       FORMAT(//7X,'YOUR RESPONSE WAS CORRECT.'/2X,'THE TABLEAU IS IN
     . THE PROPER FORM.',//)
```

```
          PAUSE
          GO TO 300
        ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
          WRITE(1,150)
150       FORMAT(//6X,'YOUR RESPONSE WAS INCORRECT.'/)
          WRITE(1,160)
160       FORMAT(/'ONLY SLACK VARIABLES HAVE BEEN ADDED, SO'/2X,'NO FURT
     .HER MODIFICATIONS ARE NEEDED.'//3X,'YOU PRESENTLY HAVE AN INITIAL
     . BASIC'/15X,'SOLUTION.'//)
          PAUSE
          GO TO 300
        ELSE
          WRITE(1,180)
180       FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
          WRITE(1,'(2(/))')
          PAUSE
          GO TO 100
        ENDIF
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        WRITE(1,200)
200     FORMAT(//7X,'YOUR RESPONSE WAS CORRECT.'/2X,'FURTHER MODIFICATIO
     .NS ARE REQUIRED.'//)
        PAUSE
        GO TO 300
      ELSEIF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,150)
        WRITE(1,220)
220     FORMAT(/'ARTIFICAL VARIABLES HAVE BEEN ADDED, YET'/'THE OBJECTIV
     .E FUNCTION HAS NOT BEEN'/'MODIFIED (BIG M) TO REFLECT THIS.'//)
        PAUSE
        GO TO 300
      ELSE
        WRITE(1,180)
        WRITE(1,'(2(/))')
        PAUSE
        GO TO 100
      ENDIF
300   IF((NGC+NEC) .NE. 0)THEN
        WRITE(1,110)CHAR(12)
        WRITE(1,'(9(/),''THE TABLEAU WILL BE DISPLAYED AND YOU''/''WILL
     . BE ASKED TO IDENTIFY THOSE''/''VARIABLES WHICH THE BIG M METHOD I
     .S TO''/''BE APPLIED.''///)')
        PAUSE
        IFLAG(3)=1
        IFLAG(9)=0
        OPEN(2,FILE='CONSOLE:')
        WRITE(1,110)CHAR(12)
        CALL TDISPL
310     WRITE(1,110)CHAR(12)
        WRITE(1,320)
320     FORMAT(1X,'WHICH VARIABLES REQUIRE THE USE OF THE'/14X,'BIG M ME
     .THOD?'/9X,'(ENTER SUBSCRIPT VALUES)',/)
```

```
      WRITE(1,'(9X,''FIRST VARIABLE? '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,VT,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,180)
        PAUSE
        GO TO 310
      ENDIF
      IF(INEW .EQ. KFA)THEN
        WRITE(1,340)KFA
340     FORMAT(//7X,'YOUR RESPONSE WAS CORRECT.'/1X,I2,' IS THE FIRST
     . ARTIFICAL VARIABLE AND'/5X,'REQUIRES THE USE OF THE BIG M.')
      ELSE
        WRITE(1,360)KFA
360     FORMAT(//6X,'YOUR RESPONSE WAS INCORRECT.'/1X,'THE CORRECT RES
     .PONSE WAS VARIABLE ',I2/'THIS IS THE FIRST ARTIFICAL VARIABLE AND'
     ./2X,'REQUIRES THE USE OF THE BIG M METHOD.')
      ENDIF
      PAUSE
380   WRITE(1,110)CHAR(12)
      WRITE(1,'(3X,''VARIABLES '',I2,'' THRU X(?) REQUIRE THE''/13X,''
     .BIG M METHOD''/)')KFA
      WRITE(1,'(9X,''LAST VARIABLE? '',$)')
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,VT,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,180)
        PAUSE
        GO TO 380
      ENDIF
      IF(INEW .EQ. VT)THEN
        WRITE(1,400)VT
400     FORMAT(//7X,'YOUR RESPONSE WAS CORRECT.'/1X,'THE LAST ARTIFICI
     .AL VARIABLE IS #',I2,' AND'/2X,'IS THE LAST TO REQUIRE THE USE OF
     . THE'/14X,'BIG M METHOD.')
        PAUSE
      ELSE
        WRITE(1,420)VT
420     FORMAT(//6X,'YOUR RESPONSE WAS INCORRECT.'/1X,'THE LAST ARTIFI
     .CIAL VARIABLE IS #',I2,' AND'/1X,'IS THE LAST TO REQUIRE THE USE O
     .F THE'/14X,' BIG M METHOD.')
        PAUSE
      ENDIF
      DO 450 J=KFA,VT
        C(J)=-BM
450   CONTINUE
      WRITE(1,110)CHAR(12)
      WRITE(1,'(10(/),1X,''THE TABLEAU WILL BE DISPLAYED, THEN YOU''/2
     .X,''WILL BE ASKED IF IT IS IN THE PROPER''/6X,''FORM FOR THE INITI
     .AL PIVOT.''//)')
      PAUSE
C     FLAGS ALLOW TABLEAU WITH BASIC VARIABLES ANNOTATED TO BE
```

```
C         DISPLAYED
        IFLAG(3)=1
        IFLAG(9)=0
        OPEN(2,FILE='CONSOLE:')
        WRITE(1,110)CHAR(12)
        CALL TDISPL
480     WRITE(1,110)CHAR(12)
        WRITE(1,'(1X,''IS THE TABLEAU IN THE PROPER FORM FOR''/10X,''THE
     . INITIAL PIVOT? '',$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 78)THEN
          WRITE(1,'(//7X,''YOUR RESPONSE WAS CORRECT'')')
          WRITE(1,500)
500       FORMAT('THERE IS NO INITIAL BASIC SOLUTION SINCE'/'THE OBJECTI
     .VE FUNCTION COEFFICIENTS OF'/'THE ARTIFICAL VARIABLES ARE NOT ZERO
     ..',/)
          PAUSE
        ELSEIF(ICHAR(P(1)) .EQ. 89)THEN
          WRITE(1,'(//6X,''YOUR RESPONSE WAS INCORRECT.'')')
          WRITE(1,500)
          PAUSE
        ENDIF
      ENDIF
      RETURN
      END
```

```
C ***************************************
C  MODULE 2 UNIT22                                                       *
C                                                                        *
C  SUBROUTINE CNMDU                                                      *
C  USE: DISPLAYS THE CONSTRAINTS AND A MENU OF OPTIONS WHICH MAY BE      *
C       APPLIED TO CONSTRAINTS TO PREPARE THE CONSTRAINT FOR THE         *
C       SIMPLEX ALGORITHM (ADD VARIABLES. MULTIPLY BY -1).  SOLICITS     *
C       USER INPUT AND PRESENTS IMMEDIATE FEEDBACK.  IF INCORRECT        *
C       RESPONSE GIVEN, BRIEF INSTRUCTIONAL COMMENTS ARE DISPLAYED.      *
C       CONSTRAINT IS MODIFIED CORRECTLY REGARDLESS OF USER INPUT.       *
C       CONSTRAINTS WITH NEGATIVE RHS'S ARE MULTIPLIED BY -1.            *
C  CALLED BY: PROGRAM EDUC                                               *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                        *
C             SUBROUTINE TDISPL                                          *
C  VARIABLES:                                                            *
C       USED: CN(*),IFLAG(5),INEW,INVAL,F,V,VN(*)                        *
C   MODIFIED: A(*,*),IFLAG(3),IFLAG(9),INEQ(*),NGC,NLC,PINEQ(*),         *
C             P(*),S,T,VT,XB(*)                                          *
C ***************************************
      SUBROUTINE CNMDU
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT,T,S
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
C     FLAG ALLOWS ONLY CONSTRAINTS TO BE DISPLAYED
      IFLAG(9)=1
      OPEN(2,FILE='CONSOLE:')
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(8X,''CONSTRAINT MODIFICATION'',8(/),''THE CONSTRAINTS, A
     .S ENTERED, WILL BE''/''DISPLAYED NEXT.  AFTER THE DISPLAY, YOU''/
     .''WILL BE SHOWN EACH OF THE CONSTRAINTS'')')
      WRITE(1,'(''INDIVIDUALLY AND ASKED TO SELECT THE''/''OPTION WHICH
     . TRANSFORMS THE CONSTRAINT''/''INTO THE PROPER SIMPLEX ALGORITHM F
     .ORM.'',7(/))')
      PAUSE
      WRITE(1,110)CHAR(12)
C     SET VARIABLES IN FORM APPROPRIATE FOR TDISPL SUBROUTINE
      VT=V
C     FLAG INSURES SCREEN IS CLEARED AFTER EACH 80 COLUMN DISPLAY
      IFLAG(3)=1
      CALL TDISPL
      WRITE(1,110)CHAR(12)
      WRITE(1,'(''EACH CONSTRAINT WILL BE SEPARATELY''/''DISPLAYED, THEN
     . THE FOLLOWING OPTIONS''/''WILL BE DISPLAYED FOR EACH CONSTRAINT.
     .'')')
      WRITE(1,'(''YOU WILL SELECT THE OPTION WHICH WILL'', ''PLACE THE C
     .ONSTRAINT IN THE PROPER''/''SIMPLEX ALGORITHM FORM.'' )')
```

```
      WRITE(1,150)
150   FORMAT('1. ADD SLACK VARIABLE ONLY.'//'2. SUBTRACT SURPLUS VARIABL
     .E, ADD'/3X,'ARTIFICAL VARIABLE.'//'3. ADD ARTIFICAL VARIABLE ONLY.
     .')
      WRITE(1,160)
160   FORMAT(/'4. MULTIPLY BY -1, SUBTRACT SURPLUS'/3X,'VARIABLE, ADD AR
     .TIFICAL VARIABLE.'//'5. MULTIPLY BY -1, ADD SLACK VARIABLE.'//'6.
     . MULTIPLY BY -1, ADD ARTIFICAL'/3X,'VARIABLE.')
      PAUSE
      WRITE(1,110)CHAR(12)
      T=(V/5)+1
      DO 900 I=1,K
        DO 400 N=1,T
          IF(IFLAG(5) .EQ. 1)THEN
            WRITE(1,'(13X,$)')
            DO 270 J=(N*5)-4,N*5
              IF(J .GT. V)THEN
                GO TO 270
              ENDIF
              WRITE(1,260)VN(J)
260           FORMAT(6X,A6,1X,$)
270         CONTINUE
            WRITE(1,'('' '')')
          ENDIF
          WRITE(1,'(13X,$)')
          DO 290 J=(N*5)-4,N*5
            IF(J .GT. V)THEN
              GO TO 290
            ENDIF
            WRITE(1,280)J
280         FORMAT(6X,'X(',I2,')',2X,$)
290       CONTINUE
          IF(T .EQ. 1 .OR. N .EQ. T)THEN
            WRITE(1,300)
300         FORMAT(7X,'RHS')
          ELSE
            WRITE(1,'('' '')')
          ENDIF
          WRITE(1,340)I
340       FORMAT('CN#',I2,$)
          IF(IFLAG(5) .EQ. 1)THEN
            WRITE(1,350)CN(I)
350         FORMAT(1X,A6,1X,$)
          ELSE
            WRITE(1,'(8X,$)')
          ENDIF
          DO 370 J=(N*5)-4,N*5
            IF(J .GT. V)THEN
              GO TO 370
            ENDIF
            IF(FMT .EQ. 0)THEN
              WRITE(1,'(1X,1PE12.5,$)')A(I,J)
```

```
            ELSE
              WRITE(1,'(1X,F12.5,$)')A(I,J)
            ENDIF
370       CONTINUE
          IF(T .EQ. 1 .OR. N .EQ. T)THEN
            IF(FMT .EQ. 0)THEN
              WRITE(1,'(1X,A1,1PE12.5,//)')PINEQ(I),XB(I)
            ELSE
              WRITE(1,'(1X,A1,F12.5,//)')PINEQ(I),XB(I)
            ENDIF
          ELSE
            WRITE(1,'('' '')')
          ENDIF
400     CONTINUE
        PAUSE
410     WRITE(1,110)CHAR(12)
        WRITE(1,'(13X,''CONSTRAINT #'',I2,//)')I
        WRITE(1,150)
        WRITE(1,160)
        WRITE(1,420)
420     FORMAT(/13X,'WHICH OPTION? ',$)
        READ(5,'(A1)')P(1)
        CALL CHECK2(P,1,6,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,430)
430       FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
          PAUSE
          GO TO 410
        ENDIF
C       CHECKS FOR NEGATIVE RHS
        IF(XB(I) .GT. 0)THEN
          IF(INEQ(I) .EQ. 0)THEN
C           USER SELECTION CHECKED AND FEEDBACK PROVIDED
            IF(INEW .EQ. 1)THEN
              WRITE(1,450)INEW
450           FORMAT(/10X,'OPTION #',I1,' IS CORRECT.'/)
              PAUSE
              GO TO 800
            ELSE
              S=1
              WRITE(1,460)INEW,S
460           FORMAT(/9X,'OPTION #',I1,' IS INCORRECT'//4X,'THE PROPER R
     .ESPONSE WAS OPTION #',I1,)
              PAUSE
              GO TO 800
            ENDIF
          ELSEIF(INEQ(I) .EQ. 1)THEN
            IF(INEW .EQ. 2)THEN
              WRITE(1,450)INEW
              PAUSE
              GO TO 800
            ELSE
```

```
          S=2
          WRITE(1,460)INEW,S
          PAUSE
          GO TO 800
        ENDIF
      ELSEIF(INEW .EQ. 3)THEN
        WRITE(1,450)INEW
        PAUSE
        GO TO 800
      ELSE
        S=3
        WRITE(1,460)INEW,S
        PAUSE
        GO TO 300
      ENDIF
    ELSE
      IF(INEQ(I) .EQ. 0)THEN
        IF(INEW .EQ. 4)THEN
          WRITE(1,450)INEW
          PAUSE
        ELSE
          S=4
          WRITE(1,460)INEW,S
          PAUSE
        ENDIF
      ELSEIF(INEQ(I) .EQ. 1)THEN
        IF(INEW .EQ. 5)THEN
          WRITE(1,450)INEW
          PAUSE
        ELSE
          S=5
          WRITE(1,460)INEW,S
          PAUSE
        ENDIF
      ELSEIF(INEW .EQ. 6)THEN
        WRITE(1,450)INEW
        PAUSE
      ELSE
        S=6
        WRITE(1,460)INEW,S
        PAUSE
      ENDIF
    ENDIF
C   CONSTRAINTS WITH NEGATIVE RHS MULTIPLIED BY -1
    XB(I)=-XB(I)
    DO 500 J=1,V
      A(I,J)=-A(I,J)
500 CONTINUE
C   COUNT OF INEQUALITIES BY TYPE CORRECTED DUE TO MULT BY -1
    IF(INEQ(I) .EQ. 0)THEN
      NLC=NLC-1
      NGC=NGC+1
```

```
        INEQ(I)=1
        PINEQ(I)='>'
      ELSEIF(INEQ(I) .EQ. 1)THEN
        NGC=NGC-1
        NLC=NLC+1
        INEQ(I)=0
        PINEQ(I)='<'
      ENDIF
800   WRITE(1,110)CHAR(12)
900 CONTINUE
    RETURN
    END
```

```
C ****************************************
C  MODULE 2 UNIT23                                                        *
C    UNIT $USES: UNIT26 AND UNIT27                                        *
C                                                                         *
C  SUBROUTINE OPT                                                         *
C  USE: DETERMINES THE PIVOT ELEMENT, OPTIMALITY, UNBOUNDEDNESS, AND      *
C       FEASIBILITY OF THE CURRENT TABLEAU.  DEPENDENT OF THE             *
C       INTERACTION OPTIONS SELECTED BY USER, THE USER WILL BE ASKED      *
C       QUESTIONS ON THE ABOVE CONDITIONS AND PRESENTED FEEDBACK          *
C       ACCORDINGLY.  ALSO DEPENDENT UPON OPTION SELECTION, THE PIVOT     *
C       ELEMENT MAY BE THAT SELECTED BY THE ALGORITHM OR AS INPUT BY      *
C       USER.  THIS ROUTINE ALSO DISPLAYS A TABLEAU HEADER ON THE         *
C       SELECTED OUTPUT DEVICE FOR DESIGNATED TABLEAUS.  USER MAY         *
C       SELECT PIVOT ELEMENT TO CAUSE SYSTEM OVERFLOW ERROR, BUT USER     *
C       IS GIVEN OPTION TO ABORT PIVOT PRIOR TO OVERFLOW ERROR.           *
C  CALLED BY: PROGRAM EDUC                                                *
C  CALLS    : SUBROUTINE BASDIS                                           *
C             SUBROUTINE TCAL                                             *
C             SUBROUTINE TDISPL                                           *
C  VARIABLES:                                                             *
C       USED: A(*,*),C(*),CB(*),IFTAB,K,KFA,OIU,OUTP,VT,XB(*)             *
C   MODIFIED: CO,F,GNEG,IFLAG(1),IFLAG(4),IFLAG(6),IFLAG(7),IFLAG(8),     *
C             IFLAG(9),INC,INF2,NNU,ODB,OPTS,PK,PR,SPR,TIE                *
C ****************************************
$USES UCHECK2 IN UNIT27.CODE OVERLAY
$USES UTDISPL IN UNIT26.CODE OVERLAY
      SUBROUTINE OPT
      CHARACTER CO*7,INC*5,ODB*10,NNU*14,F*1
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     ..NEC,NEC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      IFLAG(1)=0
      IFLAG(4)=0
      IFLAG(6)=0
      INF1=0
      GNEG=0.0
C     CHECKS FOR INFEASIBILITY
      DO 100 I=1,K
        IF (XB(I) .LT. 0.0)THEN
C         FLAG DENOTES VARIABLE AT NEGATIVE LEVEL
          IFLAG(1)=I
        ENDIF
        IF(CB(I) .GE. KFA)THEN
C         DETERMINES IF VARIABLE IS AN ARTIFICIAL VARIABLE
          IF(XB(I) .GE. .0001)THEN
            FLAG DENOTES ARTIFICIAL VARIABLE AT POSITIVE LEVEL
            INF1=1
          ENDIF
        ENDIF
```

```
100   CONTINUE

C     FINDS THE LARGEST NEGATIVE Z(J)-C(J)
      DO 200 J=1,VT
        IF(C(J) .LT. GNEG)THEN
          TIE=0
          GNEG=C(J)
          PK=J
        ELSEIF(C(J) .EQ. GNEG)THEN
C         TIE EXISTS FOR ENTERING BASIC VARIABLE
          TIE=1
        ENDIF
200   CONTINUE
      IF(GNEG .GT. -.0001 .AND. IFLAG(1) .EQ. 0)THEN
C       NO NEGATIVE Z(J)-C(J) EXISTS AND RHS'S ARE POSITIVE
          SO OPTIMAL
        OPTS=1
      ENDIF
      CO='CORRECT'
      INC='INCORRECT'
      ODB='OPTIMAL'
      NNU='NOT OPTIMAL'
210   IF(OIU .EQ. 1)THEN
C       USER HAS SELECTED TO IDENTIFY OPTIMALITY
        WRITE(1,220)CHAR(12)
220     FORMAT(A)
        WRITE(1,510)
        WRITE(1,'(3X,''WAS THE PREVIOUS TABLEAU OPTIMAL? '',$)')
        READ(5,'(A1)')F
        IF(ICHAR(F) .EQ. 84)THEN
C         USER ELECTS TO REVIEW TABLEAU
          CALL TCAL
          GO TO 210
        ELSEIF(ICHAR(F) .NE. 89 .AND. ICHAR(F) .NE. 78)THEN
          WRITE(1,230)
230       FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER',//)
          GO TO 210
        ENDIF
C       USER INPUT CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
        IF(OPTS .EQ. 1)THEN
          IF(ICHAR(F) .EQ. 89)THEN
            WRITE(1,240)CO,ODB
240         FORMAT(//6X,'YOUR RESPONSE WAS ',A9/2X,'THE LAST TABLEAU WAS
     . ',A15,//)
          ELSE
            WRITE(1,240)INC,ODB
            WRITE(1,'(''TO BE OPTIMAL, ALL C(J) AND RHS VALUES''/''MUST
     . BE ZERO OR POSITIVE.  THIS IS''/''CURRENTLY TRUE, THEREFORE THE T
     .ABLEAU IS''/''OPTIMAL.'')')
          ENDIF
          PAUSE
          GO TO 500
```

```
      ELSEIF(OPTS .EQ. 0)THEN
        IF(ICHAR(F) .EQ. 78)THEN
          WRITE(1,240)CO,NNU
        ELSE
          WRITE(1,240)INC,NNU
          WRITE(1,'(''TO BE OPTIMAL, ALL C(J) AND RHS VALUES''/''MUST
     . BE ZERO OR POSITIVE.  THIS IS''/''CURRENTLY NOT TRUE, THEREFORE T
     .HE''/''TABLEAU IS NOT OPTIMAL.'')')
        ENDIF
        PAUSE
      ENDIF
     ELSE
C       ALGORITHM TO IDENTIFY OPTIMAL SOLUTION
      IF(OPTS .EQ. 1)THEN
        IF(INF1 .EQ. 0 .AND. IFLAG(1) .EQ. 0)THEN
          WRITE(1,220)CHAR(12)
          WRITE(1,'(11(/),2X,''THE LAST BASIC SOLUTION WAS OPTIMAL.'',
     .6(/))')
        ELSEIF(INF1 .NE. 0)THEN
          WRITE(1,'(10(/),2X,''A FEASIBLE SOLUTION DOES NOT EXIST.''/1
     .X,''AN ARTIFICAL VARIABLE IS AT A POSITIVE''/9X,''LEVEL IN THE SOL
     .UTION.'',6(/))')
        ELSEIF(IFLAG(1) .NE. 0)THEN
          WRITE(1,'(10(/),2X,''THE SOLUTION IS INFEASIBLE SINCE THE''/
     .''BASIC VARIABLE X('',I2,'') IS NEGATIVE.'')')CB(IFLAG(1))
        ENDIF
        PAUSE
      ENDIF
     ENDIF
500  ODB='FEASIBLE'
     NNU='INFEASIBLE'
510  FORMAT(6X,'TO REVIEW TABLEAU, ENTER T',//)
     IF(OIU .EQ. 1)THEN
C       USER HAS SELECTED TO IDENTIFY INFEASIBLE SOLUTIONS
      WRITE(1,220)CHAR(12)
      WRITE(1,510)
      IF(OPTS .EQ. 1)THEN
        WRITE(1,'(1X,''IS THE OPTIMAL SOLUTION ALSO FEASIBLE?''/19X,$)
     .')
      ELSE
        WRITE(1,'(8X,''IS THE SOLUTION FEASIBLE? '',$)')
      ENDIF
      READ(5,'(A1)')F
      IF(ICHAR(F) .EQ. 84)THEN
C         USER ELECTS TO REVIEW TABLEAU
        CALL TCAL
        GO TO 500
      ELSEIF(ICHAR(F) .NE. 78 .AND. ICHAR(F) .NE. 89)THEN
        WRITE(1,230)
        PAUSE
        GO TO 500
      ENDIF
```

```
C       USER PROVIDED FEEDBACK ON RESPONSE
        IF(INF1 .EQ. 0 .AND. IFLAG(1) .EQ. 0)THEN
          IF(ICHAR(F) .EQ. 89)THEN
            WRITE(1,240)CO,ODB
          ELSEIF(ICHAR(F) .EQ. 78)THEN
            WRITE(1,240)INC,ODB
            WRITE(1,'(1X,''THE CURRENT TABLEAU IS FEASIBLE SINCE''/2X,''
     .ALL RHS VALUES ARE POSITIVE AND ALSO''/4X,''ANY ARTIFICAL VARIABLE
     .S ARE AT A''/14X,''ZERO VALUE.''/)')
          ENDIF
          PAUSE
        ELSE
          IF(ICHAR(F) .EQ. 78)THEN
            WRITE(1,240)CO,NNU
            IF(INF1 .NE. 0)THEN
              WRITE(1,540)CB(INF1)
540           FORMAT(/'THE SOLUTION IS INFEASIBLE SINCE THE'/'ARTIFICI
     .AL VARIABLE X(',I2,') IS AT A'/'POSITIVE LEVEL.')
            ELSE
              WRITE(1,560)CB(IFLAG(1))
560           FORMAT(/'THE SOLUTION IS INFEASIBLE SINCE THE'/'BASIC VA
     .RIABLE X(',I2,') IS AT A NEGATIVE'/'LEVEL.')
            ENDIF
          ELSEIF(ICHAR(F) .EQ. 89)THEN
            WRITE(1,240)INC,NNU
            IF(INF1 .NE. 0)THEN
              WRITE(1,540)CB(INF1)
            ELSE
              WRITE(1,560)CB(IFLAG(1))
            ENDIF
          ENDIF
          PAUSE
        ENDIF
      ENDIF
      IF((OPTS .EQ. 1 .AND. INF1 .NE. 0) .OR. (IFLAG(1) .NE. 0 .AND.
     . GNEG .GT. -.0001))THEN
C         OPTIMAL AND INFEASIBLE SOLUTIONS NOT CHECKED FOR UNBOUNDEDNESS
          GO TO 840
        ENDIF
      ENDIF
      ODB='DEGENERATE'
      NNU='NOT DEGENERATE'
C     SOLUTION CHECKED FOR DEGENERACY
      DO 620 I=1,K
        IF(XB(I) .GT. -.0001 .AND. XB(I) .LT. +.0001)THEN
          IFLAG(6)=1
        ENDIF
620   CONTINUE
      IF(OIU .EQ. 1)THEN
C       USER HAS ELECTED TO IDENTIFY DEGENERATE SOLUTIONS
630     WRITE(1,220)CHAR(12)
        WRITE(1,510)
```

```
      WRITE(1,'(''WAS THE PREVIOUS SOLUTION DEGENERATE? '',$)')
      READ(5,'(A1)')F
      IF(ICHAR(F) .EQ. 84)THEN
C       USER ELECTS TO REVIEW TABLEAU
        CALL TCAL
        GO TO 630
      ELSEIF(ICHAR(F) .NE. 78 .AND. ICHAR(F) .NE. 89)THEN
        WRITE(1,230)
        PAUSE
        GO TO 630
      ENDIF
C     USER RESPONSE CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
      IF(IFLAG(6) .EQ. 0)THEN
        IF(ICHAR(F) .EQ. 78)THEN
          WRITE(1,240)CO,NNU
        ELSEIF(ICHAR(F) .EQ. 89)THEN
          WRITE(1,240)INC,NNU
          WRITE(1,'(1X,''THE CURRENT TABLEAU IS NOT DEGENERATE''/''SIN
     .CE ALL BASIC VALUES ARE AT A NON-ZERO''/17X,''LEVEL.''/)')
        ENDIF
        PAUSE
      ELSE
        IF(ICHAR(F) .EQ. 89)THEN
          WRITE(1,240)CO,ODB
          WRITE(1,650)CB(IFLAG(6))
650       FORMAT(/2X,'BASIC VARIABLE X(',I2,') IS ZERO IN THIS'/2X,'SO
     .LUTION.')
        ELSEIF(ICHAR(F) .EQ. 78)THEN
          WRITE(1,240)INC,ODB
          WRITE(1,650)CB(IFLAG(6))
        ENDIF
        PAUSE
      ENDIF
    ENDIF
    IF(OPTS .EQ. 0)THEN
C     NON-OPTIMAL SOLUTION-PIVOT ROW DETERMINED
      SPR=10.E8
      DO 700 I=1,K
        IF(A(I,PK) .LE. .0001)THEN
          GO TO 700
        ELSEIF((XB(I)/A(I,PK)) .GE. SPR)THEN
          GO TO 700
        ELSE
          SPR=(XB(I)/A(I,PK))
          PR=I
        ENDIF
700   CONTINUE
      IF(SPR .GE. 10.E6)THEN
C       RATIO IS INFINITE, THEREFORE SOLUTION UNBOUNDED
        IFLAG(7)=1
      ENDIF
      ODB='BOUNDED'
```

```
         NNU='UNBOUNDED'
         IF(GIU .EQ. 1)THEN
C          USER HAS ELECTED TO IDENTIFY UNBOUNDEDNESS
710      WRITE(1,220)CHAR(12)
           WRITE(1,510)
           WRITE(1,'(2X,''HAS THE PREVIOUS SOLUTION UNBOUNDED''/1X,''BASE
     .D UPON THE NEXT PIVOT COLUMN (ROW)''/''BEING THE COLUMN (ROW) WITH
     . THE LARGEST''/4X,''NEGATIVE Z(J)-C(J) (B(J)) VALUE? '',$)')
           READ(5,'(A1)')F
           IF(ICHAR(F) .EQ. 84)THEN
C            USER ELECTS TO REVIEW TABLEAU
             CALL TCAL
             GO TO 710
           ELSEIF(ICHAR(F) .NE. 78 .AND. ICHAR(F) .NE. 89)THEN
             WRITE(1,230)
             PAUSE
             GO TO 710
           ENDIF
C          USER INPUT CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
           IF(IFLAG(7) .EQ. 1)THEN
             IF(ICHAR(F) .EQ. 89)THEN
               WRITE(1,240)CO,NNU
               WRITE(1,720)PK
720            FORMAT(/3X,'COLUMN ',I2,' COEFFICIENTS ARE ZERO OR'/1X,'NE
     .GATIVE, OR THE RATIO OF XB(I)/A(I,J)'/10X,'IS EXTREMELY LARGE.')
             ELSEIF(ICHAR(F) .EQ. 78)THEN
               WRITE(1,240)INC,NNU
               WRITE(1,720)PK
             ENDIF
             PAUSE
             GO TO 840
           ELSE
             IF(ICHAR(F) .EQ. 78)THEN
               WRITE(1,240)CO,ODB
             ELSEIF(ICHAR(F) .EQ. 89)THEN
               WRITE(1,240)INC,ODB
               WRITE(1,'(2X,''THE CURRENT TABLEAU IS BOUNDED SINCE''/1X,
     .''ALL THE A(I,J) VALUES IN COLUMN '',I2,'' ARE''/9),''NOT NEGATIVE
     . OR ZERO.''//)')PK
             ENDIF
             PAUSE
           ENDIF
         ELSE
           IF(IFLAG(7) .EQ. 1)THEN
             WRITE(1,220)CHAR(12)
             WRITE(1,'(11(/),1X,''THE LAST BASIC SOLUTION WAS UNBOUNDED.
     .'',6(/))')
             PAUSE
             GO TO 840
           ENDIF
         ENDIF
       ENDIF
```

```
C     CHECK FOR MULTIPLE OPTIMAL SOLUTIONS
      DO 760 J=1,VT
        IFLAG(8)=0
        DO 750 I=1,K
          IF(CB(I) .EQ. J)THEN
            IFLAG(8)=1
          ENDIF
750     CONTINUE
        IF(IFLAG(8) .EQ. 0)THEN
          IF(C(J) .LT. .0001 .AND. C(J) .GT. -.0001)THEN
            IFLAG(4)=1
          ENDIF
        ENDIF
760   CONTINUE
      IF(OPTS .EQ. 1)THEN
        IF(OIU .EQ. 1)THEN
C         USER HAS ELECTED TO IDENTIFY MULTIPLE OPTIMAL SOLUTIONS
770       WRITE(1,220)CHAR(12)
          WRITE(1,510)
          WRITE(1,'(1X,''ARE THERE MULTIPLE OPTIMAL SOLUTIONS? '',$)')
          READ(5,'(A1)')F
          IF(ICHAR(F) .EQ. 84)THEN
C           USER ELECTS TO REVIEW TABLEAU
            CALL TCAL
            GO TO 770
          ELSEIF(ICHAR(F) .NE. 78 .AND. ICHAR(F) .NE. 89)THEN
            WRITE(1,230)
            PAUSE
            GO TO 770
          ENDIF
C         USER RESPONSE CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
          IF(IFLAG(4) .EQ. 1)THEN
            IF(ICHAR(F) .EQ. 89)THEN
              WRITE(1,780)CO
780           FORMAT(//6X,'YOUR RESPONSE WAS ',A9/6X,'THERE ARE MULTIPLE
     . SOLUTIONS.')
              WRITE(1,800)
800           FORMAT(/'A NON-BASIC VARIABLE HAS A ZERO'/'COEFFICIENT IN
     .THE OBJECTIVE FUNCTION OF'/'THE OPTIMAL SOLUTION.')
            ELSEIF(ICHAR(F) .EQ. 78)THEN
              WRITE(1,780)INC
              WRITE(1,800)
            ENDIF
            PAUSE
          ELSE
            IF(ICHAR(F) .EQ. 78)THEN
              WRITE(1,820)CO
820           FORMAT(//6X,'YOUR RESPONSE WAS ',A9/4X,'THERE ARE NO MULTI
     .PLE SOLUTIONS.')
            ELSEIF(ICHAR(F) .EQ. 89)THEN
              WRITE(1,820)INC
              WRITE(1,'(//1X,''THIS IS SINCE ALL NON-BASIC VARIABLES''/1
```

```
     .X,''HAVE A VALUE OF OTHER THAN ZERO IN THE''/''OBJECTIVE FUNCTION
     . ROW. IF A ZERO VALUE''/1X,''WAS PRESENT FOR A NON-BASIC VARIABLE.
     .'')')
               WRITE(1,'(1X,''INCREASING THE VALUE OF THIS VARIABLE''/5X,
     .''WOULD NOT CHANGE THE Z VALUE.'')')
             ENDIF
             PAUSE
           ENDIF
         ENDIF
       ENDIF
       IF(OPTS .EQ. 0)THEN
         GO TO 880
       ENDIF
840    IF(IFTAB .EQ. 2)THEN
C        FINAL TABLEAU IS NOT TO BE DISPLAYED
         RETURN
       ELSE
         WRITE(1,220)CHAR(12)
         IF(OUTP .EQ. 1)THEN
           OPEN(2,FILE='CONSOLE:')
         ELSE
           OPEN(2,FILE='PRINTER:')
         ENDIF
C        TABLEAU HEADER PRINTED
         WRITE(2,'(10X,''BASIC SOLUTION #'',I2)')BASIC
         WRITE(2,'(10X,''FINAL TABLEAU - '',$)')
         IF(INFI .NE. 0 .OR. IFLAG(1) .NE. 0)THEN
           WRITE(2,'(''INFEASIBLE''/)')
         ELSEIF(IFLAG(7) .EQ. 1)THEN
           WRITE(2,'(''UNBOUNDED''/)')
         ELSE
           WRITE(2,'(''OPTIMAL''/)')
         ENDIF
         IFLAG(9)=2
         CALL TDISPL
       ENDIF
880    CALL BASDIS
       RETURN
       END
```

```
C ***************************************
C  MODULE 2 UNIT23                       *
C                                        *
C  SUBROUTINE TCAL                       *
C  USE: OPENS OUTPUT TO CONSOLE TO DISPLAY CURRENT TABLEAU ON SCREEN. *
C  CALLED BY: SUBROUTINE OPT             *
C  CALLS    : SUBROUTINE TDISPL          *
C  VARIABLES:                            *
C       USED: NONE                       *
C   MODIFIED: IFLAG(3),IFLAG(9)          *
C ***************************************
      SUBROUTINE TCAL
      INTEGER ARTV,CB
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      IFLAG(9)=2
      IFLAG(3)=1
      OPEN(2,FILE='CONSOLE:')
      WRITE(2,'(A)')CHAR(12)
      CALL TDISPL
      RETURN
      END
```

```
C ***************************************
C  MODULE 2 UNIT 24                                                  *
C    UNIT $USES: UNIT25 AND UNIT27                                   *
C                                                                    *
C  SUBROUTINE PIVOT                                                  *
C  USE: DEPENDENT UPON USER OPTION SELECTION, EITHER SOLICITS INPUT  *
C       OF USER SELECTED PIVOT COLUMN AND ROW OR PASSES CONTROL TO   *
C       SUBROUTINE WORK USING ALGORITHM SELECTED PIVOT ELEMENT.      *
C       PROVIDES FEEDBACK, IF OPTION SELECTED, AND ALERTS USER OF    *
C       PIVOT ELEMENT SELECTION WITH VALUE OF APPROXIMATELY ZERO.    *
C       WITH THE DUAL PIVOT OPTION, USER MUST SELECT PIVOT ELEMENTS. *
C       ROUTINE WILL NOT ALLOW FURTHER PIVOTS UPON REACHING BOTH     *
C       PRIMAL AND DUAL OPTIMALITY CONDITIONS.                       *
C  CALLED BY: PROGRAM EDUC                                           *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                    *
C             SUBROUTINE OVER(RES)                                   *
C             SUBROUTINE WORK                                        *
C  VARIABLES:                                                        *
C       USED: A(*,*),C(*),DUAL,FMT,INEW,INVAL,K,PES,RES,VT,XB(*)     *
C   MODIFIED: IFLAG(4),IFLAG(6),IFLAG(10),L,M,P(*),PK,PKS,PR,PRS,    *
C             RATIO,SPR,TIE                                          *
C ***************************************
$USES UCHECK2 IN UNIT27.CODE OVERLAY
$USES UHEADER IN UNIT25.CODE OVERLAY
      SUBROUTINE PIVOT
     . CHARACTER VN*5,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,DIU,DUAL,OUTP,
     .TIE,FMT,RES,ASK
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,DIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(20),CN(20),PN,MN,FN,PINEQ(20),P(10),OBJN
      L=1
      M=1
C     VARIABLES IPP,IPK CONTAIN ALGORITHM SELECTED PIVOT ELEMENT
      IFLAG(4)=PK
      IFLAG(6)=PR
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      PK=IFLAG(4)
      PR=IFLAG(6)
      IF(PES .EQ. 3)THEN
C       USER HAS ELECTED ALGORITHM TO SELECT PIVOT ELEMENT
        CALL WORK
        RETURN
      ENDIF
      IF(L .EQ. 0 .AND. M .EQ. 0)THEN
C       NO NEGATIVE RHS OR Z(J)-C(J) ELEMENTS EXIST
        WRITE(1,110)CHAR(12)
        WRITE(1,'(7/),''YOU ARE UNABLE TO PROPERLY PERFORM DUAL''/''OR
     . PRIMAL PIVOTS ON THE CURRENT TABLEAU.''//''TO OVERRIDE, ENTER DIU
```

```
     .AL OR P)RIMAL TO CONTINUE. ANY OTHER ENTRY WILL TERMINATE''/
     .''PROBLEM.''//18X,''OPTION ? '',$)')
        READ(5,'(A1)')P(1)
        WRITE(1,110)CHAR(12)
        IF(ICHAR(P(1)) .EQ. 80)THEN
          GO TO 155
        ELSEIF(ICHAR(P(1)) .EQ. 68)THEN
          GO TO 1025
        ELSE
C       FLAG DENOTES FURTHER PIVOTS NOT POSSIBLE
          IFLAG(10)=5
          RETURN
        ENDIF
      ENDIF
      IF(DUAL .EQ. 2)THEN
C       USER HAS ELECTED TO ALLOW DUAL PIVOTS
        WRITE(1,'(6(/),3X,''WHICH METHOD DO YOU WISH TO USE IN''/9X,''PE
     .RFORMING THIS PIVOT?''//5X,''1. PRIMAL''//5X,''2. DUAL''
     .//5X,''3. NO FURTHER PIVOTS DESIRED''//)')
        WRITE(1,240)
        READ(5,'(A1)')P(1)
        CALL CHECK2(P,1,3,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,130)
130       FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
          GO TO 100
        ENDIF
        IF(INEW .EQ. 2)THEN
C         USER HAS ELECTED TO ATTEMPT DUAL PIVOT
          GO TO 1000
        ELSEIF(INEW .EQ. 3)THEN
          IFLAG(10)=5
          RETURN
        ENDIF
      ENDIF
      L=0
      DO 140 J=1,VT
        IF(C(J) .LT. 0.0)THEN
          L=1
        ENDIF
140   CONTINUE
      IF(L .EQ. 0)THEN
        WRITE(1,110)CHAR(12)
        WRITE(1,'(10(/),1X,''TO PERFORM PRIMAL PIVOTS, AT LEAST ONE''/3X
     ..''C(J) MUST BE NEGATIVE. THIS IS NOT''/2X,''PRESENT SO A PRIMAL P
     .IVOT CAN NOT BE''/17X,''DONE.''//)')
        PAUSE
        GO TO 100
      ENDIF
      L=1
      DO 150 I=1,K
        IF(XB(I) .LT. 0.0)THEN
```

```
          L=0
          WRITE(1,110)CHAR(12)
          WRITE(1,'(10(/),''TO PERFORM PRIMAL PIVOTS, ALL RHS VALUES''/
     .''MUST BE POSITIVE. THIS IS NOT PRESENT SO''/4X,''A PRIMAL PIVOT C
     .AN NOT BE DONE.'',//)')
          PAUSE
          GO TO 100
        ENDIF
150   CONTINUE
      WRITE(1,110)CHAR(12)
      WRITE(1,160)
160   FORMAT(9(/),2X,'WHICH COLUMN CONTAINS THE CANDIDATE'/11X,'ENTERING
     . VARIABLE?',/)
170   WRITE(1,180)
180   FORMAT(15X,'COLUMN = ',$)
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,VT,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 170
      ELSE
        PKS=INEW
      ENDIF
      IF(PES .EQ. 2)THEN
C       USER HAS ELECTED TO SELECT PIVOT WITHOUT CHECK
        PK=PKS
        GO TO 300
      ENDIF
      WRITE(1,110)CHAR(12)
C     USER SELECTION CHECKED FOR CORRECTNESS AND FEEDBACK PROVIDED
      IF(PK .EQ. PKS)THEN
        WRITE(1,190)
190     FORMAT(11(/),1X,'YOUR PIVOT COLUMN SELECTION MATCHES THE'/10X,'A
     .LGORITHM SELECTION.',8(/))
        PAUSE
      ELSE
        IF(TIE .EQ. 1)THEN
C         TIE FOR PIVOT COLUMN EXISTS
          IF(C(PK)+.0001 .GE. C(PKS) .AND. C(PK)-.0001 .LE. C(PKS))THEN
C           CHECKS IF USER SELECTION ONE OF TIES
            WRITE(1,190)
            PK=PKS
            PAUSE
          ENDIF
        ELSE
          WRITE(1,200)
200       FORMAT(5(/),'YOUR SELECTION OF PIVOT COLUMN DOES NOT'/6X,'MATC
     .H THAT OF THE ALGORITHM.'//2X,'WHICH SELECTION DO YOU WISH TO USE?
     .')
          WRITE(1,220)PKS,PK
220       FORMAT(/3X,'1. YOUR SELECTION  COLUMN = ',I2//16X,'OR'//3X,'2.
     . ALGORITHM SELECTION  COLUMN = ',I2)
```

```
230       WRITE(1,240)
240       FORMAT(//13X,'WHICH OPTION? ',$)
          READ(5,'(A1)')P(1)
          CALL CHECK2(P,1,2,INVAL,INEW)
          IF(INVAL .EQ. 1)THEN
            WRITE(1,130)
            GO TO 230
          ELSEIF(INEW .EQ. 1)THEN
            PK=PKS
          ENDIF
        ENDIF
      ENDIF
300   WRITE(1,110)CHAR(12)
C     RATIOS FOR PIVOT COLUMN CALCULATED AND DISPLAYED
      WRITE(1,320)PK
320   FORMAT(10X,'RATIOS FOR COLUMN ',I2/)
      SPR=10.E8
      TIE=0
      DO 400 I=1,K
        WRITE(1,'(10X,''ROW '',I2,'' = '',$)')I
        IF(A(I,PK) .LE. .0001)THEN
          IF(A(I,PK) .GE. -.0001)THEN
            IF(XB(I) .LE. .0001 .AND. XB(I) .GE. -.0001)THEN
              WRITE(1,'(''0.0'')')
            ELSE
              WRITE(1,'(''INFINITE'')')
            ENDIF
          ELSE
            WRITE(1,'(''NEGATIVE RATIO'')')
          ENDIF
        ELSEIF((XB(I)/A(I,PK)) .GE. 10.E6)THEN
          WRITE(1,'(''INFINITE'')')
        ELSE
          RATIO=(XB(I)/A(I,PK))
          IF(FMT .EQ. 0)THEN
            WRITE(1,'(1PE12.5)')RATIO
          ELSE
            WRITE(1,'(F12.5)')RATIO
          ENDIF
          IF(RATIO .LT. SPR)THEN
            TIE=0
            SPR=RATIO
            PR=I
          ELSEIF(RATIO+.0001 .GE. SPR .AND. RATIO-.0001 .LE. SPR)THEN
            TIE=1
          ENDIF
        ENDIF
400   CONTINUE
C     USER SELECTS PIVOT ROW
      WRITE(1,410)
410   FORMAT(/4X,'WHICH ROW CONTAINS THE CANDIDATE'/11X,'LEAVING VAR
     .IABLE?')
```

```
420   WRITE(1,430)
430   FORMAT(16X,'ROW = ',$)
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,K,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 420
      ELSE
        PRS=INEW
      ENDIF
      IF(PES .EQ. 2)THEN
        PR=PRS
        GO TO 700
      ENDIF
      WRITE(1,110)CHAR(12)
C     USER SELECTION CHECKED AND FEEDBACK PROVIDED
      IF(PR .EQ. PRS)THEN
        WRITE(1,450)
450     FORMAT(11(/),1X,'YOUR PIVOT ROW SELECTION MATCHES THE'/10X,'ALGO
     .RITHM SELECTION.',7(/))
        PAUSE
      ELSEIF(TIE .EQ. 1)THEN
        IF(((XB(PR)/A(PR,PK))+.0001) .GE. (XB(PRS)/A(PRS,PK)) .AND.
     .((XB(PR)/A(PR,PK))-.0001) .LE. (XB(PRS)/A(PRS,PK)))THEN
          WRITE(1,450)
          PR=PRS
          PAUSE
        ENDIF
      ELSE
        WRITE(1,470)
470     FORMAT(5(/),2X,'YOUR SELECTION OF PIVOT ROW DOES NOT'/6X,'MATCH
     . THAT OF THE ALGORITHM.'//2X,'WHICH SELECTION DO YOU WISH TO USE?'
     .)
        WRITE(1,490)PRS,PR
490     FORMAT(/4X,'1. YOUR SELECTION  ROW = ',I2//15X,'OR'//4X,'2. ALGO
     .RITHM SELECTION  ROW = ',I2)
500     WRITE(1,240)
        READ(5,'(A1)')P(1)
        CALL CHECK2(P,1,2,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,130)
          GO TO 500
        ELSEIF(INEW .EQ. 1)THEN
          PR=PRS
        ENDIF
      ENDIF
C     PIVOT ELEMENT CHECKED TO INSURE NOT ZERO
700   IF(A(PR,PK) .LT. .0001 .AND. A(PR,PK) .GT. -.0001)THEN
C     USER GIVEN OPTION TO CONTINUE WITH ZERO PIVOT ELEMENT
        CALL OVER(RES)
        IF(RES .EQ. 0)THEN
          GO TO 100
```

```
        ENDIF
      ENDIF
      CALL WORK
      RETURN
C     DUAL PIVOT ELEMENT DETERMINED
1000  WRITE(1,110)CHAR(12)
      DO 1010 J=1,VT
        IF(C(J) .LT. 0.0)THEN
          WRITE(1,'(10(/),''TO PERFORM DUAL PIVOTS, ALL C(J) MUST''/''BE
     . POSITIVE. THIS IS NOT PRESENT AT THIS''/''TIME SO A DUAL PIVOT CA
     .N NOT BE DONE.'',///)')
          M=0
          PAUSE
          GO TO 100
        ENDIF
1010  CONTINUE
      M=0
      DO 1020 I=1,K
        IF(XB(I) .LT. 0.0)THEN
          M=1
        ENDIF
1020  CONTINUE
      IF(M .EQ. 0)THEN
        WRITE(1,'(10(/),''TO PERFORM DUAL PIVOTS, AT LEAST ONE RHS''/ ''
     .MUST BE NEGATIVE. THIS IS NOT PRESENT AT''/''THIS TIME SO A DUAL P
     .IVOT WILL NOT BE''/''DONE.'',///)')
        PAUSE
        GO TO 100
      ENDIF
C     USER SELECTS PIVOT ROW
      WRITE(1,410)
1030  WRITE(1,430)
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,K,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 1030
      ELSE
        PRS=INEW
      ENDIF
      IF(PES .EQ. 2)THEN
        PR=PRS
        GO TO 1300
      ENDIF
C     ALGORITHM SELECTS PIVOT ROW
      GNEG=0.0
      DO 1050 I=1,K
        IF(XB(I) .GT. GNEG)THEN
          GO TO 1050
        ELSEIF(XB(I) .EQ. GNEG)THEN
          TIE=1
        ELSE
```

```
         TIE=0
         GNEG=XB(I)
         PR=I
       ENDIF
1050  CONTINUE
C     USER SELECTION CHECKED AND FEEDBACK PROVIDED
      WRITE(1,110)CHAR(12)
      IF(PR .EQ. PRS)THEN
        WRITE(1,450)
        PAUSE
      ELSE
        WRITE(1,470)
        WRITE(1,490)PRS,PR
1070    WRITE(1,240)
        READ(5,'(A1)')P(1)
        CALL CHECK2(P,1,2,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,130)
          GO TO 1070
        ELSEIF(INEW .EQ. 1)THEN
          PR=PRS
        ENDIF
      ENDIF
1300  WRITE(1,110)CHAR(12)
C     RATIOS FOR ROW CALCULATED
      WRITE(1,1320)PR
1320  FORMAT(11X,'RATIOS FOR ROW ',I2/)
      SPR=-10.E8
      DO 1400 J=1,VT
        DO 1350 I=1,K
          IF(CB(I) .EQ. J)THEN
            GO TO 1400
          ENDIF
1350    CONTINUE
        WRITE(1,'(9X,''COLUMN '',I2,'' = '',$)')J
        IF(A(PR,J) .GE. -.0001)THEN
          IF(A(PR,J) .LE. .0001)THEN
            IF(C(J) .LE. .0001 .AND. C(J) .GE. -.0001)THEN
              WRITE(1,'(''0.0'')')
            ELSE
              WRITE(1,'(''INFINITE'')')
            ENDIF
          ENDIF
        ELSEIF((C(J)/A(PR,J)) .LE. -10.E6)THEN
          WRITE(1,'(''NEGATIVE INFINITE'')')
        ELSE
          RATIO=(C(J)/A(PR,J))
          IF(FMT .EQ. 0)THEN
            WRITE(1,'(1PE12.5)')RATIO
          ELSE
            WRITE(1,'(F12.5)')RATIO
          ENDIF
```

```
          IF(RATIO .GE. SPR)THEN
            PK=J
            SPR=RATIO
          ENDIF
        ENDIF
1400  CONTINUE
      PAUSE
      WRITE(1,110)CHAR(12)
C     USER SELECTS PIVOT COLUMN
      WRITE(1,160)
1420  WRITE(1,180)
      READ(5,'(2A1)')P(1),P(2)
      CALL CHECK2(P,2,VT,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,130)
        GO TO 1420
      ELSE
        PKS=INEW
      ENDIF
      IF(PES .EQ. 2)THEN
        PK=PKS
        GO TO 1700
      ENDIF
      WRITE(1,110)CHAR(12)
C     USER SELECTION CHECKED AND FEEDBACK PROVIDED
      IF(PK .EQ. PKS)THEN
        WRITE(1,190)
        PAUSE
      ELSE
        WRITE(1,200)
        WRITE(1,220)PKS,PK
1440    WRITE(1,240)
        READ(5,'(A1)')P(1)
        CALL CHECK2(P,1,2,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,130)
          GO TO 1440
        ELSEIF(INEW .EQ. 1)THEN
          PK=PKS
        ENDIF
      ENDIF
      IF(A(PR,PK) .LT. .0001 .AND. A(PR,PK) .GT. -.0001)THEN
C       USER GIVEN OPTION TO CONTINUE WITH PIVOT ELEMENT OF ZERO VALUE
        CALL OVER(RES)
        IF(RES .EQ. 0)THEN
          GO TO 100
        ENDIF
      ENDIF
1700  CALL WORK
      RETURN
      END
```

```
C ***************************************
C  MODULE 2 UNIT24                                                  *
C                                                                   *
C  SUBROUTINE WORK                                                  *
C  USE: PERFORMS SIMPLEX PIVOT USING DESIGNATED PIVOT ELEMENT, NO   *
C        USER INTERFACE.                                            *
C  CALLED BY: SUBROUTINE PIVOT                                      *
C  CALLS    : NONE                                                  *
C  VARIABLES:                                                       *
C        USED: K,PK,PR,VT                                           *
C   MODIFIED: A(*,*),C(*),CB(*),HOLD,PELE,XB(*),Z                   *
C ***************************************
      SUBROUTINE WORK
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,CIU,DUAL,OUTP
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     ..NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      PELE=A(PR,PK)
      DO 200 J=1,VT
        A(PR,J)=A(PR,J)/PELE
200   CONTINUE
      XB(PR)=XB(PR)/PELE
      CB(PR)=PK
      DO 300 I=1,K
        IF(I .EQ. PR)THEN
          GO TO 300
        ENDIF
        HOLD=A(I,PK)
        DO 250 J=1,VT
          A(I,J)=A(I,J)-HOLD*A(PR,J)
250     CONTINUE
        XB(I)=XB(I)-HOLD*XB(PR)
300   CONTINUE
      HOLD=C(PK)
      DO 350 J=1,VT
        C(J)=C(J)-HOLD*A(PR,J)
350   CONTINUE
      Z=Z-HOLD*XB(PR)
      RETURN
      END
```

```
C ****************************************
C  MODULE 2 UNIT24                                             *
C                                                              *
C  SUBROUTINE OVER(RES)                                        *
C  USE: DISPLAYS STATEMENT THAT PIVOT ELEMENT IS APPROXIMATELY ZERO  *
C       AND PERFORMANCE OF PIVOT MAY RESULT IN SYSTEM OVERFLOW ERROR. *
C       SOLICITS RESPONSE AS WHETHER TO CONTINUE WITH PIVOT AND SETS  *
C       FLAGS TO REFLECT THIS RESPONSE.                        *
C  CALLED BY: SUBROUTINE PIVOT                                 *
C  CALLS    : NONE                                             *
C  VARIABLES:                                                  *
C       USED: NONE                                             *
C   MODIFIED: P(*).RES                                         *
C ****************************************
      SUBROUTINE OVER(RES)
      CHARACTER P*1
      INTEGER RES
      RES=0
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,150)
150   FORMAT(8(/),3X,'THE PIVOT ELEMENT SELECTED IS NOT'/2X,'SIGNIFICANT
     .LY DIFFERENT FROM ZERO.'//3X,'THIS MAY CAUSE AN OVERFLOW ERROR'/
     .7X,'IF THE PIVOT IS PERFORMED.')
160   WRITE(1,170)
170   FORMAT(/7X,'DO YOU WISH TO CONTINUE? ',$)
      READ(5,'(A1)')P
      IF(ICHAR(P) .EQ. 89)THEN
        RES=1
        RETURN
      ELSEIF(ICHAR(P) .NE. 78)THEN
        WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER'')')
        GO TO 160
      ENDIF
      RETURN
      END
```

```
C ***************************************
C  MODULE 2 UNIT25                       *
C    UNIT $USES: NONE                    *
C                                        *
C  SUBROUTINE HEADER                     *
C  USE: DISPLAYS TITLE PAGE OF MODULE 2, EDUCATIONAL MODULE.   *
C  CALLED BY: PROGRAM EDUC               *
C  CALLS    : NONE                       *
C  VARIABLES: NONE                       *
C ***************************************
      SUBROUTINE HEADER
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(4(/),9X,22(''*'')/9X,''*'',20X,''*''/9X,''*'',7X,''LINEA
     .R'',7X,''*''/9X,''*'',20X,''*''/9X,''*'',4X,''PROGRAMMING''.5X,''*
     .''/9X,''*''.20X,''*''/9X,''*'',4X,''EDUCATIONAL''.5X,''*'')')
      WRITE(1,'(9X,''*'',20X,''*''/9X,''*'',7X,''MODULE'',7X,''*''/9X,''
     .*'',20X,''*''/9X,''*'',20X,''*''/9X,''*'',7X,''MODULE 2'',5X,''*''
     ./9X,''*'',20X,''*''/9X,22(''*''),3(/))')
      PAUSE
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 2 UNIT25                                                       *
C                                                                        *
C  SUBROUTINE ASKQ(ASK)                                                  *
C  USE: ROUTINE PROMPTS USER TO RESPOND WHETHER OR NOT FINAL             *
C       SOLUTION IS TO BE SAVED TO DISK FOR SENSITIVITY ANALYSIS.  IF    *
C       USER REQUESTS FILE TO BE SAVED, A VOLUME:FILENAME IS             *
C       REQUESTED.  THIS FILE IS OPENED AND FORMATTED SO AS TO BE        *
C       COMPATIBLE WITH MODULE 4 INPUT REQUIREMENTS.  ALSO, THIS         *
C       FILENAME IS WRITTEN TO DISK IN THE DATA FILE LP2:LPDATAW         *
C       FOR TRANSITION TO MODULE 4.  USER IS NEXT PROMPTED TO            *
C       DETERMINE WHETHER OR NOT ANOTHER MODEL IS TO BE STUDIED WITH     *
C       THIS MODULE.  IF NO OTHER MODEL IS TO BE STUDIED, MODULE 2 IS    *
C       TERMINATED WITH INSTRUCTIONS ON ENTERING MODULE 1.               *
C  CALLED BY: PROGRAM EDUC                                               *
C  CALLS    : NONE                                                       *
C  VARIABLES:                                                            *
C       USED: A(*,*),B(*),C(*),CB(*),CN(*),CO(*),IFLAG(5),INDEXE,        *
C             INEQ(*),K,MM,MXMN,NEC,NGC,NLC,OBJN,PINEQ(*),PN,V,VT,       *
C             VN(*),XB(*)                                                *
C   MODIFIED: AO(*,*),ASK,FN,FNO,IFLAG(1),IFLAG(10),P(*)                 *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE ASKQ(ASK)
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10,FNO*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,ASK
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      DIMENSION AO(20,20),B(20),CO(20)
      FNO=FN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      ASK=0
      IFLAG(10)=0
      WRITE(1,'(8(/),''TO PERFORM SENSITIVITY ANALYSIS ON THIS''/''MODEL
     ., THE INFORMATION OF THE CURRENT''/''TABLEAU MUST BE SAVED TO DISK
     ..''//)')
130   WRITE(1,'(''DO YOU WISH TO SAVE THIS FILE TO DISK? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,110)CHAR(12)
        WRITE(1,'(//9X,''SAVE LP MODEL TO DISK''///2X,''ENTER THE DISK D
     .RIVE NUMBER AND FILE''/2X,''NAME YOU WANT THE CURRENT TABLEAU OF''
     ./3X,A20,'' SAVED UNDER.''/)')FN
        WRITE(1,'(/8X,''ENTER EXACTLY AS FOLLOWS''/10X,''DISK DRIVE:FILE
     .NAME''//12X,''EG.   #4:FILENAM''//''THE DRIVE:FILENAME MUST BE 10
     . CHARACTERS''/15X,''OR LESS''//1X,''DO NOT USE THE SAME NAME USED
     . WHEN THE''/6X,''ORIGINAL MODEL WAS ENTERED.'')')
        WRITE(1,'(/7X,''DISK:FILENAME = '',$)')
        READ(5,'(A10)')FN
150     WRITE(1,'(/7X,''ARE CORRECTIONS NEEDED? '',$)')
```

```
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 100
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER'')')
        GO TO 150
      ENDIF
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/)
     .)')
      PAUSE
C     TRANSFER FILE OPENED AND FILE NAME WRITTEN
      OPEN(3,FILE='LP2:LPDATAW',STATUS='OLD',FORM='UNFORMATTED')
      WRITE(3)FN
C     TRANSFER FILE CLOSED
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
C     USER PROMPTED TO INSERT DISK WHICH SOLVED MODEL IS TO BE SAVED
      WRITE(1,'(9(/),2X,''INSURE THE DISK TO CONTAIN THE FILE''//15X,
     .A10//13X,''IS AVAILABLE.'',7(/))')FN
      PAUSE
      WRITE(1,110)CHAR(12)
C     CURRENT STATUS OF FILE INPUT BY USER
      WRITE(1,'(9(/),''HAS THIS DISK:FILENAME COMBINATION BEEN''/12X,
     .''USED PREVIOUSLY?''//''(ARE YOU UPDATING A CURRENTLY EXISTING''/
     .17X,''FILE?)'')')
200   WRITE(1,'(//16X,''(Y/N) '',$)')
      READ(5,'(A1)')P(1)
C     FILE OF STATUS DESIGNATED BY USER OPENED
      IF(ICHAR(P(1)) .EQ. 89)THEN
        OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        OPEN(3,FILE=FN,STATUS='NEW',FORM='UNFORMATTED')
      ELSE
        WRITE(1,210)
210     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 200
      ENDIF
C     SOLVED MODEL WRITTEN TO DISK
      WRITE(3)FN,MXMN,K,V,IFLAG(5)
      WRITE(1,110)CHAR(12)
      WRITE(1,'(9(/),5X,''INSURE THE DISK CONTAINING THE''//15X,A10
     .//10X,''MODEL IS AVAILABLE.'',7(/))')FNO
      PAUSE
C     ORIGINAL MODEL FILE OPENED TO READ ORIGINAL PARAMETERS
      OPEN(4,FILE=FNO,STATUS='OLD',FORM='UNFORMATTED')
      READ(4)FN,MXMN,MM,K,V,NEC,NGC,NLC
      DO 220 I=1,10
        READ(4)IFLAG(I)
220   CONTINUE
      DO 240 I=1,K
        READ(4)INEQ(I),PINEQ(I),B(I)
```

```
      DO 230 J=1,V
        READ(4)AO(I,J)
230   CONTINUE
240   CONTINUE
      DO 250 J=1,V
        READ(4)CO(J)
250   CONTINUE
      CLOSE(4,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(9(/),''INSURE THE DISK TO CONTAIN THE FILE''//15X,A10
     .//13X,''IS AVAILABLE.'',7(/))')FN
      PAUSE
      DO 270 I=1,K
        WRITE(3)INEQ(I),B(I)
        DO 260 J=1,V
          WRITE(3)AO(I,J)
260     CONTINUE
270   CONTINUE
C     SOLVED MODEL WITH ORIGINAL PARAMETERS WRITTEN TO FILE
      DO 275 J=1,V
        WRITE(3)CO(J)
275   CONTINUE
      IFLAG(10)=0
      WRITE(3)IFLAG(10),VT
      DO 290 I=1,K
        WRITE(3)XB(I),CB(I)
        DO 280 J=1,VT
          WRITE(3)A(I,J)
280     CONTINUE
290   CONTINUE
      DO 300 J=1,VT
        WRITE(3)C(J)
300   CONTINUE
      WRITE(3)Z
      IF(IFLAG(5) .EQ. 1)THEN
        DO 310 I=1,K
          WRITE(3)CN(I)
310     CONTINUE
        DO 320 J=1,V
          WRITE(3)VN(J)
320     CONTINUE
        WRITE(3)OBJN
      ENDIF
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/)
     .)')
      PAUSE
     ELSEIF(ICHAR(P(1)) .NE. 78)THEN
      WRITE(1,210)
      GO TO 130
     ENDIF
```

```
390   WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''WOULD YOU LIKE TO STUDY ANOTHER MODEL''/4X,''
     .WHICH HAS BEEN SAVED TO DISK? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
400     WRITE(1,110)CHAR(12)
        WRITE(1,'(9(/),2X,''ENTER DISK DRIVE NUMBER AND FILENAME''/4X,''
     .WHICH THE MODEL IS SAVED UNDER.'')')
        WRITE(1,'(/6X,''MODEL TO STUDY = '',$)')
        READ(5,'(A10)')FN
450     WRITE(1,'(//7X,''ARE CORRECTIONS NEEDED? '',$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          GO TO 400
        ELSEIF(ICHAR(P(1)) .NE. 78)THEN
          WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER'')')
          GO TO 450
        ENDIF
        WRITE(1,110)CHAR(12)
        WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/)
     .)')
        PAUSE
C       TRANSFER FILE OPENED AND NEW MODEL FILE NAME WRITTEN
        OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
        WRITE(3)FN
        CLOSE(3,STATUS='KEEP')
        ASK=1
        RETURN
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER'')')
        GO TO 390
      ENDIF
      WRITE(1,110)CAHR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/))')
      PAUSE
      WRITE(1,110)CHAR(12)
      WRITE(1,'(8(/),1X,''TO ENTER THE LP DATABASE MODULE:''//17X,''TYPE
     .''//19X,''X''/11X,''LP1:SYSTEM.STARTUP.'',3(/))')
      STOP
      RETURN
      END
```

```
C ************************************************************
C  MODULE 2 UNIT25                                           *
C                                                            *
C  SUBROUTINE QUESTN                                         *
C  USE: DISPLAYS INSTRUCTIONS TO USER NOTING THAT TABLEAU SHOULD BE  *
C       STUDIED TO ALLOW USER TO ANSWER IDENTIFIED QUESTIONS WHICH   *
C       FOLLOW.  ONLY DISPLAYED IF USER IS SELECTING PIVOT ELEMENT   *
C       AND OUTPUT IS TO SCREEN.                             *
C  CALLED BY: PROGRAM EDUC                                   *
C  CALLS    : NONE                                           *
C  VARIABLES: NONE                                           *
C ************************************************************
      SUBROUTINE QUESTN
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(2(/),1X,''FROM THIS POINT ON, YOU WILL BE ASKED''/9X,''Q
     .UESTIONS CONCERNING:''//''1. PIVOT COLUMN SELECTIONS''//''2. PIVOT
     . ROW SELECTIONS''/)')
      WRITE(1,'(''3. DUAL PIVOTS(IF OPTION SELECTED)''//''4. OPTIMALITY,
     . FEASIBILITY, BOUNDEDNESS''/3X''(IF OPTION SELECTED)'')')
      WRITE(1,'(//''BE SURE TO EXAMINE THE SCREEN OUTPUT''/''OF THE TABL
     .EAUS CAREFULLY BEFORE''/''CONTINUING SO YOU MAY ANSWER QUESTIONS''
     ./''AS NOTED ABOVE.''/)')
      PAUSE
      RETURN
      END
```

```
C ***************************************
C  MODULE 2 UNIT25                                                  *
C                                                                   *
C  SUBROUTINE BIGM                                                  *
C  USE: PERFORMS CALCULATIONS TO ACQUIRE A INITIAL BASIC SOLUTION   *
C       WHEN ARTIFICAL VARIABLES HAVE BEEN ADDED TO MODEL.  NO USER *
C       INTERFACE.                                                  *
C  CALLED BY: PROGRAM EDUC                                          *
C  CALLS    : NONE                                                  *
C  VARIABLES:                                                       *
C       USED: A(*,*),ARTV(*),BM,KFA,NGC,V,VT,XB(*)                  *
C   MODIFIED: C(*),KFSU,IA,M.SUM,Z                                  *
C ***************************************
      SUBROUTINE BIGM
      INTEGER ARTV,BASIC,FK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      IA=IA-1
      KFSU=V+NGC
      DO 300 J=1,KFSU
        SUM=0.0
        DO 200 I=1,IA
          SUM=SUM+A(ARTV(I),J)
200     CONTINUE
        C(J)=C(J)-(BM*SUM)
300   CONTINUE
      DO 400 J=KFA,VT
        C(J)=0.0
400   CONTINUE
      SUM=0.0
      DO 500 I=1,IA
        SUM=SUM+XB(ARTV(I))
500   CONTINUE
      Z=Z-(BM*SUM)
      RETURN
      END
```

```
C **************************************
C  MODULE 2 UNIT 25                                                     *
C                                                                       *
C  SUBROUTINE INDEX                                                     *
C  USE: DETERMINES THE COLUMN POSITION OF ADDED SLACK, SURPLUS, AND     *
C       ARTIFICIAL VARIABLES.  PLACES THE APPROPRIATE COEFFICIENT IN    *
C       THE A(*,*) ARRAY FOR EACH AND IDENTIFIES THE INITIAL BASIC      *
C       VARIABLE.  CHANGES ALL INEQUALITIES TO EQUALITIES IN PINEQ(*)   *
C       AND NAMES THE ADDED VARIABLES FOR NAMED MODELS.                 *
C  CALLED BY: PROGRAM EDUC                                              *
C  CALLS    : NONE                                                      *
C  VARIABLES:                                                           *
C       USED: IFLAG(5),INEQ(*),K,NEC,NGC,NLC,V                          *
C   MODIFIED: A(*,*),ARTV(*),CB(*),CN(*),IA,INDEXE,INDEXG,INDEXL,KFA,   *
C             KFS,KFSA,KFSU,PINEQ(*),VN(*),VT                           *
C **************************************
      SUBROUTINE INDEX
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      IA=1
      KFSA=V+1
      VT=V+(2*NGC)+NLC+NEC
      KFS=V+NGC+1
      KFA=KFS+NLC
      KFSU=V+NGC
      INDEXG=V+1
      INDEXL=V+NGC+1
      INDEXE=V+NGC+NLC+1
      DO 200 I=1,K
        IF(INEQ(I) .EQ. 0)THEN
C         SLACK VARIABLE ADDED TO CONSTRAINT
          CB(I)=INDEXL
          A(I,INDEXL)=1.
          INDEXL=INDEXL+1
        ELSEIF(INEQ(I) .EQ. 1)THEN
C         SURPLUS AND ARTIFICAL VARIABLE ADDED TO CONSTRAINT
          CB(I)=INDEXE
          ARTV(IA)=1
          IA=IA+1
          A(I,INDEXE)=1.
          INDEXE=INDEXE+1
          A(I,INDEXG)=-1.
          INDEXG=INDEXG+1
        ELSE
C         ARTIFICAL VARIABLE ADDED TO CONTRAINT
          ARTV(IA)=I
          IA=IA+1
          CB(I)=INDEXE
```

```
          A(I,INDEXE)=1.
          INDEXE=INDEXE+1
        ENDIF
      PINEQ(I)='='
200   CONTINUE
      IF(IFLAG(5) .EQ. 1)THEN
        DO 210 J=KFSA,KFS-1
          VN(J)='SURPLS'
210     CONTINUE
        DO 220 J=KFS,KFA-1
          VN(J)='SLACK'
220     CONTINUE
        DO 230 J=KFA,VT
          VN(J)='ARTIF'
230     CONTINUE
      ENDIF
      RETURN
      END
```

```
C ****************************************
C  MODULE 2 UNIT25                                              *
C                                                               *
C  SUBROUTINE MODIFA                                            *
C  USE: PERFORMS THE SAME FUNCTION AS SUBROUTINE OBMDU AND SUBROUTINE *
C       CNMDV EXCEPT NO USER INTERFACE.  NOTE THAT CONSTRAINTS WITH  *
C       NEGATIVE RHS'S ARE MULTIPLIED BY -1.                    *
C  CALLED BY: PROGRAM EDUC                                      *
C  CALLS    : NONE                                              *
C  VARIABLES:                                                   *
C       USED: K,MM,MXMN,V                                       *
C   MODIFIED: A(*,*),C(*),INEQ(*),NGC,NLC,PINEQ(*),XB(*)        *
C ****************************************
      SUBROUTINE MODIFA
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
      IF(MXMN .EQ. 1)THEN
C        PROBLEM STATED AS MAXIMIZATION
         DO 160 J=1,V
           C(J)=-C(J)
160      CONTINUE
      ELSE
        MM='MAX'
      ENDIF
      DO 300 I=1,K
        IF(XB(I) .LT. 0.0)THEN
          XB(I)=-XB(I)
          DO 200 J=1,V
            A(I,J)=-A(I,J)
200       CONTINUE
C         COUNT OF INEQUALITIES BY TYPE UPDATED DUE TO MULT BY -1
          IF(INEQ(I) .EQ. 0)THEN
            NLC=NLC-1
            NGC=NGC+1
            INEQ(I)=1
            PINEQ(I)='>'
          ELSEIF(INEQ(I) .EQ. 1)THEN
            NGC=NGC-1
            NLC=NLC+1
            INEQ(I)=0
            PINEQ(I)='<'
          ENDIF
        ENDIF
300   CONTINUE
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 2 UNIT25                                                           *
C                                                                            *
C  SUBROUTINE INTRO                                                          *
C  USE: INITIALIZES ALL VARIABLES TO ZERO EXCEPT CHARACTER VARIABLES         *
C       AND READS MODEL DESIGNATED AS MODEL TO SOLVE FROM DISK FILE          *
C       CREATED WITH MODULE 1.  ALSO CALCULATES A VALUE TO BE USED IN        *
C       "BIG M" METHOD.                                                      *
C  CALLED BY: PROGRAM EDUC                                                   *
C  CALLS    : NONE                                                           *
C  VARIABLES:                                                                *
C       USED: NONE                                                           *
C   MODIFIED: A(*,*),ARTV(*),BASIC,BM,C(*),CB(*),CJ,CN(*),FN,IA,             *
C             IFLAG(1) THRU IFLAG(10),INDEXE,INDEXG,INDEXL,INEQ(*),K,        *
C             MM,MXMN,NEC,NGC,NLC,OBJN,OPTS,PINEQ(*),PN,V,VN(*),             *
C             XB(*),Z                                                        *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE INTRO
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     ..NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
110   FORMAT(A)
C     VARIABLES INITIALIZED
      DO 180 I=1,20
        ARTV(I)=0
        CB(I)=0
        INEQ(I)=0
        XB(I)=0.0
        DO 170 J=1,60
          A(I,J)=0.0
170     CONTINUE
180   CONTINUE
      DO 190 J=1,60
        C(J)=0.0
190   CONTINUE
      DO 200 I=1,10
        IFLAG(I)=0
200   CONTINUE
      NEC=0
      NGC=0
      NLC=0
      Z=0.0
      IA=0
      BASIC=0
      OPTS=0
      INDEXE=0
      INDEXG=0
```

```
      INDEXL=0
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/))')
      PAUSE
C     TRANSFER FILE OPENED AND FILE NAME READ
      OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
      READ(3)FN
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(9(/),5X,''INSURE THE DISK CONTAINING THE''//15X,A10
     .//10X,''MODEL IS AVAILABLE.'',7(/))')FN
      PAUSE
C     FILE WHICH CONTAINS MODEL OPENED AND READ FROM DISK
      OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
      READ(3)PN,MXMN,MM,K,V,NEC,NGC,NLC
      DO 220 I=1,10
        READ(3)IFLAG(I)
220   CONTINUE
      DO 300 I=1,K
        READ(3)INEQ(I),PINEQ(I),XB(I)
        DO 280 J=1,V
          READ(3)A(I,J)
280     CONTINUE
300   CONTINUE
      DO 320 J=1,V
        READ(3)C(J)
320   CONTINUE
      IF(IFLAG(5) .EQ. 1)THEN
        DO 350 I=1,K
          READ(3)CN(I)
350     CONTINUE
        DO 360 J=1,V
          READ(3)VN(J)
360     CONTINUE
        DO 380 J=V+1,20
          VN(J)='      '
380     CONTINUE
        READ(3)OBJN
      ENDIF
      IFLAG(2)=1
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/)')
      PAUSE
C     FIND APPPROPRIATE VALUE FOR BIG M
      CJ=0.0
      BM=0.0
      DO 400 J=1,V
        IF(ABS(C(J)) .GT. BM)THEN
          BM=ABS(C(J))
        ENDIF
400   CONTINUE
```

```
BM=(ANINT(BM))*10.0
IF(BM .LT. 1.0)THEN
  BM=10.0
ENDIF
RETURN
END
```

```
C ****************************************
C  MODULE 2 UNIT26                                                   *
C    UNIT $USES: UNIT27                                              *
C                                                                    *
C  SUBROUTINE TDISPL                                                 *
C  USE: DISPLAYS EITHER THE CURRENT CONSTRAINTS, COMPLETE LP MODEL,   *
C       OR THE CURRENT TABLEAU, DEPENDENT ON FLAGS SET IN CALLING     *
C       SUBROUTINE.  VARIABLES AND CONSTRAINTS ARE DISPLAYED WITH     *
C       NAMES, IF PRESENT, AND THE BASIC VARIABLE OF THE CONSTRAINT   *
C       IS IDENTIFIED WHEN DISPLAYING THE CURRENT TABLEAU.            *
C       CALCULATES THE NUMBER OF 80 COLUMN WIDTHS TO DISPLAY COMPLETE *
C       TABLEAU AND PRESENTS ON OUTPUT DEVICE IN SECTIONS.  NO USER   *
C       INTERFACE.                                                    *
C  CALLED BY: PROGRAM EDUC                                           *
C             SUBROUTINE CNMOU                                        *
C             SUBROUTINE OPT                                          *
C             SUBROUTINE READY                                        *
C             SUBROUTINE TCAL                                         *
C  CALLS    : NONE                                                    *
C  VARIABLES:                                                         *
C       USED: A(*,*),C(*),CB(*),CN(*),FMT,IFLAG(3),IFLAG(5),IFLAG(9), *
C             K,MM,OBJ,OUTP,PINEQ(*),PN,VN(*),VT,XB(*)                *
C   MODIFIED: NONE                                                    *
C ****************************************
$USES UCHECK2 IN UNIT27.CODE OVERLAY
      SUBROUTINE TDISPL
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     .TIE,FMT,T
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(20),CN(20),PN,MM,FN,PINEQ(20),P(10),OBJN
110   FORMAT(A)
      IF(IFLAG(9) .EQ. 1)THEN
C       ONLY CONSTRAINTS ARE DISPLAYED
        WRITE(2,220)PN
220     FORMAT(10X,A20/10X,'CURRENT CONSTRAINTS'/)
      ELSEIF(IFLAG(9) .EQ. 0)THEN
C       OBJ FUNCTION AND CONSTRAINTS DISPLAYED
        WRITE(2,230)PN,MM
230     FORMAT(10X,A20/7X,'CURRENT LP MODEL: ',A3,'IMIZE ',$)
        IF(IFLAG(5) .EQ. 1)THEN
          WRITE(2,240)OBJN
240       FORMAT(A10)
        ELSE
          WRITE(2,250)
250       FORMAT(' ')
        ENDIF
      ENDIF
C     NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED
```

```
      T=(VT/5)+1
      DO 470 N=1,T
        IF(IFLAG(5) .EQ. 1)THEN
C         VARIABLE NAMES PRINTED AS COLUMN HEADERS
          WRITE(2,'(13X,$)')
          DO 270 J=(N*5)-4,N*5
            IF(J .GT. VT)THEN
              GO TO 270
            ENDIF
            WRITE(2,260)VN(J)
260         FORMAT(5X,A6,2X,$)
270       CONTINUE
          WRITE(2,'('' '')')
        ENDIF
        WRITE(2,'(13X,$)')
        DO 290 J=(N*5)-4,N*5
          IF(J .GT. VT)THEN
            GO TO 290
          ENDIF
          WRITE(2,280)J
280       FORMAT(5X,'X(',I2,')',3X,$)
290     CONTINUE
C       IF LAST 80 COLUMN DISPLAY, DISPLAY RHS
        IF(T .EQ. 1 .OR. N .EQ. T)THEN
          WRITE(2,300)
300       FORMAT(6X,'RHS')
        ELSE
          WRITE(2,'('' '')')
        ENDIF
C       DISPLAY IS WITH OBJ FUNCTION
        IF(IFLAG(9) .EQ. 0 .OR. IFLAG(9) .EQ. 2)THEN
          WRITE(2,'(''OBJ FUNCTION'',1X,$)')
          DO 320 J=(N*5)-4,N*5
            IF(J .GT. VT)THEN
              GO TO 320
            ENDIF
            IF(FMT .EQ. 0)THEN
              WRITE(2,'(1PE12.5,1X,$)')C(J)
            ELSE
              WRITE(2,'(F12.5,1X,$)')C(J)
            ENDIF
320       CONTINUE
          IF(T .EQ.1 .OR. N .EQ. T)THEN
            IF(FMT .EQ. 0)THEN
              WRITE(2,'(''= '',1PE12.5,1X)')Z
            ELSE
                WRITE(2,'(''= '',F12.5,1X)')Z
            ENDIF
          ELSE
            WRITE(2,'('' '')')
          ENDIF
        ENDIF
```

```
      IF(IFLAG(9) .EQ. 2)THEN
C       BASIC VARIABLES ARE TO BE ANNOTATED
        WRITE(2,'(''CN  NAME  VAR'',2X,65(''$''))')
      ELSE
        WRITE(2,'(''CN  NAME'',2X,70(''$''))')
      ENDIF
C     CONSTRAINT NUMBER, NAME, BASIC VARIABLE, COEFFICIENTS,
C       INEQUALITY, AND RHS DISPLAYED
      DO 400 L=1,K
        IF(L .GT. K)THEN
          GO TO 400
        ENDIF
        IF(IFLAG(5) .EQ. 1)THEN
          WRITE(2,'(I2,1X,A6,$)')L,CN(L)
        ELSE
          WRITE(2,'(I2,7X,$)')L
        ENDIF
        IF(IFLAG(9) .EQ. 2)THEN
          WRITE(2,'(1X,I2,1X,$)')CB(L)
        ELSE
          WRITE(2,'(4X,$)')
        ENDIF
        DO 370 J=(N*5)-4,N*5
          IF(J .GT. VT)THEN
            GO TO 370
          ENDIF
          IF(FMT .EQ. 0)THEN
            WRITE(2,'(1PE12.5,1X,$)')A(L,J)
          ELSE
            WRITE(2,'(F12.5,1X,$)')A(L,J)
          ENDIF
370     CONTINUE
        IF(T .EQ. 1 .OR. N .EQ. T)THEN
          IF(FMT .EQ. 0)THEN
            WRITE(2,'(A1,1X,1PE12.5)')PINEQ(L),XB(L)
          ELSE
            WRITE(2,'(A1,1X,F12.5)')PINEQ(L),XB(L)
          ENDIF
        ELSE
          WRITE(2,'('' '')')
        ENDIF
400   CONTINUE
      IF(IFLAG(3) .EQ. 1)THEN
        PAUSE
        WRITE(2,110)CHAR(12)
        GO TO 470
      ENDIF
      IF(OUTP .EQ. 1)THEN
        PAUSE
        WRITE(2,110)CHAR(12)
      ELSE
        WRITE(2,'(2(/))')
```

```
        ENDIF
470   CONTINUE
      IFLAG(3)=0
      IFLAG(9)=0
      CLOSE(2)
      RETURN
      END
```

```
C ***************************************
C  MODULE 2 UNIT26                                                      *
C                                                                       *
C  SUBROUTINE BASDIS                                                    *
C  USE: DETERMINES WHETHER OR NOT CURRENT TABLEAU WAS REQUESTED TO BE *
C       DISPLAYED AND IF SO, PROMPTS USER WHETHER OR NOT THE BASIC     *
C       SOLUTION IS DESIRED TO BE DISPLAYED AND ON WHAT DEVICE.  IF    *
C       REQUESTED, DISPLAYS BASIC VARIABLES, NAMES AND VALUES OF       *
C       CURRENT SOLUTION AND THE OBJECTIVE FUNCTION VALUE.             *
C  CALLED BY: SUBROUTINE OPT                                           *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                      *
C  VARIABLES:                                                          *
C       USED: BASIC,CB(*),FMT,IFLAG(5),IFTAB,INEW,INVAL,ITAB,K,OPTS,   *
C             PN,VN(*),XB(*),Z                                         *
C   MODIFIED: P(*)                                                     *
C ***************************************
      SUBROUTINE BASDIS
      CHARACTER VN*6,CN*6,PN*20,MN*3,FN*10,PINEQ*1,P*1,OBJN*10
      INTEGER ARTV,BASIC,PK,FKS,PR,PRS,OPTS,V,VT,CB,PES,OIU,DUAL,OUTP,
     ,TIE,FMT
      COMMON/E1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20)
     .,NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,XB(20)
      COMMON/E2/BASIC,K,KFA,KFS,KFSA,KFSU,OPTS,PK,PKS,PR,PRS,V,VT,MXMN
      COMMON/E3/MOD,PES,OIU,DUAL,OUTP,ITAB,IBTAB,IFTAB,BM,TIE,FMT
      COMMON/E4/VN(20),CN(20),PN,MN,FN,PINEQ(20),P(10),OBJN
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
C     DETERMINES IF USER HAS SELECTED TABLEAU FOR OUTPUT
150   IF(BASIC .EQ. 1 .AND. ITAB .NE. 1)THEN
        RETURN
      ELSEIF(BASIC .NE. 1 .AND. OPTS .NE. 1 .AND. IBTAB .EQ. 0)THEN
        RETURN
      ELSEIF(OPTS .EQ. 1 .AND. IFTAB .EQ. 2)THEN
        RETURN
      ENDIF
      WRITE(1,110)CHAR(12)
C     USER INPUTS SELECTION OF OUTPUT DEVICE, IF ANY
      WRITE(1,'(6(/),''WOULD YOU LIKE THE BASIC SOLUTION VALUES''/14X,
     .''DISPLAYED? '')')
      WRITE(1,'(/9X,''1. DISPLAY ON SCREEN''//9X,''2. DISPLAY ON PRINTER
     .''//9X,''3. DO NOT DISPLAY'')')
200   WRITE(1,'(//13X,''WHICH OPTION? '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,3,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,230)
230     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 200
      ENDIF
      IF(INEW .EQ. 1)THEN
        OPEN(2,FILE='CONSOLE:')
      ELSEIF(INEW .EQ. 2)THEN
```

```
      OPEN(2,FILE='PRINTER:')
    ELSE
      RETURN
    ENDIF
    WRITE(1,110)CHAR(12)
    WRITE(2,'(10X,A20/10X,''BASIC SOLUTION #'',I2,/)')PN,BASIC
    IF(IFLAG(5) .EQ. 1)THEN
      DO 250 I=1,K
        IF(FMT .EQ. 0)THEN
          WRITE(2,'(5X,A6,'' = X('',I2,'') = '',1PE12.5)')VN(CB(I)),
   .CB(I),XB(I)
        ELSE
          WRITE(2,'(5X,A6,'' = X('',I2,'') = '',F12.5)')VN(CB(I)),
   .CB(I),XB(I)
        ENDIF
250   CONTINUE
      IF(FMT .EQ. 0)THEN
        WRITE(2,'(/18X,''Z= '',1PE12.5)')Z
      ELSE
        WRITE(2,'(/18X,''Z= '',F12.5)')Z
      ENDIF
    ELSE
      DO 280 I=1,K
        IF(FMT .EQ. 0)THEN
          WRITE(2,'(10X,''X('',I2,'') = '',1PE12.5)')CB(I),XB(I)
        ELSE
          WRITE(2,'(10X,''X('',I2,'') = '',F12.5)')CB(I),XB(I)
        ENDIF
280   CONTINUE
      IF(FMT .EQ. 0)THEN
        WRITE(2,'(/14X,''Z = '',1PE12.5)')Z
      ELSE
        WRITE(2,'(/14X,''Z = '',F12.5)')Z
      ENDIF
    ENDIF
    IF(INEW .EQ. 2)THEN
      WRITE(2,'(6(/))')
    ELSE
      PAUSE
    ENDIF
    RETURN
    END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 2 UNIT27                                                         *
C    UNIT $USES: NONE                                                      *
C                                                                          *
C  SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)                                  *
C  USE: SEE MODULE 1, UNIT17                                               *
C  CALLED BY: SUBROUTINE BASDIS                                            *
C             SUBROUTINE CNMDU                                             *
C             SUBROUTINE CBMDU                                             *
C             SUBROUTINE OPTION                                            *
C             SUBROUTINE PIVOT                                             *
C             SUBROUTINE READY                                             *
C  CALLS    : NONE                                                         *
C  VARIABLES: SEE MODULE 1, UNIT17                                         *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
      CHARACTER ALLOW*1,E*1
      DIMENSION E(10),ALLOW(11)
      INTEGER D,HVAL
      DATA ALLOW/'1','2','3','4','5','6','7','8','9','0',' '/
      INEW=0
      INVAL=0
      DO 300 I=1,D
        DO 200 J=1,10
C           CHECKS FIRST FOR BLANK CHARACTERS
          IF(E(I) .EQ. ALLOW(11))THEN
            GO TO 300
          ELSEIF(E(I) .EQ. ALLOW(J))THEN
            INEW=INEW*10 + (ICHAR(E(I))-48)
            GO TO 300
          ELSEIF(J .EQ. 10)THEN
            INVAL=1
            INEW=0
            RETURN
          ENDIF
200     CONTINUE
300   CONTINUE
      IF(INEW .EQ. 0 .OR. INEW .GT. HVAL)THEN
        INVAL=1
        INEW=0
        RETURN
      ENDIF
      RETURN
      END
```

```
C ***************************************
C  MODULE 3 UNIT30                                                    *
C    UNIT $USES: UNIT32 THRU UNIT37                                   *
C                                                                     *
C  PROGRAM PROBS                                                      *
C  USE: MAIN PROGRAM OF MODULE 3.  PURPOSE OF MODULE IS TO PROVIDE A  *
C       MEANS OF SOLVING LINEAR PROGRAMMING PROBLEM IN THE MOST       *
C       EFFICIENT SIMPLEX METHOD.  THIS MODULE ALLOWS THE USER TO     *
C       SPECIFY THE PRIMAL OR DUAL PROBLEM, AND FURTHER, PRIMAL OR    *
C       DUAL SIMPLEX APPLICATION TO THE SELECTED PROBLEM.  SELECTED   *
C       DISPLAY OF OUTPUT ALLOWS USER TO LIMIT OUTPUT TO THAT         *
C       REQUIRED.  MODULE 3 CONSISTS OF 7 SEPARATELY COMPILED UNITS   *
C       (UNIT30, UNIT32 THRU UNIT37) WITH ALL UNITS EXCEPT UNITO      *
C       BEING OVERLAY UNITS.                                          *
C       PROGRAM PROBS ACTS AS A MEMORY RELEASE LOCATION.  WHENEVER    *
C       THE PROGRAM CONTROL RETURNS TO THIS UNIT, ALL OVERLAY UNITS   *
C       ARE RELEASED FROM MEMORY PRIOR TO NEW UNITS BEING SUMMONED.   *
C  CALLED BY: NONE                                                    *
C  CALLS    : SUBROUTINE ACNCH                                        *
C             SUBROUTINE ASKQ(ASK)                                    *
C             SUBROUTINE BIGM                                         *
C             SUBROUTINE CONVRT                                       *
C             SUBROUTINE INDEX                                        *
C             SUBROUTINE INRD                                         *
C             SUBROUTINE MODIFD                                       *
C             SUBROUTINE MODIFP                                       *
C             SUBROUTINE OPTB                                         *
C             SUBROUTINE OPTN                                         *
C             SUBROUTINE PSHED                                        *
C             SUBROUTINE TDISPL                                       *
C             SUBROUTINE WORK                                         *
C  VARIABLES:                                                         *
C       USED: ASK,BM,DUAL,IFLAG(7),IFLAG(9),KFA,NEC,NGC,PROBT,VT      *
C   MODIFIED: BASIC,C(*)                                              *
C ***************************************
$USES UCHECK2 IN UNIT37.CODE OVERLAY
$USES UTDISPL IN UNIT36.CODE OVERLAY
$USES UPSHED IN UNIT35.CODE OVERLAY
$USES UCONVRT IN UNIT34.CODE OVERLAY
$USES UWORK IN UNIT33.CODE OVERLAY
$USES UOPTN IN UNIT32.CODE OVERLAY
      PROGRAM PROBS
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT,ASK
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P3/DUAL,OUTP,ITAB,IETAB,IFTAB,FMT,PROBT
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      OPEN(1,FILE='CONSOLE:')
      OPEN(5,FILE='CONSOLE:')
```

```
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      CALL PSHED
C     ROUTINE CALLED WHICH ALLOWS USER TO CHANGE DEFAULTS
120   CALL OPTN
      BASIC=0
      IF(PROBT .EQ. 1 .AND. DUAL .EQ. 1)THEN
C       USER HAS ELECTED TO SOLVE PRIMAL PROBLEM WITHOUT DUAL PIVOTS
        GO TO 140
      ELSEIF(PROBT .EQ. 2)THEN
C       USER HAS ELECTED TO SOLVE DUAL PROBLEM
        CALL CONVRT
      ENDIF
      IF(DUAL .EQ. 2)THEN
C       USER HAS ELECTED TO USE DUAL PIVOTS
        CALL ACNCH
      ENDIF
C     ROUTINE CALLED WHICH INITIALIZES VARIABLES AND READS MODEL
140   CALL INRO
      IF(DUAL .EQ. 1)THEN
C       OBJ FUNCTION MODIFIED FOR PRIMAL PROBLEM
        CALL MODIFP
      ELSE
C       OBJ FUNCTION MODIFIED FOR DUAL PROBLEM
        CALL MODIFD
      ENDIF
C     ROUTINE CALLED WHICH ADDS SLACK,SURPLUS, AND ARTIFICAL VARIABLES
      CALL INDEX
      DO 160 J=KFA,VT
        C(J)=-BM
160   CONTINUE
C     CHECKS IF ARTIFICAL VARIABLES HAVE BEEN ADDED
      IF((NEC+NGC) .NE. 0)THEN
        CALL BIGM
      ENDIF
170   BASIC=BASIC+1
      CALL OPTB
      IF(IFLAG(9) .EQ. 1)THEN
C       FLAG INDICATES TABLEAU IS TO BE DISPLAYED
        CALL TDISPL
      ENDIF
      IF((OPTS .EQ. 1) .OR. (IFLAG(7) .EQ. 1))THEN
C       LAST TABLEAU EITHER OPTIMAL OR UNBOUNDED
        CALL ASKQ(ASK)
        IF(ASK .EQ. 1)THEN
          GO TO 120
        ENDIF
      ENDIF
      CALL WORK
      GO TO 170
      STOP
      END
```

```
C ****************************************
C  MODULE 3 UNIT32                                                 *
C    UNIT $USES: UNIT37                                            *
C                                                                  *
C  SUBROUTINE OPTN                                                 *
C  USE: DISPLAYS DEFAULT OPTION VALUES AND SOLICITS RESPONSE TO    *
C       CHANGE THESE DEFAULTS.  IF OPTION IS SELECTED TO BE CHANGED, *
C       USER REVIEWS MENU AND SELECTS DESIRED METHOD, THEN IS      *
C       RETURNED TO DEFAULT OPTION DISPLAY.  SOME OPTIONS ARE CHANGED *
C       UPON SELECTION DUE TO ONLY TWO METHODS BEING POSSIBLE.     *
C       OPTIONS ARE RESET TO PROGRAMMER SPECIFIED DEFAULT UPON EACH *
C       CALL TO ROUTINE.                                           *
C  CALLED BY: PROGRAM PROBS                                        *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                  *
C  VARIABLES:                                                      *
C       USED: INEW,INVAL                                           *
C   MODIFIED: DUAL,FMT,FN,IBTAB,IFTAB,ITAB,OUTP,PN,PROBT           *
C ****************************************
$USES UCHECK2 IN UNIT37.CODE OVERLAY
      SUBROUTINE OPTN
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT
      COMMON/P3/DUAL,OUTP,ITAB,IBTAB,IFTAB,FMT,PROBT
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/))')
      PAUSE
      WRITE(1,110)CHAR(12)
      WRITE(1,130)
130   FORMAT(4X,'PROBLEM SOLVER OPTION SELECTION'/)
C     TRANSFER FILE OPENED AND FILE NAME READ
      OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
      READ(3)FN
      CLOSE(3,STATUS='KEEP')
150   WRITE(1,'(''THE PROBLEM CURRENTLY IDENTIFIED AS THE''/7X,''PROBLEM
     . TO BE STUDIED IS:''//15X,A10)')FN
      WRITE(1,'(//''IS THIS THE PROBLEM YOU DESIRE TO STUDY? ''/19X,$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 78)THEN
        WRITE(1,'(//1X,''PLEASE ENTER THE DISK DRIVE NUMBER AND''/''FILE
     . NAME OF THE FILE YOU WISH TO STUDY.''//1X,''(INSURE THIS IS ENTER
     .ED EXACTLY AS IT'')')
        WRITE(1,'(1X,''WAS SAVED PREVIOUSLY AND ALSO THAT THE''/2X,''PRO
     .PER DISK IS IN THE PROPER DRIVE.)'')')
        WRITE(1,'(/4X,''MODEL TO BE STUDIED = '',$)')
        READ(5,'(A10)')FN
160     WRITE(1,180)
180     FORMAT(/7X,'ARE CORRECTIONS NEEDED? ',$)
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          GO TO 100
```

```
          ELSEIF(ICHAR(P(1)) .NE. 78)THEN
            WRITE(1,190)
190         FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER'/)
            PAUSE
            GO TO 160
          ENDIF
          WRITE(1,110)CHAR(12)
          WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/)
     .)')
          PAUSE
C         TRANSFER FILE REWRITTEN TO INDICATE NEW USER MODEL SELECTION
          OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
          WRITE(3)FN
          CLOSE(3,STATUS='KEEP')
        ELSEIF(ICHAR(P(1)) .NE. 89)THEN
          WRITE(1,190)
          PAUSE
          GO TO 100
        ENDIF
        WRITE(1,110)CHAR(12)
        WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/))')
        PAUSE
C       DEFAULT OPTIONS SET
        PROBT=1
        DUAL=1
        OUTP=1
        FMT=1
        ITAB=1
        IBTAB=1
        IFTAB=1
200     WRITE(1,110)CHAR(12)
C       DEFAULT OPTIONS DISPLAYED
        WRITE(1,'(12X,''DEFAULT OPTIONS''/5X,''ENTER OPTION NUMBER TO CHAN
     .GE''/)')
        WRITE(1,'(''1. PROBLEM TO SOLVE'',11X,$)')
        IF(PROBT .EQ. 1)THEN
          WRITE(1,'(4X,''PRIMAL'')')
        ELSE
          WRITE(1,'(6X,''DUAL'')')
        ENDIF
        WRITE(1,'(/,''2. SOLVE BY DUAL PIVOTS'',16X,$)')
        IF(DUAL .EQ. 1)THEN
          WRITE(1,'(''N'')')
        ELSE
          WRITE(1,'(''Y'')')
        ENDIF
        WRITE(1,'(/,''3. OUTPUT LOCATION'',12X,$)')
        IF(OUTP .EQ. 1)THEN
          WRITE(1,'(4X,''SCREEN'')')
        ELSE
          WRITE(1,'(3X,''PRINTER'')')
        ENDIF
```

```
      WRITE(1,'(/,''4. OUTPUT FORMAT'',16X,$)')
      IF(FMT .EQ. 1)THEN
        WRITE(1,'(''F FORMAT'')')
      ELSE
        WRITE(1,'(''E FORMAT'')')
      ENDIF
      WRITE(1,'(/,''5. TABLEAUS TO BE DISPLAYED'',/6X,''INITIAL'',26X,$
     .)')
      IF(ITAB .EQ. 1)THEN
        WRITE(1,'(''Y'')')
      ELSE
        WRITE(1,'(''N'')')
      ENDIF
      WRITE(1,'(6X,''INTERMEDIATE'',16X,''N = '',I2)')IBTAB
      WRITE(1,'(6X,''FINAL'',28X,$)')
      IF(IFTAB .EQ. 1)THEN
        WRITE(1,'(''Y'')')
      ELSE
        WRITE(1,'(''N'')')
      ENDIF
      WRITE(1,'(/,''6. NO CHANGES''//''* SEE DOCUMENTATION FOR EXPLANATI
     .ON''/)')
      WRITE(1,'(6X,''WHICH OPTION (ENTER 1-6) ? '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,6,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,190)
        PAUSE
        GO TO 200
      ENDIF
      GOTO(230,260,290,430,350,460)INEW
C     PROBLEM TYPE TO BE SOLVED CHANGED
230   IF(PROBT .EQ. 1)THEN
        PROBT=2
      ELSE
        PROBT=1
      ENDIF
      GO TO 200
C     DUAL PIVOT OPTION CHANGED
260   IF(DUAL .EQ. 1)THEN
        DUAL=2
      ELSE
        DUAL=1
      ENDIF
      GO TO 200
C     OUTPUT LOCATION CHANGED
290   IF(OUTP .EQ. 1)THEN
        OUTP=2
      ELSE
        OUTP=1
      ENDIF
      GO TO 200
```

```
C     TABLEAU OUTPUT OPTIONS DISPLAYED
350   WRITE(1,110)CHAR(12)
      WRITE(1,130)
      WRITE(1,'(''WHICH TABLEAUS WOULD YOU LIKE DISPLAYED?'')')
370   WRITE(1,'(/5X,''INITIAL TABLEAU? (Y/N) '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        ITAB=1
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        ITAB=2
      ELSE
        WRITE(1,190)
        GO TO 370
      ENDIF
390   WRITE(1,'(/5X,''INTERMEDIATE TABLEAUS? (Y/N) '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,'(/,''EVERY N(TH) INTERMEDIATE TABLEAU WILL BE''/15X,
     ,''DISPLAYED.'')')
400     WRITE(1,'(/4X,''WHAT VALUE DO YOU DESIRE FOR N?''//17X,''N = ''
     ,,$)')
        READ(5,'(2A1)')P(1),P(2)
        CALL CHECK2(P,2,99,INVAL,INEW)
        IF(INVAL .EQ. 1)THEN
          WRITE(1,190)
          PAUSE
          GO TO 400
        ENDIF
        IBTAB=INEW
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        IBTAB=0
      ELSE
        WRITE(1,190)
        GO TO 390
      ENDIF
410   WRITE(1,'(/5X,''FINAL TABLEAU? (Y/N) '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        IFTAB=1
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        IFTAB=2
      ELSE
        WRITE(1,190)
        GO TO 410
      ENDIF
      GO TO 200
C     OUTPUT FORMAT CHANGED
430   IF(FMT .EQ. 1)THEN
        FMT=0
      ELSE
        FMT=1
      ENDIF
```

```
      GO TO 200
460   RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 3 UNIT33                                                       *
C     UNIT $USES: NONE                                                   *
C                                                                        *
C  SUBROUTINE WORK                                                       *
C  USE: SEE MODULE 2, UNIT24, SUBROUTINE WORK                            *
C  CALLED BY: PROGRAM PROBS                                              *
C  CALLS    : NONE                                                       *
C  VARIABLES: SEE MODULE 2, UNIT24, SUBROUTINE WORK                      *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE WORK
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      PELE=A(PR,PK)
      DO 200 J=1,VT
        A(PR,J)=A(PR,J)/PELE
200   CONTINUE
      XB(PR)=XB(PR)/PELE
      CB(PR)=PK
      DO 300 I=1,K
        IF(I .EQ. PR)THEN
          GO TO 300
        ENDIF
        HOLD=A(I,PK)
        DO 250 J=1,VT
          A(I,J)=A(I,J)-HOLD*A(PR,J)
250     CONTINUE
        XB(I)=XB(I)-HOLD*XB(PR)
300   CONTINUE
      HOLD=C(PK)
      DO 350 J=1,VT
        C(J)=C(J)-HOLD*A(PR,J)
350   CONTINUE
      Z=Z-HOLD*XB(PR)
      RETURN
      END
```

```
C *************************************
C  MODULE 3 UNIT33                                                 *
C                                                                  *
C  SUBROUTINE OPTB                                                 *
C  USE: DETERMINES THE PIVOT COLUMN AND PIVOT ROW FOR BOTH PRIMAL AND *
C       DUAL SIMPLEX PROCEDURES, DEPENDENT ON OPTION SELECTION.    *
C       DETERMINES AND SETS FLAGS ACCORDING TO THE FEASIBILITY,    *
C       OPTIMALITY, UNBOUNDEDNESS AND DEGENERACY OF THE CURRENT    *
C       SOLUTION.  DEPENDENT ON USER SELECTION OF TABLEAU OUTPUT,  *
C       DISPLAYS TABLEAU HEADER ON SELECTED OUTPUT DEVICE.         *
C  CALLED BY: PROGRAM PROBS                                        *
C  CALLS    : NONE                                                 *
C  VARIABLES:                                                      *
C       USED: A(*,*),BASIC,C(*),CB(*),DUAL,IBTAB,IFTAB,ITAB,K,KFA,PN, *
C             VT,XB(*)                                             *
C   MODIFIED: GNEG,IFLAG(4),IFLAG(6),IFLAG(7),IFLAG(8),IFLAG(9),INFP, *
C             OPTS,PK,PR,SPR                                       *
C *************************************
      SUBROUTINE OPTB
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P3/DUAL,OUTP,ITAB,IBTAB,IFTAB,FMT,PROBT
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      IFLAG(4)=0
      IFLAG(6)=0
      INFP=0
      GNEG=0.0
      IF(DUAL .EQ. 2)THEN
C       USER HAS ELECTED TO PERFORM DUAL PIVOTS
        GO TO 300
      ENDIF
C     FINDS LARGEST Z(J)-C(J)
110   DO 130 J=1,VT
        IF(C(J) .GE. GNEG)THEN
          GO TO 130
        ELSE
          GNEG=C(J)
          PK=J
        ENDIF
130   CONTINUE
      IF(GNEG .GT. -.0001)THEN
C       NO NEGATIVE Z(J)-C(J) EXIST SO OPTIMAL
        OPTS=1
      ENDIF
C     CHECKS FOR INFEASIBILITY
      DO 150 I=1,V
        IF(CB(I) .LT. KFA)THEN
          IF(XB(I) .LT. 0.0)THEN
```

```
            INFP=1
          ENDIF
        ELSEIF(XB(I) .LE. 0.0)THEN
          GO TO 150
        ELSE
          INFP=1
        ENDIF
150   CONTINUE
      IF(OPTS .EQ. 0)THEN
C       PREVIOUS SOLUTION NOT OPTIMAL SO FIND PIVOT ROW
        SPR=10.E8
        DO 190 I=1,K
          IF(A(I,PK) .LE. .0001)THEN
            GO TO 190
          ELSEIF((XB(I)/A(I,PK)) .GE. SPR)THEN
            GO TO 190
          ELSE
            SPR=XB(I)/A(I,PK)
            PR=I
          ENDIF
190     CONTINUE
C       CHECKS FOR UNBOUNDEDNESS
        IF(SPR .GE. 10.E6)THEN
          IFLAG(7)=1
        ENDIF
      ENDIF
      GO TO 500
C     DUAL PIVOT CALCULATIONS PERFORMED
300   DO 320 J=1,VT
        IF(C(J) .LT. 0.0)THEN
          GO TO 110
        ENDIF
320   CONTINUE
      DO 340 I=1,K
        IF(XB(I) .GE. GNEG)THEN
          GO TO 340
        ELSE
          GNEG=XB(I)
          PR=I
        ENDIF
340   CONTINUE
      IF(GNEG .EQ. 0.0)THEN
        OPTS=1
        GO TO 500
      ELSE
        SPR=-10.E8
        DO 370 J=1,VT
          DO 360 I=1,K
            IF(CB(I) .EQ. J)THEN
              GO TO 370
            ENDIF
360       CONTINUE
```

```
          IF(A(PR,J) .GE. -.0001)THEN
            GO TO 370
          ELSEIF((C(J)/A(PR,J)) .LE. SPR)THEN
            GO TO 370
          ELSE
            SPR=C(J)/A(PR,J)
            PK=J
          ENDIF
370     CONTINUE
      ENDIF
      IF(SPR .LE. -10.E6)THEN
        IFLAG(7)=1
      ENDIF
500   IF(OPTS .EQ. 1)THEN
        IF(INFP .EQ. 1)THEN
          GO TO 600
        ENDIF
C       CHECKS FOR MULTIPLE OPTIMAL SOLUTIONS
        DO 540 J=1,VT
          IFLAG(8)=0
          DO 520 I=1,K
            IF(CB(I) .EQ. J)THEN
              IFLAG(8)=1
            ENDIF
520       CONTINUE
          IF(IFLAG(8) .EQ. 0)THEN
            IF(C(J) .LT. .0001 .AND. C(J) .GT. -.0001)THEN
              IFLAG(4)=1
            ENDIF
          ENDIF
540     CONTINUE
      ENDIF
      IF(IFLAG(7) .EQ. 1)THEN
        GO TO 600
      ENDIF
C     CHECKS FOR DEGENERACY
      DO 560 I=1,K
        IF(XB(I) .GE. -.0001 .AND. XB(I) .LE. .0001)THEN
          IFLAG(6)=1
        ENDIF
560   CONTINUE
C     DETERMINES IF PRESENT SOLUTION TO BE DISPLAYED
600   IF(BASIC .EQ. 1 .AND. ITAB .EQ. 1)THEN
        GO TO 700
      ELSEIF(BASIC .NE. 1)THEN
        IF(OPTS .EQ. 0 .AND. IBTAB .NE. 0)THEN
          IF(((FLOAT(BASIC-1))/(FLOAT(IBTAB))) .EQ. FLOAT((BASIC-1)/
     .IBTAB))THEN
            GO TO 700
          ELSE
            IFLAG(9)=0
            RETURN
```

```
        ENDIF
      ELSEIF(OPTS .EQ. 1 .AND. IFTAB .EQ. 1)THEN
        GO TO 700
      ELSEIF(IFLAG(7) .EQ. 1 .AND. IFTAB .EQ. 1)THEN
        GO TO 700
      ENDIF
    ENDIF
    IFLAG(9)=0
    RETURN
C   PROPER OUTPUT DEVICE OPENED AND TABLEAU HEADER DISPLAYED
700 IF(OUTP .EQ. 1)THEN
      OPEN(2,FILE='CONSOLE:')
      WRITE(2,'(A)')CHAR(12)
    ELSE
      OPEN(2,FILE='PRINTER:')
    ENDIF
    WRITE(2,'(10X,A20/10X,''BASIC SOLUTION #'',I2)')PN,BASIC
    IF(OPTS .EQ. 1 .OR. IFLAG(7) .EQ. 1)THEN
      WRITE(2,'(10X,''FINAL TABLEAU - '',$)')
      IF(INFP .EQ. 1)THEN
        WRITE(2,'(''INFEASIBLE'')')
      ELSEIF(IFLAG(7) .EQ. 1)THEN
        WRITE(2,'(''UNBOUNDED'')')
      ELSE
        WRITE(2,'(''OPTIMAL'')')
      ENDIF
      IF(IFLAG(6) .EQ. 1)THEN
        WRITE(2,'(26X,''DEGENERATE'')')
      ENDIF
    ENDIF
    IF(OPTS .EQ. 1 .AND. IFLAG(4) .EQ. 1)THEN
      WRITE(2,'(5X,''MULTIPLE OPTIMAL SOLUTIONS EXIST'')')
    ENDIF
    WRITE(2,'(/)')
    IFLAG(9)=1
    RETURN
    END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 3 UNIT34                                                         *
C    UNIT $USES: NONE                                                      *
C                                                                          *
C  SUBROUTINE CONVRT                                                       *
C  USE: DEPENDENT ON USER OPTION SELECTION, CONVERTS THE PRIMAL            *
C       PROBLEM INTO ITS DUAL PROBLEM AND ADDS THE NEEDED VARIABLES        *
C       TO ALLOW FOR VARIABLES WHICH ARE UNCONSTRAINED.  USER IS           *
C       INFORMED OF ADDED VARIABLES AND THEIR ASSOCIATION WITH             *
C       PRESENT VARIABLES.  REWRITES DATA FILE WHICH CORRESPONDS TO        *
C       DUAL PROBLEM.                                                      *
C  CALLED BY: PROGRAM PROBS                                                *
C  CALLS    : SUBROUTINE INRD                                              *
C             SUBROUTINE NFILE(N)                                          *
C  VARIABLES:                                                              *
C       USED: C(*),IFLAG(1) THRU IFLAG(10),K,OBJN,V                        *
C   MODIFIED: A(*,*),CN(*),C2(*),IE,INEQ(*),K2,MM,MXMN,N,NEC,NGC,NLC,      *
C             PINEQ(*),VN2(*),V2,XB(*)                                     *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE CONVRT
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10,VN2*6
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT,V2,K2
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      DIMENSION C2(20),VN2(20)
C     ROUTINE CALLED WHICH INITIALIZES VARIABLES AND READS MODEL
      CALL INRD
      IF(MXMN .EQ. 1)THEN
C       PROBLEM STATED AS A MAXIMIZATION PROBLEM
        DO 110 I=1,K
          IF(INEQ(I) .EQ. 1)THEN
            DO 100 J=1,V
              A(I,J)=-A(I,J)
100         CONTINUE
            XB(I)=-XB(I)
          ENDIF
110     CONTINUE
      ELSE
        DO 130 I=1,K
          IF(INEQ(I) .EQ. 0)THEN
            DO 120 J=1,V
              A(I,J)=-A(I,J)
120         CONTINUE
            XB(I)=-XB(I)
          ENDIF
130     CONTINUE
      ENDIF
C     RHS PLACED IN TEMPORARY STORAGE LOCATION
      DO 150 I=1,K
```

```
        C2(I)=XB(I)
150   CONTINUE
      DO 160 I=K+1,20
        C2(I)=0.0
160   CONTINUE
C     PRIMAL C(J)-Z(J) VALUES CONVERTED TO RHS
      DO 170 J=1,V
        XB(J)=C(J)
170   CONTINUE
C     NUMBER OF DUAL VARIABLES INCLUDING UNCONSTRAINED VARIABLES FOUND
      IF(NEC .NE. 0)THEN
        V2=K+NEC
      ELSE
        V2=K
      ENDIF
      K2=V
C     COEFFICIENT MATRIX ROTATED AND PLACED IN UNUSED ORIGINAL A MATRIX
      DO 190 I=1,K
        DO 180 J=1,V
          A(J,I+20)=A(I,J)
180     CONTINUE
190   CONTINUE
      WRITE(1,'(A)')CHAR(12)
C     VARIABLES ADDED TO ALLOW FOR UNCONSTRAINED VARIABLES
      IF(NEC .NE. 0)THEN
        IE=0
        DO 210 I=1,K
          IF(INEQ(I) .EQ. 2)THEN
            IE=IE+1
            WRITE(1,'(A)')CHAR(12)
            WRITE(1,'(9(/),3X,''VARIABLE '',I2,'' HAS BEEN ADDED DUE TO
     .''/''VARIABLE '',I2,'' BEING UNCONSTRAINED IN SIGN.'',7(/))')K+IE,
     .I
            PAUSE
            DO 200 J=1,K2
              IF(A(J,I+20) .NE. 0.0)THEN
                A(J,K+IE+20)=-1.0*(A(J,I+20))
              ENDIF
200         CONTINUE
            IF(C2(I) .NE. 0.0)THEN
              C2(K+IE)=-1.0*C2(I)
            ENDIF
          ENDIF
210     CONTINUE
      ENDIF
      IF(NEC .NE. 0)THEN
        N=1
      ELSE
        N=0
      ENDIF
      NEC=0
      NGC=0
```

```
      NLC=0
C     COUNT BY INEQUALITY TYPE PERFORMED
      DO 220 I=1,K2
        IF(MXMN .EQ. 1)THEN
          INEQ(I)=1
          PINEQ(I)='>'
          NGC=NGC+1
        ELSE
          INEQ(I)=0
          PINEQ(I)='<'
          NLC=NLC+1
        ENDIF
220   CONTINUE
      IF(MXMN .EQ. 1)THEN
        MXMN=2
        MM='MIN'
      ELSE
        MXMN=1
        MM='MAX'
      ENDIF
      IF(IFLAG(5) .EQ. 1)THEN
C       MODEL INCLUDES NAMES SO NAMES ARE CHANGED TO REFLECT DUAL PROB
        OBJN='        '
        DO 230 I=1,K
          VN2(I)=CN(I)
230     CONTINUE
        DO 240 J=1,V
          CN(J)=VN(J)
240     CONTINUE
        DO 250 I=K+1,20
          VN2(I)='      '
250     CONTINUE
      ENDIF
      N=1
      CALL NFILE(N)
C     DUAL MODEL WRIITEN TO DISK UNDER USER SPECIFIED NAME
      WRITE(3)PN,MXMN,MM,K2,V2,NEC,NGC,NLC
      DO 260 I=1,10
        WRITE(3)IFLAG(I)
260   CONTINUE
      DO 280 I=1,K2
        WRITE(3)INEQ(I),PINEQ(I),XB(I)
        DO 270 J=1,V2
          WRITE(3)A(I,J+20)
270     CONTINUE
280   CONTINUE
      DO 290 J=1,V2
        WRITE(3)C2(J)
290   CONTINUE
      IF(IFLAG(5) .EQ. 1)THEN
        DO 300 I=1,K2
          WRITE(3)CN(I)
```

```
300     CONTINUE
        DO 310 J=1,V2
          WRITE(3)VN2(J)
310     CONTINUE
        WRITE(3)OBJN
      ENDIF
      CLOSE(3,STATUS='KEEP')
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/))')
      PAUSE
      RETURN
      END
```

```
C ***************************************
C  MODULE 3 UNIT34                       *
C                                        *
C  SUBROUTINE ACNCH                      *
C  USE: DEPENDENT UPON USER OPTION SELECTION, EXAMINES PRESENT MODEL  *
C       TO INSURE A NEGATIVE C(J) WILL BE PRESENT.  IF NOT PRESENT ,  *
C       CONSTRAINT IS ADDED TO INSURE AN INITIAL PRIMAL PIVOT  WILL   *
C       BE POSSIBLE.  USER IS NOTIFIED OF ADDED CONSTRAINT.  ROUTINE  *
C       REWRITES DATA FILE WHICH CORRESPONDS TO MODEL WITH ADDED      *
C       CONSTRAINT.                      *
C  CALLED BY: PROGRAM PROBS              *
C  CALLS    : SUBROUTINE INRD            *
C             SUBROUTINE NFILE(N)        *
C  VARIABLES:                            *
C       USED: BM,C(*),IFLAG(1) THRU IFLAG(10),MM,MXMN,NEC,NGC,OBJN,  *
C             PN,V,VN(*)                 *
C   MODIFIED: A(*,*),CN(*),INEQ(*),IT,K,N,NLC,PINEQ(*),XB(*)         *
C ***************************************
      SUBROUTINE ACNCH
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P3/DUAL,OUTP,ITAB,IBTAB,IFTAB,FMT,PROBT
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
C     ROUTINE CALLED WHICH INITIALIZES VARIABLES AND READS MODEL
      CALL INRD
      IT=0
      IF(MXMN .EQ. 1)THEN
C       PROBLEM STATED AS A MAXIMIZATION PROBLEM
        DO 160 J=1,V
          IF(C(J) .GT. 0.0)THEN
            IT=1
          ENDIF
160     CONTINUE
      ELSE
        DO 180 J=1,V
          IF(C(J) .LT. 0.0)THEN
            IT=1
          ENDIF
180     CONTINUE
      ENDIF
C     DETERMINES IF INITIAL PIVOT POSSIBLE
      IF(IT .EQ. 0)THEN
        N=0
        GO TO 200
      ENDIF
C     TO INSURE INITIAL PIVOT, CONSTRAINT ADDED
      WRITE(1,'(A)')CHAR(12)
      K=K+1
```

```
      INEQ(K)=0
      PINEQ(K)='<'
      NLC=NLC+1
      XB(K)=BM
      DO 190 J=1,V
        A(K,J)=1.0
190   CONTINUE
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(9(/),2X,''A CONSTRAINT HAS BEEN ADDED TO THIS''/1X,''PRO
     .BLEM TO INSURE AN INITIAL PIVOT MAY''/13X,''BE PERFORMED.''//''THE
     . CONSTRAINT IS THE SUM OF X(1) THRU'')')
      WRITE(1,'(5X,''X('',I2,'') IS LESS-THAN OR EQUAL TO''/3X,1PE12.5,2
     .X,''(THE VALUE OF BIG M)'',5(/))')V,BM
      PAUSE
      IF(IFLAG(5) .EQ. 1)THEN
        CN(K)='ADDED'
      ENDIF
      N=2
200   CALL NFILE(N)
C     NEW FILE CONTAINING MODEL WITH ADDED CONSTRAINT WRITTEN TO DISK
      WRITE(3)PN,MXMN,MN,K,V,NEC,NGC,NLC
      DO 230 I=1,10
        WRITE(3)IFLAG(I)
230   CONTINUE
      DO 250 I=1,K
        WRITE(3)INEQ(I),PINEQ(I),XB(I)
        DO 240 J=1,V
          WRITE(3)A(I,J)
240     CONTINUE
250   CONTINUE
      DO 270 J=1,V
        WRITE(3)C(J)
270   CONTINUE
      IF(IFLAG(5) .EQ. 1)THEN
        DO 290 I=1,K
          WRITE(3)CN(I)
290     CONTINUE
        DO 310 J=1,V
          WRITE(3)VN(J)
310     CONTINUE
        DO 330 J=V+1,20
          VN(J)='      '
330     CONTINUE
        WRITE(3)OBJN
      ENDIF
      CLOSE(3,STATUS='KEEP')
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/))')
      PAUSE
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 3 UNIT34                                                     *
C                                                                      *
C  SUBROUTINE INRD                                                     *
C  USE: INITIALIZES VARIABLES TO ZERO EXCEPT CHARACTER VARIABLES.      *
C       PROMPTS USER TO INSURE DISK WITH FILE TO BE STUDIED IS         *
C       PRESENT AND READS FILE FROM DISX.                              *
C  CALLED BY: PROGRAM PROBS                                            *
C             SUBROUTINE CONVRT                                        *
C             SUBROUTINE INRD                                          *
C  CALLS    : NONE                                                     *
C  VARIABLES:                                                          *
C       USED: NONE                                                     *
C   MODIFIED: A(*,*),ARTV(*),BASIC,BM,C(*),CB(*),CJ,CN(*),FN,IA,       *
C             IFLAG(1) THRU IFLAG(10),INDEXE,INDEXG,INDEXL,INEQ(*),K,  *
C             NEC,NGC,NLC,OBJN,OPTS,PINEQ(*),V,VN(*),XB(*),Z           *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE INRD
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      DO 150 I=1,20
        ARTV(I)=0
        CB(I)=0
        INEQ(I)=0
        XB(I)=0.0
        DO 130 J=1,60
          A(I,J)=0.0
130     CONTINUE
150   CONTINUE
      DO 170 J=1,60
        C(J)=0.0
170   CONTINUE
      DO 190 I=1,10
        IFLAG(I)=0
190   CONTINUE
      NEC=0
      NGC=0
      NLC=0
      Z=0.0
      IA=0
      BASIC=0
      OPTS=0
      INDEXE=0
      INDEXG=0
      INDEXL=0
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(9(/),5X,''INSURE THE DISK CONTAINING THE''//15X,A10
```

```
     .//10X,''MODEL IS AVAILABLE.'',7(/))')FN
      PAUSE
C     FILE WHICH CONTAINS MODEL OPENED AND READ
      OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
      READ(3)PN,MXMN,MM,K,V,NEC,NGC,NLC
      DO 210 I=1,10
        READ(3)IFLAG(I)
210   CONTINUE
      DO 230 I=1,K
        READ(3)INEQ(I),PINEQ(I),XB(I)
        DO 220 J=1,V
          READ(3)A(I,J)
220     CONTINUE
230   CONTINUE
      DO 250 J=1,V
        READ(3)C(J)
250   CONTINUE
      IF(IFLAG(5) .EQ. 1)THEN
        DO 270 I=1,K
          READ(3)CN(I)
270     CONTINUE
        DO 290 J=1,V
          READ(3)VN(J)
290     CONTINUE
        DO 310 J=V+1,20
          VN(J)='      '
310     CONTINUE
        READ(3)OBJN
      ENDIF
      IFLAG(2)=1
      CLOSE(3,STATUS='KEEP')
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/))')
      PAUSE
C     FIND APPROPRIATE VALUE FOR BIG M
      CJ=0.0
      BM=0.0
      DO 350 J=1,V
        IF(ABS(C(J)) .GT. BM)THEN
          BM=ABS(C(J))
        ENDIF
350   CONTINUE
      BM=(ANINT(BM))*10.0
      IF(BM .LT. 1.0)THEN
        BM=10.0
      ENDIF
      RETURN
      END
```

```
C *************************************
C MODULE 3 UNIT34                      *
C                                      *
C SUBROUTINE NFILE(N)                  *
C USE: DEPENDENT ON FLAG (N), DISPLAYS BRIEF EXPLANATION OF NEW   *
C      FILE BEING CREATED.  SOLICITS VOLUME:FILENAME INPUT BY USER *
C      FOR CREATION OF NEW FILE AND OPENS THIS FILE ON VOLUME     *
C      SPECIFIED.                      *
C CALLED BY: SUBROUTINE ACNCH          *
C            SUBROUTINE CONVRT         *
C CALLS    : NONE                      *
C VARIABLES:                           *
C      USED: N                         *
C  MODIFIED: P(*),FN,TN                *
C *************************************
      SUBROUTINE NFILE(N)
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10,TN*11
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      WRITE(1,110)CHAR(12)
110   FORMAT(A)
      IF(N .EQ. 1)THEN
        TN='VARIABLES'
      ELSEIF(N .EQ. 2)THEN
        TN='CONSTRAINTS'
      ELSE
        WRITE(1,'(1X,''A NEW DISK:FILENAME MUST BE CREATED SO''/''THAT T
     .HE ORIGINAL''//15X,A10//''WILL NOT BE DESTROYED AND MAY BE REUSED.
     .'')')FN
        GO TO 130
      ENDIF
      WRITE(1,'(1X,''DUE TO THE ADDITION OF '',A11,'', A''/''NEW DISK:FI
     .LE MUST BE CREATED TO CONTAIN''/''THE NEW MODEL. THIS WILL ALLOW T
     .HE USER''/'' TO REUSE''//15X,A10//''AT A LATER TIME.'')')TN,FN
130   WRITE(1,'(/,''PLEASE ENTER, IN 10 CHARACTERS OR LESS,''/2X,''A DRI
     .VE:FILENAME TO STORE THIS FILE.'')')
140   WRITE(1,'(/,''NEW MODELS DRIVE:FILENAME = '',$)')
      READ(5,'(A10)')FN
150   WRITE(1,'(/7X,''ARE CORRECTIONS NEEDED? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 140
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,160)
160     FORMAT(/5X,'INVALID ENTRY. PLEASE REENTER')
        GO TO 150
      ENDIF
      WRITE(1,'(/,''HAS THIS DISK:FILENAME COMBINATION BEEN''/12X,''USED
     . PREVIOUSLY?''/''(ARE YOU UPDATING A CURRENTLY EXISTING''/17X,''FI
     .LE?'')')
200   WRITE(1,'(16X,''(Y/N) '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
```

```
  OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
  OPEN(3,FILE=FN,STATUS='NEW',FORM='UNFORMATTED')
ELSE
  WRITE(1,160)
  GO TO 200
ENDIF
WRITE(1,110)CHAR(12)
WRITE(1,'(9(/),6X,''INSURE THE DISK TO CONTAIN''//15X,A10//10X,
.''MODEL IS AVAILABLE.'',7(/))')FN
PAUSE
RETURN
END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 3 UNIT35                                                      *
C     UNIT $USES: NONE                                                  *
C                                                                       *
C  SUBROUTINE PSHED                                                     *
C  USE: DISPLAYS TITLE PAGE OF MODULE 3, PROBLEM SOLVER MODULE.         *
C  CALLED BY: PROGRAM PROBS                                             *
C  CALLS    : NONE                                                      *
C  VARIABLES: NONE                                                      *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE PSHED
      WRITE(1,'(A)')CHAR(12)
      WRITE(1,'(3(/),9X,22(''*'')/9X,''*'',20X,''*''/9X,''*'',7X,''LINEA
     .R'',7X,''*''/9X,''*'',20X,''*''/9X,''*'',4X,''PROGRAMMING'',5X,''*
     .''/9X,''*'',20X,''*'')')
      WRITE(1,'(9X,''*'',7X,''PROBLEM'',6X,''*''/9X,''*'',20X,''*''/9X,
     .''*'',7X,''SOLVER'',7X,''*''/9X,''*'',20X,''*''/9X,''*'',7X,''MODU
     .LE'',7X,''*'')')
      WRITE(1,'(2(9X,''*'',20X,''*''/),9X,''*'',6X,''MODULE 3'',6X,''*''
     ./9X,''*'',20X,''*''/9X,22(''*''),3(/))')
      PAUSE
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  MODULE 3 UNIT35                                                            *
C                                                                             *
C  SUBROUTINE ASKQ(ASK)                                                       *
C  USE: SEE MODULE 2, UNIT25, SUBROUTINE ASKQ(ASK)                            *
C  CALLED BY: PROGRAM PROBS                                                   *
C  CALLS    : NONE                                                            *
C  VARIABLES: SEE MODULE 2, UNIT25, SUBROUTINE ASKQ(ASK)                      *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE ASKQ(ASK)
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10,
     .FNO*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT,ASK
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P3/DUAL,OUTP,ITAB,IBTAB,IFTAB,FMT,PROBT
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      DIMENSION AO(20,20),B(20),CO(20)
      FNO=FN
100   WRITE(1,110)CHAR(12)
110   FORMAT(A)
      ASK=0
      IFLAG(10)=DUAL-1
      WRITE(1,'(8(/),''TO PERFORM SENSITIVITY ANALYSIS ON THIS''/''MODEL
     ., THE INFORMATION OF THE CURRENT''/''TABLEAU MUST BE SAVED TO DISK
     ..''//)')
130   WRITE(1,'(''DO YOU WISH TO SAVE THIS FILE TO DISK? '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        WRITE(1,110)CHAR(12)
        WRITE(1,'(//9X,''SAVE LP MODEL TO DISK''///2X,''ENTER THE DISK D
     .RIVE NUMBER AND FILE''/2X,''NAME YOU WANT THE CURRENT TABLEAU OF''
     ./3X,A20,'' SAVED UNDER.'')')PN
        WRITE(1,'(/8X,''ENTER EXACTLY AS FOLLOWS''/10X,''DISK DRIVE:FILE
     .NAME''//12X,''EG.   $4:FILENAM''//''THE DRIVE:FILENAME MUST BE 10
     . CHARACTERS''/16X,''OR LESS.''//1X,''DO NOT USE THE SAME NAME USED
     . WHEN THE''/6X,''ORIGINAL MODEL WAS ENTERED.'')')
        WRITE(1,'(/7X,''DISK:FILENAME = '',$)')
        READ(5,'(A10)')FN
150     WRITE(1,'(/7X,''ARE CORRECTIONS NEEDED? '',$)')
        READ(5,'(A1)')P(1)
        IF(ICHAR(P(1)) .EQ. 89)THEN
          GO TO 100
        ELSEIF(ICHAR(P(1)) .NE. 78)THEN
          WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER'')')
          GO TO 150
        ENDIF
        WRITE(1,110)CHAR(12)
        WRITE(1,'(11(/),1X,''INSURE DISK    LP2:    IS AVAILABLE.'',7(/)
     .)')
```

```
      PAUSE
C     TRANSFER FILE OPENED AND FILE NAME WRITTEN
      OPEN(3,FILE='LP2:LFDATAN',STATUS='OLD',FORM='UNFORMATTED')
      WRITE(3)FN
C     TRANSFER FILE CLOSED
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
C     USER PROMPTED TO INSERT DISK WHICH SOLVED MODEL IS TO BE SAVED
      WRITE(1,'(9(/),2X,''INSURE THE DISK TO CONTAIN THE FILE''//15X,
     .A10//13X,''IS AVAILABLE'',7(/))')FN
      PAUSE
      WRITE(1,110)CHAR(12)
C     CURRENT STATUS OF FILE INPUT BY USER
      WRITE(1,'(9(/),''HAS THIS DISK:FILENAME COMBINATION BEEN''/12X,
     .''USED PREVIOUSLY?''//''(ARE YOU UPDATING A CURRENTLY EXISTING''/
     .17X,''FILE?)'')')
200   WRITE(1,'(//16X,''(Y/N) '',$)')
      READ(5,'(A1)')P(1)
      IF(ICHAR(P(1)) .EQ. 89)THEN
        OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
      ELSEIF(ICHAR(P(1)) .EQ. 78)THEN
        OPEN(3,FILE=FN,STATUS='NEW',FORM='UNFORMATTED')
      ELSE
        WRITE(1,210)
210     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 200
      ENDIF
C     SOLVED MODEL WRITTEN TO DISK
      WRITE(3)PN,MXMN,K,V,IFLAG(5)
      WRITE(1,110)CHAR(12)
      WRITE(1,'(9(/),5X,''INSURE THE DISK CONTAINING THE''//15X,A10
     .//10X,''MODEL IS AVAILABLE.'',7(/))')FNO
      PAUSE
C     ORIGINAL MODEL FILE OPENED TO READ ORIGINAL PARAMETERS
      OPEN(4,FILE=FNO.STATUS='OLD',FORM='UNFORMATTED')
      READ(4)PN,MXMN,MM,K,V,NEC,NGC,NLC
      DO 220 I=1,10
        READ(4)IFLAG(I)
220   CONTINUE
      DO 240 I=1,K
        READ(4)INEQ(I),PINEQ(I),B(I)
        DO 230 J=1,V
          READ(4)AO(I,J)
230     CONTINUE
240   CONTINUE
      DO 250 J=1,V
        READ(4)CO(J)
250   CONTINUE
      CLOSE(4,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(9(/),''INSURE THE DISK TO CONTAIN THE FILE FOR''//15X,
     .A10//13X,''IS AVAILABLE.'',7(/))')FN
```

```
      PAUSE
      DO 270 I=1,K
        WRITE(3)INEQ(I),B(I)
        DO 260 J=1,V
          WRITE(3)AO(I,J)
260     CONTINUE
270   CONTINUE
C     SOLVED MODEL AND ORIGINAL PARAMETERS WRITTEN TO DISK
      DO 275 J=1,V
        WRITE(3)CO(J)
275   CONTINUE
      WRITE(3)IFLAG(10),VT
      DO 290 I=1,K
        WRITE(3)XB(I),CB(I)
        DO 280 J=1,VT
          WRITE(3)A(I,J)
280     CONTINUE
290   CONTINUE
      DO 300 J=1,VT
        WRITE(3)C(J)
300   CONTINUE
      WRITE(3)Z
      IF(IFLAG(5) .EQ. 1)THEN
        DO 310 I=1,K
          WRITE(3)CN(I)
310     CONTINUE
        DO 320 J=1,V
          WRITE(3)VN(J)
320     CONTINUE
        WRITE(3)OBJN
      ENDIF
      CLOSE(3,STATUS='KEEP')
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/)
     .)')
      PAUSE
    ELSEIF(ICHAR(P(1)) .NE. 78)THEN
      WRITE(1,210)
      GO TO 130
    ENDIF
390 WRITE(1,110)CHAR(12)
    WRITE(1,'(11(/),1X,''WOULD YOU LIKE TO STUDY ANOTHER MODEL''/4X,''
   .WHICH HAS BEEN SAVED TO DISK? '',$)')
    READ(5,'(A1)')P(1)
    IF(ICHAR(P(1)) .EQ. 89)THEN
400   WRITE(1,110)CHAR(12)
      WRITE(1,'(9(/),2X,''ENTER DISK DRIVE NUMBER AND FILENAME''/4X,''
   .WHICH THE MODEL IS SAVED UNDER.'')')
      WRITE(1,'(/6X,''MODEL TO STUDY = '',$)')
      READ(5,'(A10)')FN
450   WRITE(1,'(//7X,''ARE CORRECTIONS NEEDED? '',$)')
      READ(5,'(A1)')P(1)
```

```
      IF(ICHAR(P(1)) .EQ. 89)THEN
        GO TO 400
      ELSEIF(ICHAR(P(1)) .NE. 78)THEN
        WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER'')')
        GO TO 450
      ENDIF
      WRITE(1,110)CHAR(12)
      WRITE(1,'(11(/),1X,''INSURE DISK    LP1    IS AVAILABLE.'',7(/)
     .)')
      PAUSE
C     TRANSFER FILE OPENED AND NEW MODEL FILE NAME WRITTEN
      OPEN(3,FILE='LP1:LPDATA',STATUS='OLD',FORM='UNFORMATTED')
      WRITE(3)FN
      CLOSE(3,STATUS='KEEP')
      ASK=1
      RETURN
    ELSEIF(ICHAR(P(1)) .NE. 78)THEN
      WRITE(1,'(/5X,''INVALID ENTRY, PLEASE REENTER'')')
      GO TO 390
    ENDIF
    WRITE(1,110)CHAR(12)
    WRITE(1,'(11(/),1X,''INSURE DISK    LP2    IS AVAILABLE.'',7(/))')
    PAUSE
    WRITE(1,110)CHAR(12)
    WRITE(1,'(8(/),1X,''TO ENTER THE LP DATABASE MODULE:''//17X,''TYPE
   .''//19X,''X''/11X,''LP1:SYSTEM.STARTUP.'',3(/))')
    STOP
    RETURN
    END
```

```
C **************************************
C  MODULE 3 UNIT35                      *
C                                       *
C  SUBROUTINE BIGM                      *
C  USE: SEE MODULE 2, UNIT25, SUBROUTINE BIGM  *
C  CALLED BY: PROGRAM PROBS             *
C  CALLS    : NONE                      *
C  VARIABLES: SEE MODULE 2, UNIT25, SUBROUTINE BIGM  *
C **************************************
      SUBROUTINE BIGM
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NSC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P3/DUAL,OUTP,ITAB,IBTAB,IFTAB,FMT,PROBT
      IA=IA-1
      KFSU=V+NGC
      DO 300 J=1,KFSU
        SUM=0.0
        DO 200 I=1,IA
          SUM=SUM+A(ARTV(I),J)
200     CONTINUE
        C(J)=C(J)-(BM*SUM)
300   CONTINUE
      DO 400 J=KFA,VT
        C(J)=0.0
400   CONTINUE
      SUM=0.0
      DO 500 I=1,IA
        SUM=SUM+XB(ARTV(I))
500   CONTINUE
      Z=Z-(BM*SUM)
      RETURN
      END
```

```
C ***************************************
C   MODULE 3 UNIT35                                                   *
C                                                                     *
C   SUBROUTINE INDEX                                                  *
C   USE: SEE MODULE 2, UNIT25, SUBROUTINE INDEX                       *
C   CALLED BY: PROGRAM PROBS                                          *
C   CALLS    : NONE                                                   *
C   VARIABLES: SEE MODULE 2, UNIT25, SUBROUTINE INDEX                 *
C ***************************************
      SUBROUTINE INDEX
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,F(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      IA=1
      KFSA=V+1
      VT=V+(2*NGC)+NLC+NEC
      KFS=V+NGC+1
      KFA=KFS+NLC
      KFSU=V+NGC
      INDEXG=V+1
      INDEXL=V+NGC+1
      INDEXE=V+NGC+NLC+1
      DO 200 I=1,K
        IF(INEQ(I) .EQ. 0)THEN
C       SLACK VARIABLE ADDED TO CONSTRAINT
          CB(I)=INDEXL
          A(I,INDEXL)=1.
          INDEXL=INDEXL+1
        ELSEIF(INEQ(I) .EQ. 1)THEN
C         SURPLUS AND ARTIFICAL VARIABLE ADDED TO CONSTRAINT
          CB(I)=INDEXE
          ARTV(IA)=I
          IA=IA+1
          A(I,INDEXE)=1.
          INDEXE=INDEXE+1
          A(I,INDEXG)=-1.
          INDEXG=INDEXG+1
        ELSE
C         ARTIFICAL VARIABLE ADDED TO CONSTRAINT
          ARTV(IA)=I
          IA=IA+1
          CB(I)=INDEXE
          A(I,INDEXE)=1.
          INDEXE=INDEXE+1
        ENDIF
      PINEQ(I)='='
200   CONTINUE
      IF(IFLAG(5) .EQ. 1)THEN
```

```
      DO 210 J=KFSA,KFS-1
        VN(J)='SURPLS'
210   CONTINUE
      DO 220 J=KFS,KFA-1
        VN(J)='SLACK'
220   CONTINUE
      DO 230 J=KFA,VT
        VN(J)='ARTIF'
230   CONTINUE
     ENDIF
     RETURN
     END
```

```
C ***************************************
C  MODULE 3 UNIT35                                                  *
C                                                                   *
C  SUBROUTINE MODIFP                                                *
C  USE: TRANSFORMS OBJECTIVE FUNCTION FROM MAX OR MIN Z=X FORM TO MAX *
C       Z-X=0 FORM.  MULTIPLIES ALL CONSTRAINTS WITH NEGATIVE RHS'S  *
C       BY -1 AND CHANGES INEQUALITY ACCORDINGLY.  ONLY USED WHEN    *
C       PRIMAL PROBLEM IS BEING SOLVED.                             *
C  CALLED BY: PROGRAM PROBS                                         *
C  CALLS    : NONE                                                  *
C  VARIABLES:                                                       *
C       USED: IFLAG(4),K,MXMN,V                                     *
C   MODIFIED: A(*,*),C(*),INEQ(*),NGC,NLC,MM,PINEQ(*),XB(*)         *
C ***************************************
      SUBROUTINE MODIFP
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      IF(MXMN .EQ. 1)THEN
C       PROBLEM STATED AS MAXIMIZATION PROBLEM
        DO 160 J=1,V
          C(J)=-C(J)
160     CONTINUE
      ELSE
        MM='MAX'
      ENDIF
      IFLAG(4)=1
      DO 300 I=1,K
        IF(XB(I) .LT. 0.0)THEN
          XB(I)=-XB(I)
          DO 200 J=1,V
            A(I,J)=-A(I,J)
200       CONTINUE
C         COUNT OF INEQUALITIES UPDATED DUE TO MULT BY -1
          IF(INEQ(I) .EQ. 0)THEN
            NLC=NLC-1
            NGC=NGC+1
            INEQ(I)=1
            PINEQ(I)='>'
          ELSEIF(INEQ(I) .EQ. 1)THEN
            NGC=NGC-1
            NLC=NLC+1
            INEQ(I)=0
            PINEQ(I)='<'
          ENDIF
        ENDIF
300   CONTINUE
      RETURN
```

END

```
C ****************************************
C  MODULE 3 UNIT35                                                *
C                                                                 *
C  SUBROUTINE MODIFD                                              *
C  USE: TRANSFORMS OBJECTIVE FUNCTION FROM MAX OR MIN Z=X FORM TO MAX *
C       Z-X=0 FORM.  TRANSFORMS ALL CONSTRAINTS WITH GREATER-THAN OR  *
C       EQUAL INEQUALITY TO LESS-THAN OR EQUAL INEQUALITY.        *
C  CALLED BY: PROGRAM PROBS                                       *
C  CALLS    : NONE                                                *
C  VARIABLES:                                                     *
C       USED: K,MXMN,V                                            *
C   MODIFIED: A(*,*),C(*),INEQ(*),NGC,NLC,MM,PINEQ(*),XB(*)       *
C ****************************************
      SUBROUTINE MODIFD
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FNT,PROBT
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NGC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
      IF(MXMN .EQ. 1)THEN
C       PROBLEM STATED AS MAXIMIZATION PROBLEM
        DO 120 J=1,V
          C(J)=-C(J)
120     CONTINUE
      ELSE
        MM='MAX'
      ENDIF
      DO 150 I=1,K
        IF(INEQ(I) .EQ. 1)THEN
          XB(I)=-XB(I)
          DO 140 J=1,V
            A(I,J)=-A(I,J)
140       CONTINUE
C         COUNT OF INEQUALITIES UPDATED DUE TO MULY BY -1
          PINEQ(I)='<'
          INEQ(I)=0
          NGC=NGC-1
          NLC=NLC+1
        ENDIF
150   CONTINUE
      RETURN
      END
```

```
C ****************************************
C  MODULE 3 UNIT36                        *
C    UNIT $USES: UNIT37                   *
C                                         *
C  SUBROUTINE TDISPL                      *
C  USE: DETERMINES THE NUMBER OF 80 COLUMN WIDTHS REQUIRED TO DISPLAY *
C       TABLEAU.  DISPLAYS TABLEAU TO USER SELECTED OUTPUT DEVICE.   *
C       ALSO SOLICITS INPUT FROM USER AS TO DISPLAYING THE BASIC     *
C       VARIABLES VALUE AFTER EACH DISPLAYED TABLEAU.  IF REQUESTED, *
C       DISPLAYS THE BASIC VALUES ON SELECTED OUTPUT DEVICE WITH     *
C       OBJECTIVE FUNCTION VALUE ALSO DISPLAYED.                     *
C  CALLED BY: PROGRAM PROBS                                          *
C  CALLS    : SUBROUTINE CHECK2(P,N,M,INVAL,INEW)                    *
C  VARIABLES:                                                        *
C       USED: A(*,*),BASIC,C(*),CB(*),CN(*),FMT,IFLAG(5),INEW,INVAL, *
C             K,PINEQ(*),PN,VN(*),VT,XB(*),Z                         *
C    MODIFIED: P(*)                                                  *
C ****************************************
$USES UCHECK2 IN UNIT37.CODE OVERLAY
      SUBROUTINE TDISPL
      CHARACTER VN*6,CN*6,PN*20,MM*3,FN*10,PINEQ*1,P(10)*1,OBJN*10
      INTEGER ARTV,BASIC,PK,PR,OPTS,V,VT,CB,DUAL,OUTP,FMT,PROBT,T
      COMMON/P1/A(20,60),ARTV(20),C(60),Z,INEQ(20),IFLAG(10),CB(20),
     .XB(20),K,V,VT,MXMN,BASIC,OPTS,BM
      COMMON/P2/NEC,NSC,NLC,IA,INDEXE,INDEXG,INDEXL,KFA,KFS,KFSA,KFSU,
     .PK,PR
      COMMON/P3/DUAL,OUTP,ITAB,IBTAB,IFTAB,FMT,PROBT
      COMMON/P4/VN(20),CN(20),PN,MM,FN,PINEQ(20),OBJN
110   FORMAT(A)
C     NUMBER OF 80 COLUMN DISPLAYS REQUIRED DETERMINED
      T=(VT/5)+1
      DO 470 N=1,T
        IF(IFLAG(5) .EQ. 1)THEN
C         VARIABLES NAMES PRINTED AS COLUMN HEADERS
          WRITE(2,'(13X,$)')
          DO 270 J=(N*5)-4,N*5
            IF(J .GT. VT)THEN
              GO TO 270
            ENDIF
            WRITE(2,260)VN(J)
260         FORMAT(5X,A6,2X,$)
270       CONTINUE
          WRITE(2,'('' '')')
        ENDIF
        WRITE(2,'(13X,$)')
        DO 290 J=(N*5)-4,N*5
          IF(J .GT. VT)THEN
            GO TO 290
          ENDIF
          WRITE(2,280)J
280       FORMAT(5X,'X(',I2,')',3X,$)
290     CONTINUE
```

```
C       IF LAST 80 COLUMN DISPLAY, DISPLAY RHS
        IF(T .EQ. 1 .OR. N .EQ. T)THEN
          WRITE(2,300)
300       FORMAT(6X,'RHS')
        ELSE
          WRITE(2,'(''  '')')
        ENDIF
        WRITE(2,'(''OBJ FUNCTION'',1X,$)')
        DO 320 J=(N*5)-4,N*5
          IF(J .GT. VT)THEN
            GO TO 320
          ENDIF
          IF(FMT .EQ. 0)THEN
            WRITE(2,'(1PE12.5,1X,$)')C(J)
          ELSE
            WRITE(2,'(F12.5,1X,$)')C(J)
          ENDIF
320     CONTINUE
        IF(T .EQ.1 .OR. N .EQ. T)THEN
          IF(FMT .EQ. 0)THEN
            WRITE(2,'(''= '',1PE12.5,1X)')Z
          ELSE
            WRITE(2,'(''= '',F12.5,1X)')Z
          ENDIF
        ELSE
          WRITE(2,'(''  '')')
        ENDIF
        WRITE(2,'(''CN  NAME  VAR'',2X,65(''*''/)')
C       CONSTRAINT NUMBER, NAME, BASIC VARIABLE, COEFFICIENTS
C         INEQUALITY, AND RHS DISPLAYED
        DO 400 L=1,K
          IF(L .GT. K)THEN
            GO TO 400
          ENDIF
          IF(IFLAG(5) .EQ. 1)THEN
            WRITE(2,'(I2,1X,A6,$)')L,CN(L)
          ELSE
            WRITE(2,'(I2,7X,$)')L
          ENDIF
          WRITE(2,'(1X,I2,1X,$)')CB(L)
          DO 370 J=(N*5)-4,N*5
            IF(J .GT. VT)THEN
              GO TO 370
            ENDIF
            IF(FMT .EQ. 0)THEN
              WRITE(2,'(1PE12.5,1X,$)')A(L,J)
            ELSE
              WRITE(2,'(F12.5,1X,$)')A(L,J)
            ENDIF
370       CONTINUE
          IF(T .EQ. 1 .OR. N .EQ. T)THEN
            IF(FMT .EQ. 0)THEN
```

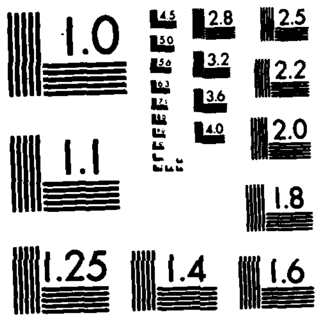
```
            WRITE(2,'(A1,1X,1PE12.5)')PINEQ(L),XB(L)
          ELSE
            WRITE(2,'(A1,1X,F12.5)')PINEQ(L),XB(L)
          ENDIF
        ELSE
          WRITE(2,'('' '')')
        ENDIF
400   CONTINUE
      IF(OUTP .EQ. 1)THEN
        PAUSE
        WRITE(2,110)CHAR(12)
      ELSE
        WRITE(2,'(2(/))')
      ENDIF
470   CONTINUE
      CLOSE(2)
C     USER SELECTS DISPLAY LOCATION OF BASIC VARIABLES AND VALUES
490   WRITE(1,110)CHAR(12)
      WRITE(1,'(5(/),''WOULD YOU LIKE THE BASIC SOLUTION VALUES''/14X,
     .''DISPLAYED? '')')
      WRITE(1,'(/9X,''1. DISPLAY ON SCREEN''//9X,''2. DISPLAY ON PRINTER
     .''//9X,''3. DO NOT DISPLAY'')')
500   WRITE(1,'(//13X,''WHICH OPTION? '',$)')
      READ(5,'(A1)')P(1)
      CALL CHECK2(P,1,3,INVAL,INEW)
      IF(INVAL .EQ. 1)THEN
        WRITE(1,530)
530     FORMAT(/5X,'INVALID ENTRY, PLEASE REENTER')
        GO TO 500
      ENDIF
      IF(INEW .EQ. 1)THEN
        OPEN(2,FILE='CONSOLE:')
      ELSEIF(INEW .EQ. 2)THEN
        OPEN(2,FILE='PRINTER:')
      ELSE
        RETURN
      ENDIF
      WRITE(1,110)CHAR(12)
      WRITE(2,'(10X,A20/10X,''BASIC SOLUTION #'',I2,/)')PN,BASIC
      IF(IFLAG(5) .EQ. 1)THEN
        DO 550 I=1,K
          IF(FMT .EQ. 0)THEN
            WRITE(2,'(5X,A6,'' = X('',I2,'') = '',1PE12.5)')VN(CB(I)),
     .CB(I),XB(I)
          ELSE
            WRITE(2,'(5X,A6,'' = X('',I2,'') = '',F12.5)')VN(CB(I)),
     .CB(I),XB(I)
          ENDIF
550     CONTINUE
          IF(FMT .EQ. 0)THEN
          WRITE(2,'(/10X,''Z= '',1PE12.5)')Z
        ELSE
```

```
        WRITE(2,'(/18X,''Z= '',F12.5)')Z
      ENDIF
    ELSE
      DO 580 I=1,K
        IF(FMT .EQ. 0)THEN
          WRITE(2,'(10X,''X('',I2,'') = '',1PE12.5)')CB(I),XB(I)
        ELSE
          WRITE(2,'(10X,''X('',I2,'') = '',F12.5)')CB(I),XB(I)
        ENDIF
580   CONTINUE
      IF(FMT .EQ. 0)THEN
        WRITE(2,'(/14X,''Z = '',1PE12.5)')Z
      ELSE
        WRITE(2,'(/14X,''Z = '',F12.5)')Z
      ENDIF
    ENDIF
    IF(INEW .EQ. 2)THEN
      WRITE(2,'(6(/))')
    ELSE
      PAUSE
    ENDIF
    RETURN
    END
```

AD-A124 804 FORTRAN BASED LINEAR PROGRAMMING FOR MICROCOMPUTERS(U) 3/3
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING T R FRALEY ET AL. DEC 82

UNCLASSIFIED AFIT/GOR/OS/82D-4 F/G 12/1 NL

END
DATE
FILMED
83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
C ***************************************
C  MODULE 3 UNIT37                                                      *
C                                                                       *
C  SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)                               *
C  USE: SEE MODULE 1, UNIT17, SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)    *
C  CALLED BY: SUBROUTINE OPTN                                           *
C             SUBROUTINE TDISPL                                         *
C  CALLS    : NONE                                                      *
C  VARIABLES: SEE MODULE 1, UNIT17, SUBROUTINE CHECK2(E,D,HVAL,INVAL,   *
C             INEW)                                                     *
C ***************************************
      SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
      CHARACTER ALLOW*1,E*1
      DIMENSION E(10),ALLOW(11)
      INTEGER D,HVAL
      DATA ALLOW/'1','2','3','4','5','6','7','8','9','0',' '/
      INEW=0
      INVAL=0
      DO 300 I=1,D
        DO 200 J=1,10
C           CHECKS FIRST FOR BLANK CHARACTERS
          IF(E(I) .EQ. ALLOW(11))THEN
            GO TO 300
          ELSEIF(E(I) .EQ. ALLOW(J))THEN
            INEW=INEW*10 + (ICHAR(E(I))-48)
            GO TO 300
          ELSEIF(J .EQ. 10)THEN
            INVAL=1
            INEW=0
            RETURN
          ENDIF
200     CONTINUE
300   CONTINUE
      IF(INEW .EQ. 0 .OR. INEW .GT. HVAL)THEN
        INVAL=1
        INEW=0
        RETURN
      ENDIF
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C                                                                           *
C  MODULE 4 UNIT40                                                          *
C    UNIT $USES: UNITS 41 THROUGH 45, 47, 48                                *
C                                                                           *
C  PROGRAM MAINSA                                                           *
C  USE: THIS IS THE MAIN PROGRAM IN THE SENSITIVITY ANALYSIS MODULE.        *
C       IT IS USED TO CALL OTHER SUBROUTINES IN RESPONSE TO USER            *
C       INPUT AND TO CONTROL THE OVERLAY PROCESS WHICH ALLOWS LARGER        *
C       PROGRAMS                                                            *
C                                                                           *
C  CALLED BY: NONE                                                          *
C  CALLS    : SUBROUTINE PETRIV                                             *
C             SUBROUTINE COMRHS                                             *
C             SUBROUTINE COEFFR                                             *
C             SUBROUTINE MULCNG                                             *
C             SUBROUTINE SELECT                                             *
C             SUBROUTINE ADDCON                                             *
C                                                                           *
C  VARIABLES:                                                               *
C  USES     : SELSUB                                                        *
C  MODIFIES : SELOUT, IFLAG(9), IFLAG(2)                                    *
C                                                                           *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK2 IN UNIT47.CODE OVERLAY
$USES URETRIV IN UNIT48.CODE OVERLAY
$USES UCOMRHS IN UNIT41.CODE OVERLAY
$USES UCOEFFR IN UNIT42.CODE OVERLAY
$USES UMULCNG IN UNIT43.CODE OVERLAY
$USES UADDCON IN UNIT44.CODE OVERLAY
$USES USOLVE  IN UNIT'[illegible].CODE OVERLAY
      PROGRAM MAINSA
      INTEGER K,V,VT,IFLAG,INEQ,CB,INDEXG,INDEXL,INDEXE,NEG
      REAL A0,AF,B0,BF,C0,CF,Z,BM
      CHARACTER SELSUB,SELOUT,SELSOL,FN*10
      COMMON/P1/OPTS,KFA,PK,PR
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
      COMMON/THREE/INFP
$INCLUDE COMVAR
      OPEN(1,FILE='CONSOLE:')
      OPEN(5,FILE='CONSOLE:')
4000  WRITE(1,'(A1,/)') CHAR(12)
C     THE USER SELECTS THE DESIRED TYPE OF SENSITIVITY ANALYSIS.
4010  WRITE(1,'(//,1X,''PLEASE SELECT ONE ITEM BY NUMBER'',///
     .,3X,''1) RANGE LIMITS-----RIGHT-HAND-SIDE'',/
     .,6X,  ''AND ASSOCIATED Z VALUES    '',//
     .,3X,''2) RANGE LIMITS-----A(I,J) & C(J)'',/)')
      WRITE(1,'(3X,''3) CHECK OPTIMALITY FOR MULTIPLE'',/
     .,6X,  ''A(I,J), B(I), OR C(J) CHANGES'',//
     .,3X,''4) ADD A VARIABLE OR A CONSTRAINT'',//
```

```
     .,3X,''5) EXIT PROGRAM''/)')
      READ(5,'(A1)') SELSUB
      IF(SELSUB .EQ.'5') THEN
        GOTO 4030
      ENDIF
C     CHECK FOR VALID INPUT
      IF ((SELSUB .NE. '1') .AND.(SELSUB .NE. '2') .AND. (SELSUB .NE.
     . '3') .AND. (SELSUB .NE. '4')) THEN
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(/,8X,''INVALID RESPONSE'',/)')
        GOTO 4010
      ENDIF
C     CLEAR THE SCREEN AND SELECT OUTPUT.
      WRITE(1,'(A1)') CHAR(12)
4020  WRITE(1,'(4(/),4X,''DO YOU WANT THE OUTPUT TO GO TO:'',///
     .,9X,     ''S)CREEN'',//
     .,9X,     ''P)PRINTER'',/)')
      WRITE(1,'(13X,          ''OR'',//
     .,9X,     ''B)OTH'',///
     .,4X,''SELECT S, P, OR B''///)')
      READ(5,'(A1)') SELOUT
      IF ((SELOUT .NE. 'P') .AND. (SELOUT .NE. 'S') .AND.
     . (SELOUT .NE. 'B')) THEN
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(/,8X,''INVALID RESPONSE'')')
        GOTO 4020
      ENDIF
      IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN
        OPEN(6,FILE='PRINTER:')
      ENDIF
C     THE MAIN PART OF THE PROGRAM.  SUBROUTINE RETRIV READS ALL
C     NECESSARY DATA FROM A DISK FILE AND THEN THE SUBROUTINE IS CALLED
C     TO COMPLETE THE SENSITIVITY ANALYSIS.  FOLLOWING THE ANALYSIS,
C     THE USER IS RETURNED TO THE MAIN MENU.
      CALL RETRIV
      IF (SELSUB .EQ. '1') THEN
          CALL COMRHS
          GOTO 4000
        ELSEIF (SELSUB .EQ. '2') THEN
          CALL COEFFR
          GOTO 4000
        ELSEIF (SELSUB .EQ. '3') THEN
          CALL MULCNG
          WRITE(1,'(A1)') CHAR(12)
          IF(SELOUT.EQ.'S'.OR.SELOUT.EQ.'B')THEN
            IF(IFLAG(9).EQ.1)THEN
              WRITE(1,'(''THE ADDED CHANGES HAVE MADE THE PROBLEM'')')
              WRITE(1,'(''ILL-CONDITIONED, RETURNING TO MAIN MENU'')')
              PAUSE
            ENDIF
          ENDIF
          IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN
```

```
            IF(IFLAG(9).EQ.1)THEN
              WRITE(6,'(''THE ADDED CHANGES HAVE MADE THE PROBLEM'')')
              WRITE(6,'(''ILL-CONDITIONED, RETURNING TO MAIN MENU'')')
              WRITE(6,'(A1)')CHAR(12)
            ENDIF
          ENDIF
          IF(IFLAG(9).EQ.1)THEN
            IFLAG(9)=0
            GOTO 4000
          ENDIF
          IF(IFLAG(3).EQ.1)THEN
            GOTO 4000
          ENDIF
          CALL SELECT
          GOTO 4000
        ELSEIF (SELSUB .EQ. '4') THEN
          IFLAG(2)=0
          CALL ADDCON
          IF(IFLAG(3).EQ.1)THEN
            IF(SELOUT.EQ.'S'.OR.SELOUT.EQ.'B')THEN
              WRITE(1,'(5(/),5X,''******* NOT FEASIBLE *******'')')
              PAUSE
            ENDIF
            IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN
              WRITE(6,'(5(/),5X,''******* NOT FEASIBLE *******'')')
              WRITE(6,'(A1)')CHAR(12)
            ENDIF
            GOTO 4000
          ENDIF
          WRITE(1,'(A1)') CHAR(12)
          IF(IFLAG(2).EQ.0)THEN
            CALL SELECT
          ENDIF
          GOTO 4000
      ENDIF
4030  WRITE(1,'(A1)') CHAR(12)
      WRITE(1,'(8(/),1X,''TO ENTER THE LP DATABASE MODULE:''//17X,
     .''TYPE''//19X,''X''/11X,''LP1:SYSTEM.STARTUP.'',///)')
      PAUSE
      STOP
      END
```

```
C ****************************************
C                                        *
C  MODULE 4 UNIT40                       *
C    UNIT $USES: NONE                    *
C                                        *
C  SUBROUTINE SELECT                     *
C  USE: THIS SUBROUTINE CALLS SUBROUTINE SOLVE IF THE USER DESIRES A  *
C       NEW FINAL TABLEAU.               *
C                                        *
C  CALLED BY: PROGRAM MAINSA             *
C  CALLS    : SOLVE                      *
C                                        *
C  VARIABLES:                            *
C  USES     : SELSOL                     *
C  MODIFIES : NONE                       *
C                                        *
C ****************************************
      SUBROUTINE SELECT
      CHARACTER SELSOL
4090  WRITE(1,'(5(/),5X,''DO YOU WISH TO SOLVE THIS TABLEAU'')')
      WRITE(1,'(//,10X,''SELECT ''''Y'''' OR ''''N'''' '')')
      READ(5,'(A1)') SELSOL
      WRITE(1,'(A1)') CHAR(12)
      IF (SELSOL .EQ. 'Y') THEN
          CALL SOLVE
        ELSEIF(SELSOL .NE. 'N') THEN
          WRITE(1,'(//,10X,''IMPROPER RESPONSE'')')
          GOTO 4090
      ENDIF
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C                                                                       *
C  MODULE 4 UNIT41                                                      *
C    UNIT $USES: NONE                                                   *
C                                                                       *
C  SUBROUTINE: COMRHS                                                   *
C  USE: THE SUBROUTINE DETERMINES THE MINIMUM AND MAXIMUM VALUES OF     *
C       EACH RIGHT-HAND SIDE IN THE ORIGINAL EQUATION WHICH WOULD       *
C       NOT CAUSE A BASIS CHANGE.  THE NEW SOLUTION IS SHOWN FOR EACH   *
C       OF THESE UPPER AND LOWER BOUNDS.                                *
C                                                                       *
C  CALLED BY: PROGRAM MAINSA                                            *
C  CALLS    : NONE                                                      *
C                                                                       *
C  VARIABLES:                                                           *
C  USES     : RMIN(20),RMAX(20),RSCH(20,20),LWBD(20,20),UPBD(20,20),    *
C             RSULIM(20),RSLLIM(20),ZLP,ZUP,ZLS,ZUS,TEMP,BM,SELOUT,     *
C             K,V,VT,INEQ,IFLAG,NEG(20),COL,CONSTR,INDEXL,MXMN          *
C  MODIFIES : AF,AO,BF,BO,CO,CF,Z                                       *
C                                                                       *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE  COMRHS
      INTEGER VT,V,K,J,COL,IFLAG,INEQ,NEG,
     .CB,INDEXG,INDEXL,INDEXE,CONSTR,MXMN
      REAL RMAX(20),RMIN(20),AO,RSCH(20,20),BO,CF,BF,TEMP,BM,
     .CO,RSULIM(20),RSLLIM(20),LWBD(20,20),UPBD(20,20),ZLS,ZUS,ZLP,ZUP
      CHARACTER SELOUT,FN*10
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
$INCLUDE COMVAR
C     RIGHT-HAND-SIDE RANGING IS DONE FOR EACH CONSTRAINT.
      DO 4120 CONSTR = 1,K
C       THE COLUMN OF B-INVERSE ASSOCIATED WITH THE CURRENT CONSTRAINT IS
C       DETERMINED.
        COL=CONSTR+INDEXL-1
        RMAX(CONSTR) = 10E12
        RMIN(CONSTR) =-10E12
C                     DETERMINE RESOURCE LIMITS
C       THE MINIMUM POSITIVE AND MAXIMUM NEGATIVE VALUES WHICH WILL
C       CAUSE A DEGENERATIVE CONDITION ARE DETERMINED.
        DO 4140 L = 1,K
          IF (ABS(AF(L,COL)) .GT. .0001) THEN
            IF(ABS(BF(L)).GT. .0001)THEN
              RSCH(L,CONSTR) = -BF(L)/AF(L,COL)
              IF(RSCH(L,CONSTR) .GT. 0)THEN
                RMAX(CONSTR)=AMIN1(RMAX(CONSTR),RSCH(L,CONSTR))
               ELSE
                RMIN(CONSTR)=AMAX1(RMIN(CONSTR),RSCH(L,CONSTR))
              ENDIF
             ELSEIF(CB(L) .GE. INDEXE)THEN
              RMAX(CONSTR)=0
              RMIN(CONSTR)=0
```

```
              ELSEIF(AF(L,COL) .LT. 0)THEN
                RMAX(CONSTR)=0
              ELSE
                RMIN(CONSTR)=0
            ENDIF
          ENDIF
4140    CONTINUE
        RSULIM(CONSTR) = BO(CONSTR) + RMAX(CONSTR)
        RSLLIM(CONSTR) = BO(CONSTR) + RMIN(CONSTR)
C       IF THE CONSTRAINT HAS BEEN MULTIPLIED BY MINUS ONE, THE
C       RESOURCE UPPER LIMIT (RSULIM) AND THE RESOURCE LOWER LIMIT
C       (RSLLIM) AS WELL AS THE ORIGINAL RIGHT-HAND SIDE ARE REVERSED.
        IF(NEG(CONSTR) .EQ.1)THEN
          TEMP=-RSULIM(CONSTR)
          RSULIM(CONSTR)=-RSLLIM(CONSTR)
          RSLLIM(CONSTR)=TEMP
          BO(CONSTR)=-BO(CONSTR)
        ENDIF
C       THE UPPER BOUND (UPBD) AND LOWER BOUND (LWBD) FOR EACH RHS IN
C       THE FINAL SOLUTION ARE OBTAINED.
        DO 4150 L = 1,K
          UPBD(L,CONSTR) = AF(L,COL) * RMAX(CONSTR) +BF(L)
          LWBD(L,CONSTR) = AF(L,COL) * RMIN(CONSTR) +BF(L)
          IF (NEG(CONSTR) .EQ. 1) THEN
            TEMP=UPBD(L,CONSTR)
            UPBD(L,CONSTR)=LWBD(L,CONSTR)
            LWBD(L,CONSTR)=TEMP
          ENDIF
4150    CONTINUE
        ZUS=0
        ZLS=0
        ZUP=0
        ZLP=0
C       THE RESULTS ARE PRINTED FOR EACH BOUND DEPENDING ON THE
C       CONDITIONS.
        DO 4160 L = 1,K
          IF(SELOUT .EQ. 'S' .OR. SELOUT .EQ.'B')THEN
            IF(L.EQ.1.OR.L.EQ.8.OR.L.EQ.15)THEN
              WRITE(1,'(A1)') CHAR(12)
              WRITE(1,'(/,40(''*''),/)')
              WRITE(1,'(6X,''RIGHT HAND SIDE RANGE LIMITS'')')
              WRITE(1,'(12X,''CONSTRAINT # '',I2,/)')CONSTR
              WRITE(1,'(''ORIGINAL RIGHT HAND SIDE = '',F12.5)')
     .        BO(CONSTR)
              IF(ABS(RSLLIM(CONSTR)).GT.1E6)THEN
                  WRITE(1,'(13X,''LOWER BOUND =     NO LIMIT'')')
                ELSE
                  WRITE(1,'(13X,''LOWER BOUND = '',F12.5)')
     .            RSLLIM(CONSTR)
              ENDIF
              IF(ABS(RSULIM(CONSTR)).GT.1E6)THEN
                  WRITE(1,'(13X,''UPPER BOUND =     NO LIMIT'')')
```

```
        ELSE
          WRITE(1,'(13X,''UPPER BOUND = '',F12.5)')
          RSULIM(CONSTR)
        ENDIF
        WRITE(1,'(40(''*''),/)')
        WRITE(1,'(''AT THE LOWER BOUND--AT THE UPPER ''
        ''BOUND'')')
      ENDIF
      IF(CB(L) .LT.10)THEN
         IF(ABS(UPBD(L,CONSTR)).GE.1E6.AND.ABS(LWBD(L,CONSTR))
            .GE.1E6)THEN
            WRITE(1,'(''X('',I1,'') =    NO LIMIT X('',I1,
            '') =     NO LIMIT'')')
            IF(CB(L) .LE.V)THEN
              ZUS=ZUS+10E8
              ZLS=ZLS-10E8
            ENDIF
          ELSEIF(ABS(UPBD(L,CONSTR)).GE.1E6)THEN
            WRITE(1,'(''X('',I1,'') = '',F12.5,'' X(''
            ,I1,'') =     NO LIMIT'')')CB(L),LWBD(L,CONSTR),
            CB(L)
            IF(CB(L).LE.V)THEN
              ZLS=ZLS+CO(CB(L))*LWBD(L,CONSTR)
              ZUS=ZUS+10E8
            ENDIF
          ELSEIF(ABS(LWBD(L,CONSTR)).GE.1E6)THEN
            WRITE(1,'(''X('',I1,'') =    NO LIMIT X('',I1,
            '') = '',F12.5)')CB(L),CB(L),UPBD(L,CONSTR)
            IF(CB(L) .LE.  V )THEN
              ZLS=-10E8
              ZUS=ZUS+CO(CB(L))*UPBD(L,CONSTR)
            ENDIF
          ELSE
            WRITE(1,'(''X('',I1,'') = '',F12.5,'' X('',I1,
            '') = '',F12.5)')CB(L),LWBD(L,CONSTR),CB(L),
            UPBD(L,CONSTR)
            IF(CB(L).LE.V)THEN
              ZLS=ZLS+CO(CB(L))*LWBD(L,CONSTR)
              ZUS=ZUS+CO(CB(L))*UPBD(L,CONSTR)
            ENDIF
         ENDIF
        ELSE
         IF(ABS(UPBD(L,CONSTR)).GE.1E6.AND.ABS(LWBD(L,CONSTR))
            .GE.1E6)THEN
            WRITE(1,'(''X('',I2,'')=    NO LIMIT X('',I2,
            '')=     NO LIMIT'')')
            IF(CB(L) .LE.V)THEN
              ZUS=ZUS+10E8
              ZLS=ZLS-10E8
            ENDIF
          ELSEIF(ABS(UPBD(L,CONSTR)).GE.1E6)THEN
            WRITE(1,'(''X('',I2,'')= '',F12.5,'' X(''
```

```
     .        ,I2,'')=      NO LIMIT'')')CB(L),LWBD(L,CONSTR),
     .        CB(L)
              IF(CB(L).LE.V)THEN
                ZLS=ZLS+CO(CB(L))*LWBD(L,CONSTR)
                ZUS=ZUS+10E8
              ENDIF
            ELSEIF(ABS(LWBD(L,CONSTR)).GE.1E6)THEN
              WRITE(1,'(''X('',I2,'')=     NO LIMIT X('',I2,
     .        '')='',F12.5)')CB(L),CB(L),UPBD(L,CONSTR)
              IF(CB(L) .LE. V  )THEN
                ZLS=-10E8
                ZUS=ZUS+CO(CB(L))*UPBD(L,CONSTR)
              ENDIF
            ELSE
              WRITE(1,'(''X('',I2,'')= '',F12.5,'' X('',I2,
     .        '')= '',F12.5)')CB(L),LWBD(L,CONSTR),CB(L),
     .        UPBD(L,CONSTR)
              IF(CB(L).LE.V)THEN
                ZLS=ZLS+CO(CB(L))*LWBD(L,CONSTR)
                ZUS=ZUS+CO(CB(L))*UPBD(L,CONSTR)
              ENDIF
          ENDIF
        ENDIF
        IF(L.EQ.7.OR.L.EQ.14)THEN
          PAUSE
        ENDIF
      ENDIF
      IF(CONSTR .EQ. 1 .AND. L .EQ. 1)THEN
        LINES=0
      ENDIF
      IF(SELOUT .EQ.'P'.OR. SELOUT .EQ.'B')THEN
        IF(L.EQ.1.)THEN
          IF(CONSTR .NE. 1)THEN
            IF(LINES+K .GT. 44)THEN
              WRITE(6,'(A1)') CHAR(12)
              LINES=0
            ENDIF
          ENDIF
          WRITE(6,'(/,15X,A10)')FN
          WRITE(6,'(//,50(''*''),/)')
          WRITE(6,'(6X,''RIGHT HAND SIDE RANGE LIMITS'')')
          WRITE(6,'(12X,''CONSTRAINT # '',I2,/)')CONSTR
          WRITE(6,'(''ORIGINAL RIGHT HAND SIDE = '',F12.5)')
     .    BO(CONSTR)
          IF(ABS(RSLLIM(CONSTR)).GE.1E6)THEN
              WRITE(6,'(13X,''LOWER BOUND =     NO LIMIT'')')
            ELSE
              WRITE(6,'(13X,''LOWER BOUND = '',F12.5)')
     .        RSLLIM(CONSTR)
          ENDIF
          IF (ABS(RSULIM(CONSTR)).GE.1E6)THEN
              WRITE(6,'(13X,''UPPER BOUND =     NO LIMIT'')')
```

```
         ELSE
           WRITE(6,'(13X,''UPPER BOUND = '',F12.5)')
     *     RSULIM(CONSTR)
       ENDIF
       WRITE(6,'(50(''*''),/)')
       WRITE(6,'(''AT THE LOWER BOUND'',12X,''AT THE UPPER ''
     * ''BOUND'')')
       LINES=LINES+15
     ENDIF
     IF(CB(L) .LT.10)THEN
         IF(ABS(UPBD(L,CONSTR)).GE.1E6.AND.ABS(LWBD(L,CONSTR))
     *       .GE.1E6)THEN
             WRITE(6,'(''X('',I1,'') =      NO LIMIT'',11X,''X('',
     *       I1,'') =      NO LIMIT'')')
             IF(CB(L) .LE.V)THEN
               ZUP=ZUP+10E8
               ZLP=ZLP-10E8
             ENDIF
           ELSEIF(ABS(UPBD(L,CONSTR)).GE.1E6)THEN
             WRITE(6,'(''X('',I1,'') = '',F12.5,10X,'' X(''
     *       ,I1,'') =      NO LIMIT'')')CB(L),LWBD(L,CONSTR),
     *       CB(L)
             IF(CB(L).LE.V)THEN
               ZLP=ZLP+CO(CB(L))*LWBD(L,CONSTR)
               ZUP=ZUP+10E8
             ENDIF
           ELSEIF(ABS(LWBD(L,CONSTR)).GE.1E6)THEN
             WRITE(6,'(''X('',I1,'') =      NO LIMIT'',11X,'' X(''
     *       ,I1,'') = '',F12.5)')CB(L),CB(L),UPBD(L,CONSTR)
             IF(CB(L) .LE. V)THEN
               ZLP=-10E8
               ZUP=ZUP+CO(CB(L))*UPBD(L,CONSTR)
             ENDIF
           ELSE
             WRITE(6,'(''X('',I1,'') = '',F12.5,10X,'' X('',I1,
     *       '') = '',F12.5)')CB(L),LWBD(L,CONSTR),CB(L),
     *       UPBD(L,CONSTR)
             IF(CB(L).LE.V)THEN
               ZLP=ZLP+CO(CB(L))*LWBD(L,CONSTR)
               ZUP=ZUP+CO(CB(L))*UPBD(L,CONSTR)
             ENDIF
         ENDIF
       ELSE
         IF(ABS(UPBD(L,CONSTR)).GE.1E6.AND.ABS(LWBD(L,CONSTR))
     *       .GE.1E6)THEN
             WRITE(6,'(''X('',I2,'')=     NO LIMIT'',11X,''X('',
     *       I2,'')=     NO LIMIT'')')
             IF(CB(L) .LE.V)THEN
               ZUP=ZUP+10E8
               ZLP=ZLP-10E8
             ENDIF
           ELSEIF(ABS(UPBD(L,CONSTR)).GE.1E6)THEN
```

```
                    WRITE(6,'(''X('',I2,'')= '',F12.5,10X,'' X(''
     .              ,I2,'')=     NO LIMIT'')')CB(L),LWBD(L,CONSTR),
     .              CB(L)
                    IF(CB(L).LE.V)THEN
                      ZLP=ZLP+CO(CB(L))*LWBD(L,CONSTR)
                      ZUP=ZUP+10E8
                    ENDIF
                  ELSEIF(ABS(LWBD(L,CONSTR)).GE.1E6)THEN
                    WRITE(6,'(''X('',I2,'')=    NO LIMIT'',11X,'' X(''
     .              ,I2,'')= '',F12.5)')CB(L),CB(L),UPBD(L,CONSTR)
                    IF(CB(L) .LE. V )THEN
                      ZLP=-10E8
                      ZUP=ZUP+CO(CB(L))*UPBD(L,CONSTR)
                    ENDIF
                  ELSE
                    WRITE(6,'(''X('',I2,'')= '',F12.5,10X,'' X('',I2,
     .              '')= '',F12.5)')CB(L),LWBD(L,CONSTR),CB(L),
     .              UPBD(L,CONSTR)
                    IF(CB(L).LE.V)THEN
                      ZLP=ZLP+CO(CB(L))*LWBD(L,CONSTR)
                      ZUP=ZUP+CO(CB(L))*UPBD(L,CONSTR)
                    ENDIF
                ENDIF
              ENDIF
              LINES=LINES+1
            ENDIF
4160      CONTINUE
          IF(MXMN .EQ.2)THEN
            ZLP=-ZLP
            ZUP=-ZUP
            ZLS=-ZLS
            ZUS=-ZUS
          ENDIF
C         THE VALUE OF Z IS PRINTED FOR EACH UPPER AND LOWER LIMIT.
          IF(SELOUT .EQ. 'S' .OR. SELOUT .EQ. 'B')THEN
            IF(ABS(ZLS).GE.1E9.AND.ABS(ZUS).GE.1E9)THEN
                WRITE(1,'(''Z=         NO LIMIT Z=         NO LIMIT'')')
              ELSEIF(ABS(ZLS).GE.1E9.AND.ABS(ZUS).LT.1E9)THEN
                WRITE(1,'(''Z=         NO LIMIT Z= '',F16.5)')ZUS
              ELSEIF(ABS(ZLS).LT.1E9.AND.ABS(ZUS).GE.1E9)THEN
                WRITE(1,'(''Z= '',F16.5,'' Z=         NO LIMIT'')')ZLS
              ELSE
                WRITE(1,'(''Z= '',F16.5,'' Z= '',F16.5)')ZLS,ZUS
            ENDIF
          ENDIF
          IF(SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B')THEN
            IF(ABS(ZLP).GE.1E9.AND.ABS(ZUP).GE.1E9)THEN
                WRITE(6,'(''Z=         NO LIMIT'',11X,
     .          ''Z=         NO LIMIT'')')
              ELSEIF(ABS(ZLP).GE.1E9.AND.ABS(ZUP).LT.1E9)THEN
                WRITE(6,'(''Z=         NO LIMIT'',11X,''Z= '',F16.5)')
     .          ZUP
```

```
          ELSEIF(ABS(ZLP).LT.1E9.AND.ABS(ZUP).GE.1E9)THEN
            WRITE(6,'(''Z= '',F16.5,11X,''Z=         NO LIMIT''
            )')ZLP
          ELSE
            WRITE(6,'(''Z= '',F16.5,11X,''Z= '',F16.5)')ZLP,ZUP
        ENDIF
        WRITE(6,'(50(''*''))')
        LINES=LINES+2
      ENDIF
      IF(SELOUT .EQ.'S'.OR.SELOUT .EQ.'B')THEN
        PAUSE
      ENDIF
4120  CONTINUE
      IF(SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B')THEN
        WRITE(6,'(A1)')CHAR(12)
      ENDIF
      RETURN
      END
```

```
C***************************************
C                                      *
C  MODULE 4 UNIT42                     *
C    UNIT $USES: NONE                  *
C                                      *
C  SUBROUTINE COEFFR                   *
C  USE: THIS SUBROUTINE DETERMINES THE MAXIMUM AND MINIMUM VALUES   *
C       FOR THE CONSTRAINT COEFFICIENTS AND THE OBJECTIVE FUNCTION  *
C       COEFFICIENTS WHICH WILL NOT CAUSE A BASIS CHANGE.           *
C                                      *
C  CALLED BY: PROGRAM MAINSA           *
C  CALLS    : NONE                     *
C                                      *
C  VARIABLES:                          *
C  USES     : V,K,VT,IFLAG,INEQ,INDEXG,INDEXL,CB,NEG,INDEXE,AO,AF,  *
C             BO,BF,CO,CF,Z            *
C  MODIFIES : BASIC(20),ILL1,ILL2,HEAD1.HEAD2,CLOWER,CUPPER,TEMP,   *
C             TEMPA,DELAUP,DELADN,CKILL1,CKILL2                     *
C                                      *
C***************************************
      SUBROUTINE  COEFFR
      INTEGER V,K,J,VT,COL,IFLAG,INEQ,
     .INDEXG,INDEXL,CONSTR,LINES,CB,NEG,INDEXE,LINEP,
     .BASIC(20),ROW,ILL1,ILL2.HEAD1.HEAD2,MXMN
      REAL AO,AF,BO,BF,CO,CF,Z,CLOWER,CUPPER,UPBO,
     .LWBO,TEMP,TEMPA.DELAUP,DELADN,CKILL1,CKILL2,BM
      CHARACTER SELOUT*1,FN*10
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL.INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
$INCLUDE CONVAR
C     EACH COLUMN IS CHECKED TO DETERMINE IF THE VARIABLE IS IN THE
C     BASIS.
      DO 4201 COL= 1,VT
        BASIC(COL)=0
        DO 4202 CONSTR=1,K
          IF(CB(CONSTR) .EQ. COL)THEN
            BASIC(COL)=1
          ENDIF
4202    CONTINUE
4201  CONTINUE
      HEAD1=0
      HEAD2=0
      LINES=0
      LINEP=0
C     THE RANGES ARE DETERMINED FOR EACH CONSTRAINT COEFFICIENT IN THE
C     ORIGINAL PROBLEM.
      DO 4210 CONSTR=1,K
        DO 4220 COL=1,V
C         A SEPARATE ALGORITHM IS USED FOR COLUMNS IN THE BASIS VS.
C         THOSE NOT IN THE BASIS.
          IF(BASIC(COL) .EQ. 1)THEN
              DO 4230 ROW=1,K
```

```
C               EACH ROW IS CHECKED TO DETERMINE IF IT IS THE ROW WHICH
C               HAS THE BASIS VARIABLE OF THE COLUMN UNDER INVESTIGATION.
                IF(CB(ROW) .EQ. COL)THEN
                  DELADN=-10E8
                  DELAUP=10E8
                  DO 4245 L=1,VT
C                   THE ONLY COLUMNS WHICH HAVE AN OBJECTIVE FUNCTION
C                   COEFFICIENT WHICH COULD BE DRIVEN NEGATIVE ARE THOSE
C                   WHICH DO NOT HAVE ARTIFICIAL VARIABLES.
                    IF(L.LT.INDEXL.OR.INEQ(CONSTR).EQ.0)THEN
C                     ONLY NON-BASIC COLUMNS HAVE THE POTENTIAL TO ENTER
C                     TO ENTER THE BASIS.
                      IF(BASIC(L) .NE. 1)THEN
C                       DIVISION BY ZERO IS AVOIDED.
                        IF(ABS(AF(ROW,CONSTR+INDEXL-1)*CF(L)-AF(ROW,
     .                      L)*CF(CONSTR+INDEXL-1)) .LT. .00001)THEN
                            IF(CF(L) .LT. 0)THEN
                              TEMPA=9.9E8
                            ELSE
                              TEMPA=-9.9E8
                            ENDIF
                          ELSE
C                           IF ALL CONDITIONS HAVE BEEN MET, THE VALUE
C                           OF THE CHANGES TO THE COEFFICIENT IS FOUND
C                           WHICH WOULD CAUSE AN OBJECTIVE FUNCTION
C                           COEFFICIENT TO BE DRIVEN TO ZERO.
                            TEMPA=-CF(L)/(AF(ROW,CONSTR+INDEXL-1)
     .                      *CF(L)-AF(ROW,L)*CF(CONSTR+INDEXL-1))
                        ENDIF
C                       THE MINIMUM POSITIVE AND MAXIMUM NEGATIVE
C                       (MINIMUM ABSOLUTE) VALUES ARE RETAINED.
                        IF(TEMPA .GT. 0)THEN
                            DELAUP=AMIN1(DELAUP,TEMPA)
                          ELSE
                            DELADN=AMAX1(DELADN,TEMPA)
                        ENDIF
                      ENDIF
                    ENDIF
4245              CONTINUE
                  DO 4260 M=1,K
C                   FOR EACH ROW NOT POSSESSING THE BASIS VARIABLE UNDER
C                   INVESTIGATION, THE VALUE OF THE CHANGE TO THE
C                   COEFFICIENT IS FOUND WHICH WOULD CAUSE A RIGHT-HAND-
C                   SIDE VALUE TO BE DRIVEN TO ZERO.
                    IF(CB(M) .NE. COL)THEN
                      IF(ABS(BF(M)*AF(ROW,CONSTR+INDEXL-1)-BF(ROW)*
     .                    AF(M,CONSTR+INDEXL-1)) .LT. .00001)THEN
                          IF(BF(M) .LT. 0)THEN
                            TEMPA=9.9E8
                          ELSE
                            TEMPA=-9.9E8
                          ENDIF
```

```
                  ELSEIF(CB(M).GE.INDEXE)THEN
                    DELAUP=0
                    DELADN=0
                  ELSE
                    TEMPA=-BF(M)/(BF(M)*AF(ROW,CONSTR+INDEXL-1)-
                    BF(ROW)*AF(M,CONSTR+INDEXL-1))
                ENDIF
C               THE MINIMUM VALUES ARE COMPARED TO THE PREVIOUSLY
C               FOUND MINIMUM VALUES, AND THE SMALLEST ARE
C               RETAINED.
                IF(TEMPA .GT.0)THEN
                    DELAUP=AMIN1(DELAUP,TEMPA)
                  ELSE
                    DELADN=AMAX1(DELADN,TEMPA)
                ENDIF
              ENDIF
4260          CONTINUE
C             THE JUST DETERMINED MINIMUM VALUES ARE CHECKED FOR
C             ILL-CONDITIONING.
              CKILL1=1+DELADN*AF(ROW,CONSTR+INDEXL-1)
              CKILL2=1+DELAUP*AF(ROW,CONSTR+INDEXL-1)
              IF(ABS(CKILL1).LT. .1 .OR. ABS(CKILL2).LT..1)THEN
                ILL1=1
                ILL2=1
              ENDIF
              ENDIF
4230        CONTINUE
            UPBD=AO(CONSTR,COL)+DELAUP
            LWBD=AO(CONSTR,COL)+DELADN
          ELSE
C           THE UPPER AND LOWER BOUNDS ARE DETERMINED FOR THOSE
C           COLUMNS WITH VARIABLES NOT IN THE BASIS.
            UPBD=10E8
            LWBD=-10E8
            IF(ABS(CF(CONSTR+INDEXL-1)) .GT. .0001)THEN
              IF(CF(CONSTR+INDEXL-1) .GT.0)THEN
                LWBD=AO(CONSTR,COL)-CF(COL)/CF(CONSTR+INDEXL-1)
              ELSE
                UPBD=AO(CONSTR,COL)-CF(COL)/CF(CONSTR+INDEXL-1)
              ENDIF
            ENDIF
        ENDIF
        IF(NEG(CONSTR) .EQ. 1)THEN
          TEMP=UPBD
          UPBD=-LWBD
          LWBD=-TEMP
        ENDIF
C       THE CONSTRAINT COEFFICIENT RANGES ARE PRINTED ACCORDING TO
C       THE CURRENT CONDITIONS.
        IF(SELOUT.EQ.'S'.OR.SELOUT.EQ.'B')THEN
          IF(HEAD1 .EQ. 0 .OR.LINES .GE.18)THEN
            IF(LINES .GE.18)THEN
```

```
           PAUSE
         ENDIF
         WRITE(1,'(A1)')CHAR(12)
         WRITE(1,'(//,40(''*''))')
         WRITE(1,'(''COEFFICIENT    LOWERLIMIT     UPPERLIMIT'')')
         LINES=5
         HEAD1=1
       ENDIF
           IF(ABS(LWBD).GE.1E7.AND.ABS(UPBD).GE.1E8)THEN
               WRITE(1,'(1X,''A('',I1,'','',I1,'')   =      ''
     .         ''NO LIMIT       NO LIMIT'')')CONSTR,COL
             ELSEIF(ABS(LWBD).GE.1E7)THEN
               WRITE(1,'(1X,''A('',I1,'','',I1,'')   =      ''
     .         ''NO LIMIT'',F15.5)')CONSTR,COL,UPBD
             ELSEIF(ABS(UPBD).GE.1E8)THEN
               WRITE(1,'(1X,''A('',I1,'','',I1,'')   ='',F14.5,
     .         ''       NO LIMIT'')')CONSTR,COL,LWBD
             ELSE
               WRITE(1,'(1X,''A('',I1,'','',I1,'')   ='',F14.5,
     .         F15.5)')CONSTR,COL,LWBD,UPBD
           ENDIF
       IF(ILL1.EQ.1)THEN
         WRITE(1,'(8X,''MAY BE ILL-CONDITIONED''/)')
         ILL1=0
         LINES=LINES+2
       ENDIF
       LINES=LINES+1
     ENDIF
     IF(SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B')THEN
       IF(HEAD2 .EQ. 0 .OR. LINEP .GE. 56)THEN
         IF(LINEP .GE.56)THEN
           WRITE(6,'(A1)')CHAR(12)
         ENDIF
         WRITE(6,'(///,15X,A10)')FN
         WRITE(6,'(47(''*''))')
         WRITE(6,'(5X,''COEFFICIENT    LOWERLIMIT     UPPERLIMIT''
     .   //)')
         HEAD2=1
         LINEP=8
       ENDIF
           IF(ABS(LWBD).GE.1E7.AND.ABS(UPBD).GE.1E8)THEN
               WRITE(6,'(6X,''A('',I1,'','',I1,'')   =      ''
     .         ''NO LIMIT       NO LIMIT'')')CONSTR,COL
             ELSEIF(ABS(LWBD).GE.1E7)THEN
               WRITE(6,'(6X,''A('',I1,'','',I1,'')   =      ''
     .         ''NO LIMIT'',F15.5)')CONSTR,COL,UPBD
             ELSEIF(ABS(UPBD).GE.1E8)THEN
               WRITE(6,'(6X,''A('',I1,'','',I1,'')   ='',F14.5,
     .         ''       NO LIMIT'')')CONSTR,COL,LWBD
             ELSE
               WRITE(6,'(6X,''A('',I1,'','',I1,'')   ='',F14.5,
     .         F15.5)')CONSTR,COL,LWBD,UPBD
```

```
              ENDIF
            IF(ILL2.EQ.1)THEN
              WRITE(6,'(10X,''MAY BE ILL-CONDITIONED''/)')
              ILL2=0
              LINEP=LINEP+2
            ENDIF
            LINEP=LINEP+1
          ENDIF
4220    CONTINUE
4210  CONTINUE
      LINES = 0
      IF (SELOUT .EQ. 'S' .OR. SELOUT .EQ.'B') THEN
          PAUSE
          WRITE(1,'(A1)') CHAR(12)
          WRITE(1,'(/,40(''*''))')
          WRITE(1,'(''COEFFICIENT    LOWERLIMIT''
     .    ''     UPPERLIMIT''/)')
      ENDIF
      IF (SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B' ) THEN
        IF(LINEP + K .GT. 43)THEN
            WRITE(6,'(A1)')CHAR(12)
            LINEP = 0
        ENDIF
        WRITE(6,'(////,15X,A10,/,47(''*''))')FN
        WRITE(6,'(4X,''COEFFICIENT    LOWERLIMIT''
     .  ''     UPPERLIMIT''/)')
      ENDIF
C     OBJECTIVE FUNCTION COEFFICIENT RANGING IS DETERMINED FOR EACH
C     COLUMN IN THE ORIGINAL PROBLEM.
      DO 4240 COL = 1,V
        CLOWER=-10E8
        CUPPER= 10E8
C       DIFFERENT ALGORITHMS ARE USED DEPENDING ON WHETHER OR NOT THE
C       COLUMN'S VARIABLE IS IN THE BASIS.
        IF(BASIC(COL) .EQ. 1)THEN
            DO 4270 CONSTR=1,K
C             THE CONSTRAINT IS FOUND WHICH HAS THE VARIABLE IN THE
C             BASIS ASSOCIATED WITH THE CURRENT COLUMN.
              IF(CB(CONSTR).EQ.COL)THEN
                DO 4280 J=1,VT
C                 ALL COLUMNS WITH VARIABLES NOT IN THE BASIS AND NOT
C                 INCLUDING THE CURRENT COLUMN ARE CHECKED TO FIND THE
C                 CHANGE (IF ANY) WHICH WOULD DRIVE THE OBJECTIVE
C                 FUNCTION COEFFICIENT TO ZERO.
                  IF(BASIC(J).NE.1)THEN
                    IF(J .NE.COL)THEN
                      IF(ABS(AF(CONSTR,J)) .LT. .00001)THEN
                          TEMP=10E8
                          TEMP=-10E8
                        ELSE
                          IF(MXMN .EQ.2)THEN
                              TEMP=CF(J)/AF(CONSTR,J)
```

```
                      ELSE
                        TEMP=-CF(J)/AF(CONSTR,J)
                      ENDIF
                    ENDIF
C                   MINIMUM VALUES ARE RETAINED.
                    IF(TEMP.GT.0)THEN
                      CUPPER=AMIN1(CUPPER,TEMP)
                     ELSE
                      CLOWER=AMAX1(CLOWER,TEMP)
                    ENDIF
                   ENDIF
                  ENDIF
4280            CONTINUE
               ENDIF
4270         CONTINUE
            ELSE
C            VALUES ARE FOUND FOR COLUMNS NOT ASSOCIATED WITH THE BASIS.
             IF(MXMN .EQ. 2)THEN
               CLOWER=-CF(COL)
              ELSE
               CUPPER= CF(COL)
             ENDIF
           ENDIF
           CLOWER=CO(COL)+CLOWER
           CUPPER=CO(COL)+CUPPER
C          THE RESULTS ARE PRINTED.
           IF (SELOUT .EQ. 'S' .OR. SELOUT .EQ. 'B') THEN
             IF(ABS(CLOWER).GE.1E7.AND.ABS(CUPPER).GE.1E7)THEN
                WRITE(1,'(4X,''C('',I1,'')  =      NO LIMIT       NO''
     .          '' LIMIT'')')COL
              ELSEIF(ABS(CLOWER).GE.1E7)THEN
                WRITE(1,'(4X,''C('',I1,'')  =      NO LIMIT '',F14.5)'
     .          )COL,CUPPER
              ELSEIF(ABS(CUPPER).GE.1E7)THEN
                WRITE(1,'(4X,''C('',I1,'')  ='',F14.5,
     .          ''       NO LIMIT'')')COL,CLOWER
              ELSE
                WRITE(1,'(4X,''C('',I1,'')  ='',F14.5,1X,F14.5)')
     .          COL,CLOWER,CUPPER
             ENDIF
             LINES = LINES + 1
             IF (LINES .GT. 8 ) THEN
              LINES = 0
              PAUSE 4
              WRITE(1,'(A1)') CHAR(12)
              WRITE(1,'(''COEFFICIENT  LOWERLIMIT''
     .        ''  UPPERLIMIT''/)')
             ENDIF
           ENDIF
           IF(SELOUT .EQ.'P' .OR. SELOUT .EQ.'B')THEN
             IF(ABS(CLOWER).GE.1E7.AND.ABS(CUPPER).GE.1E7)THEN
                WRITE(6,'(8X,''C('',I1,'')  =      NO LIMIT       NO''
```

```
     .        '' LIMIT'')')COL
            ELSEIF(ABS(CLOWER).GE.1E7)THEN
              WRITE(6,'(8X,''C('',I1,'')  =      NO LIMIT '',F14.5)'
     .        )COL,CUPPER
            ELSEIF(ABS(CUPPER).GE.1E7)THEN
              WRITE(6,'(8X,''C('',I1,'')  ='',F14.5,
     .        ''        NO LIMIT'')')COL,CLOWER
            ELSE
              WRITE(6,'(8X,''C('',I1,'')  ='',F14.5,1X,F14.5)')
     .        COL,CLOWER,CUPPER
          ENDIF
        ENDIF
4240  CONTINUE
      IF(SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B')THEN
        WRITE(6,'(A1)')CHAR(12)
      ENDIF
      IF(SELOUT.EQ.'S'.OR.SELOUT.EQ.'B')THEN
        PAUSE
      ENDIF
      RETURN
      END
```

```
C ***************************************
C                                                                 *
C  MODULE 4 UNIT43                                                *
C    UNIT $USES: UNIT 47                                          *
C                                                                 *
C  SUBROUTINE MULCNG                                              *
C  USE: THIS SUBROUTINE ACCEPTS MULTIPLE CHANGES TO ANY OR ALL VALUES *
C       OF THE ORIGINAL PROBLEM.  IT THEN DETERMINES THE TOTAL EFFECT *
C       ON THE FINAL TABLEAU.                                     *
C                                                                 *
C  CALLED BY: PROGRAM MAINSA                                      *
C  CALLS    : SUBROUTINE CHECK2                                   *
C             SUBROUTINE CHECK                                    *
C             SUBROUTINE CONMUL                                   *
C                                                                 *
C  VARIABLES:                                                     *
C  USES     : V,K,IFLAG,INEQ,CB,VT,NEG,AO,AF,BO,BF,CO,CF,Z,INDEXL  *
C  MODIFIES : NEWA(20,20),NEWB(20),NEWCJ(20),DELTAA(20,20),DELTAB(20) *
C             DELTAC(20)                                          *
C                                                                 *
C ***************************************
$USES UCHECK2 IN UNIT47.CODE OVERLAY
      SUBROUTINE MULCNG
      INTEGER I,V,K,J,COL,ROW,IFLAG,INEQ,
     .CONSTR,CB,VT,NEG
      REAL AO,AF,BO,BF,CO,CF,Z,RM,
     .NEWA(20,20),NEWB(20),NEWCJ(20),
     .DELTAA(20,20),DELTAB(20),DELTAC(20)
      CHARACTER SELOUT,SELINP(10)*1,P(10)*1,FN*10
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,RM
$INCLUDE COMVAR
      IF(SELOUT .EQ.'P'.OR. SELOUT .EQ.'B')THEN
        WRITE(6,'(/,15X,A10)')FN
      ENDIF
      WRITE(1,'(A1)') CHAR(12)
      WRITE(1,'(5(/),
     .,2X,''THIS PROGRAM ACCEPTS MULTIPLE CHANGES''/
     .,2X,''TO A FINAL TABLEAU AND CHECKS WHETHER''/
     .,2X,''OR NOT THE CURRENT SOLUTION IS OPTIMAL''/
     .,2X,''FOR THE NEW PARAMETERS''/)')
      PAUSE
      WRITE(1,'(A1,8(/),''************STANDBY************'')') CHAR(12)
C     PERTINENT VARIABLES ARE SET TO ZERO.
      DO 4380 COL=1,V
        DELTAC(COL)=0.0
        NEWCJ(COL)=0.0
4380  CONTINUE
      DO 4305 CONSTR=1,K
        DELTAB(CONSTR)=0.0
        NEWB(CONSTR)=0.0
        DO 4390 COL=1,V
```

```
         DELTAA(CONSTR,COL)=0.0
         NEWA(CONSTR,COL)=0.0
4390    CONTINUE
4305  CONTINUE
4370  WRITE(1,'(A1)') CHAR(12)
C     THE USER CAN SELECT ANY OF THE THREE TYPES OF CHANGES.  WHEN ALL
C     CHANGES HAVE BEEN COMPLETED, THE PROGRAM MOVES TO THE SOLVING
C     PHASE.
      WRITE(1,'(3X,''SELECT THE PARAMETERS TO BE CHANGED''//,
     .6X,''1) C(J)''//,
     .6X,''2) A(I,J)''//,
     .6X,''3) B(I)''//,
     .6X,''4) CHANGES COMPLETE''//,
     .6X,''5) RETURN TO MAIN MENU''/)')
      READ(5,'(A1)') SELINP(1)
      WRITE(1,'(A1)') CHAR(12)
      IF (SELINP(1) .EQ. '1') THEN
C       INPUT OF CHANGES TO THE OBJECTIVE FUNCTION COEFFICIENTS.  EACH
C       INPUT COLUMN IS CHECKED FOR VALIDITY AS IS THE NEW VALUE OF THE
C       COEFFICIENT.
4315    WRITE(1,'(5(/),5X,''PLEASE ENTER THE COLUMN''/,
     .  5X,''TO BE CHANGED''//,
     .  5X,''PRESS D)ONE IF COMPLETE''/)')
        READ(5,'(2A1)') SELINP(1),SELINP(2)
        IF (SELINP(1) .EQ.'D' ) THEN
          GOTO 4370
        ENDIF
        CALL CHECK2(SELINP,2,V,INVAL,COL)
        IF (INVAL .EQ. 1 ) THEN
          WRITE(1,'(A1)') CHAR(12)
          WRITE(1,'(2(/),5X,''INVALID RESPONSE, PLEASE REENTER''/)')
          GOTO 4315
        ENDIF
4302    WRITE(1,'(5X,''THE ORIGINAL VALUE OF C('',I2,'') WAS ''/)')COL
        WRITE(1,'(10X,F10.3)') CO(COL)
        WRITE(1,'(/,5X,''PLEASE ENTER NEW VALUE'')')
        READ(5,'(10A1)') (P(L),L=1,10)
        CALL CHECK(P,INVAL,NEWCJ(COL))
        IF(INVAL .EQ. 1)THEN
          WRITE(1,'(A1)')CHAR(12)
          WRITE(1,'(//,''INVALID RESPONSE, PLEASE REENTER'')')
          GOTO 4302
        ENDIF
        IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN
          WRITE(6,'(''THE OLD VALUE OF C('',I2,'')     WAS '',F10.3,
     .    ''---THE NEW VALUE IS '',F15.8)')COL,CO(COL),NEWCJ(COL)
        ENDIF
C       THE CHANGE IS DETERMINED.
        DELTAC(COL)=NEWCJ(COL)-CO(COL)
        WRITE(1,'(A1,//)') CHAR(12)
        GOTO 4315
      ELSEIF (SELINP(1) .EQ.'2') THEN
```

```
C       CHANGES TO THE CONSTRAINT COEFFICIENT ARE DETERMINED.
4325    WRITE(1,'(5(/),3X,''PLEASE ENTER THE ROW TO BE CHANGED''/)')
        WRITE(1,'(5X,''PRESS D)ONE IF COMPLETE''/)')
        READ(5,'(2A1)') SELINP(1), SELINP(2)
        IF (SELINP(1) .EQ. 'D' ) THEN
          GOTO 4370
        ENDIF
        CALL CHECK2(SELINP,2,K,INVAL,CONSTR)
        IF (INVAL .EQ. 1) THEN
          WRITE(1,'(A1)') CHAR(12)
          WRITE(1,'(2(/),5X,''INVALID RESPONSE, PLEASE REENTER''/)')
          GOTO 4325
        ENDIF
        WRITE(1,'(A1)') CHAR(12)
4335    WRITE(1,'(5(/),2X,''PLEASE ENTER THE COLUMN TO BE CHANGED''//)')
        READ(5,'(2A1)') SELINP(1),SELINP(2)
        CALL CHECK2(SELINP,2,V,INVAL,COL)
        IF (INVAL .EQ. 1) THEN
          WRITE(1,'(A1)') CHAR(12)
          WRITE(1,'(2(/),5X,''INVALID RESPONSE, PLEASE REENTER''//)')
          GOTO 4335
        ENDIF
C       IF THE CONSTRAINT HAD BEEN MULTIPLIED BY MINUS 1, THE VALUES
C       ARE SWITCHED BACK.
        IF(NEG(CONSTR).EQ.1)THEN
          AO(CONSTR,COL)=-AO(CONSTR,COL)
        ENDIF
4312    IF (CONSTR.LE. 9) THEN
            IF (COL .LE. 9 ) THEN
                WRITE(1,'(4X,''THE ORIGINAL VALUE OF A('',I1,'','',I1,
     .          '') WAS '',/,11X,F15.7)') CONSTR,COL,AO(CONSTR,COL)
              ELSE
                WRITE(1,'(4X,''THE ORIGINAL VALUE OF A('',I1,'','',I2
     .          '') WAS '',/,11X,F15.7)') CONSTR,COL,AO(CONSTR,COL)
            ENDIF
          ELSE
            IF (COL .LE. 9) THEN
                WRITE(1,'(4X,''THE ORIGINAL VALUE OF A('',I2,'','',I1,
     .          '') WAS '',/,11X, F15.7)') CONSTR,COL,AO(CONSTR,COL)
              ELSE
                WRITE(1,'(3X,''THE ORIGINAL VALUE OF A('',I2,'','',I2,
     .          '') WAS '',/,11X,F15.7)') CONSTR,COL,AO(CONSTR,COL)
            ENDIF
        ENDIF
        WRITE(1,'(/,3X,''ENTER NEW VALUE (10 CHARACTERS MAX)''/)')
        READ(5,'(10A1)') (P(L),L=1,10)
        CALL CHECK(P,INVAL,NEWA(CONSTR,COL))
        IF(INVAL .EQ. 1)THEN
          WRITE(1,'(A1)')CHAR(12)
          WRITE(1,'(//,''INVALID RESPONSE, PLEASE REENTER''//)')
          GOTO 4312
        ENDIF
```

```
      IF(SELOUT.EQ.'P'.OR.SELOUT.EQ.'B')THEN
        IF(CONSTR .LE.9)THEN
          IF(COL .LE. 9)THEN
            WRITE(6,'(''THE OLD VALUE OF A('',I1,'','',I1,
     .      '') WAS '',F14.5,''---THE NEW VALUE IS '',F15.7)')
     .      CONSTR,COL,AO(CONSTR,COL),NEWA(CONSTR,COL)
          ELSE
            WRITE(6,'(''THE OLD VALUE OF A('',I1,'','',I2,
     .      '') WAS '',F13.5,''---THE NEW VALUE IS '',F15.7)')
     .      CONSTR,COL,AO(CONSTR,COL),NEWA(CONSTR,COL)
          ENDIF
        ELSEIF(COL .LE. 9)THEN
            WRITE(6,'(''THE OLD VALUE OF A('',I2,'','',I1,
     .      '') WAS '',F13.5,''---THE NEW VALUE IS '',F15.7)')
     .      CONSTR,COL,AO(CONSTR,COL),NEWA(CONSTR,COL)
        ELSE
            WRITE(6,'(''THE OLD VALUE OF A('',I2,'','',I2,
     .      '') WAS '',F12.5,''---THE NEW VALUE IS '',F15.7)')
     .      CONSTR,COL,AO(CONSTR,COL),NEWA(CONSTR,COL)
        ENDIF
      ENDIF
      IF(NEG(CONSTR).EQ.1)THEN
        AO(CONSTR,COL)=-AO(CONSTR,COL)
        NEWA(CONSTR,COL)=-NEWA(CONSTR,COL)
      ENDIF
      DELTAA(CONSTR,COL)=NEWA(CONSTR,COL)-AO(CONSTR,COL)
      WRITE(1,'(A1)') CHAR(12)
      GOTO 4325
     ELSEIF (SELINP(1) .EQ.'3') THEN
C     CHANGES TO THE RIGHT-HAND SIDE ARE FOUND.
4345  WRITE(1,'(5(/), 5X,''PLEASE ENTER THE ROW TO BE CHANGED''/,
     . 5X,''PRESS D)ONE IF COMPLETE'')')
      READ(5,'(2A1)') SELINP(1),SELINP(2)
      IF (SELINP(1).EQ. 'D') THEN
        GOTO 4370
      ENDIF
      CALL CHECK2(SELINP,2,K,INVAL,CONSTR)
      IF (INVAL .EQ. 1) THEN
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(2(/),5X,''INVALID RESPONSE, PLEASE REENTER''/)')
        GOTO 4345
      ENDIF
      IF(NEG(CONSTR).EQ.1)THEN
        BO(CONSTR)=-BO(CONSTR)
      ENDIF
4322  WRITE(1,'(5X,''THE ORIGINAL VALUE OF B('',I2,'') WAS ''/)')
     . CONSTR
      WRITE(1,'(10X,F10.3)') BO(CONSTR)
      WRITE(1,'(/,5X,''ENTER NEW VALUE (10 CHARACTERS MAX)''/)')
      READ(5,'(10A1)') (P(L),L=1,10)
      CALL CHECK(P,INVAL,NEWB(CONSTR))
      IF(INVAL .EQ. 1)THEN
```

```
          WRITE(1,'(A1)')CHAR(12)
          WRITE(1,'(//,''INVALID RESPONSE, PLEASE REENTER''//)')
          GOTO 4322
        ENDIF
        IF(SELOUT.EQ.'F'.OR.SELOUT.EQ.'B')THEN
          WRITE(6,'(''THE OLD VALUE OF B('',I2,'')    WAS '',F10.3,
     .    ''---THE NEW VALUE IS '',F10.3)')CONSTR,BO(CONSTR),
     .    NEWB(CONSTR)
        ENDIF
        IF(NEG(CONSTR).EQ.1)THEN
          BO(CONSTR)=-BO(CONSTR)
          NEWB(CONSTR)=-NEWB(CONSTR)
        ENDIF
        DELTAB(CONSTR)=NEWB(CONSTR)-BO(CONSTR)
        WRITE(1,'(A1//)') CHAR(12)
        GOTO 4345
      ELSEIF (SELINP(1).EQ.'4') THEN
C       WHEN ALL CHANGES HAVE BEEN ENTERED, THE NET EFFECT OF THESE
C       CHANGES ON THE FINAL TABLEAU IS DETERMINED.  THIS GENERALLY
C       CONSISTS OF MATRIX MULTIPLICATION.  THE FINAL TABLEAU B-INVERSE
C       (PLUS OBJECTIVE FUNCTION COEFFICIENT) IS MULTIPLIED BY THE
C       MATRIX CONTAINING THE CUMULATIVE CHANGES.
        DO 2000 CONSTR=1,K
          Z=Z+DELTAB(CONSTR)*CF(CONSTR+INDEXL-1)
          DO 2010 ROW=1,K
            BF(CONSTR)=BF(CONSTR)+DELTAB(ROW)*AF(CONSTR,ROW+INDEXL-1)
2010      CONTINUE
2000    CONTINUE
        DO 2020 COL=1,V
          IF(MXMN .EQ. 2)THEN
              CF(COL)=CF(COL)+DELTAC(COL)
            ELSE
              CF(COL)=CF(COL)-DELTAC(COL)
          ENDIF
          DO 2030 CONSTR=1,K
            CF(COL)=CF(COL)+DELTAA(CONSTR,COL)*CF(CONSTR+INDEXL-1)
2030      CONTINUE
2020    CONTINUE
        DO 2040 CONSTR=1,K
          DO 2050 COL=1,V
            DO 2060 I=1,K
              AF(CONSTR,COL)=AF(CONSTR,COL)+DELTAA(I,COL)*
     .        AF(CONSTR,I+INDEXL-1)
2060        CONTINUE
2050      CONTINUE
2040    CONTINUE
      ELSEIF (SELINP (1) .EQ. '5') THEN
        RETURN
      ELSE
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(5(/),5X,''INVALID RESPONSE, PLEASE REENTER'')')
        GOTO 4370
```

```
ENDIF
CALL CONMUL
RETURN
END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C                                                                     *
C  MODULE 4 UNIT43                                                    *
C    UNIT $USES: NONE                                                 *
C                                                                     *
C  SUBROUTINE: CONMUL                                                 *
C  USE: A CONTINUATION OF SUBROUTINE MULCNG.  THIS SUBROUTINE HAS     *
C       BEEN SEPARATED TO ALLOW COMPILATION.  THIS SUBROUTINE         *
C       RECEIVES THE MODIFIED FINAL TABLEAU FROM MULCNG AND  CHANGES  *
C       IT INTO THE BASIC SOLUTION FORM                               *
C                                                                     *
C  CALLED BY: SUBROUTINE MULCNG                                       *
C  CALLS    : NONE                                                    *
C                                                                     *
C  VARIABLES:                                                         *
C  USES     : AF,BF,CF,Z,V,VT,CB,INEQ                                 *
C  MODIFIES : AF,BF,CF,Z                                              *
C                                                                     *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE CONMUL
C     THIS SUBROUTINE IS A CONTINUATION OF THE ABOVE SUBROUTINE MERELY
C     SEPARATED TO ALLOW COMPILATION.
      INTEGER CONSTR,COL,IFLAG,VT,ROW,CB,V
      REAL AO,AF,BO,BF,CO,CF,Z,NEWB(20),BM
      CHARACTER SELOUT*1,FN*10
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
$INCLUDE CONVAR
      IFLAG(9)=0
      DO 4303 CONSTR=1,K
C       THE MODIFIED FINAL TABLEAU IS NOW RETURNED TO THE BASIC
C       SOLUTION FORM.
        NEWB(1)=AF(CONSTR,CB(CONSTR))
        IF(ABS(NEWB(1)).LT..01)THEN
          IFLAG(9)=1
          RETURN
        ENDIF
        DO 4313 COL=1,VT
C         EACH CONSTRAINT IS DIVIDED BY THE VALUE OF THE COEFFICIENT
C         IN THE COLUMN ASSOCIATED WITH THE VARIABLE IN THE BASIS.
C         THIS WILL RETURN THE COEFFICIENT TO A VALUE OF ONE (DIVIDED
C         BY ITSELF).
          AF(CONSTR,COL)=AF(CONSTR,COL)/NEWB(1)
4313    CONTINUE
        BF(CONSTR)=BF(CONSTR)/NEWB(1)
        DO 4323 ROW=1,K
C         THIS CONSTRAINT IS THEN ADDED TO EACH OTHER CONSTRAINT AND
C         THE OBJECTIVE FUNCTION IN THE REQUIRED MULTIPLES TO DRIVE ALL
C         OTHER VALUES IN THE COLUMN TO ZERO.
          NEWB(1)=-AF(ROW,CB(CONSTR))
          IF (ROW .NE. CONSTR) THEN
            DO 4333 COL=1,VT
```

```
              AF(ROW,COL)=AF(ROW,COL)+NEWB(1)*AF(CONSTR,COL)
4333        CONTINUE
            BF(ROW)=BF(ROW)+NEWB(1)*BF(CONSTR)
          ENDIF
4323    CONTINUE
        IF(CB(CONSTR).LT.INDEXE)THEN
          NEWB(1)=-CF(CB(CONSTR))
          DO 4343 COL=1,VT
            CF(COL)=CF(COL)+NEWB(1)*AF(CONSTR,COL)
4343      CONTINUE
          Z=Z+NEWB(1)*BF(CONSTR)
        ENDIF
4303  CONTINUE
      NEWB(1)=0.0
      DO 4377 CONSTR=1,K
C       IF BIG M HAD BEEN SUBTRACTED FROM SOME OBJECTIVE FUNCTION
C       COEFFICIENTS IN UNIT 48, THEY ARE ADDED BACK.
        IF(INEQ(CONSTR) .NE. 0)THEN
          CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)+BM
        ENDIF
4377  CONTINUE
C     A TEST IS MADE FOR OPTIMALITY OF THE NEW BASIC TABLEAU.
      DO 4353 COL=1,VT
        IF(CF(COL) .LT.- 0.00001)THEN
          NEWB(1)=100.
        ENDIF
4353  CONTINUE
      DO 4363 CONSTR=1.K
        IF(BF(CONSTR) .LT.- 0.00001) THEN
          NEWB(1)=100
        ENDIF
4363  CONTINUE
      IF(NEWB(1) .GT. 99.) THEN
          DO 4373 CONSTR=1,K
            IF(CB(CONSTR).GE.INDEXE)THEN
              IF(BF(CONSTR).GT.0)THEN
                WRITE(1,'(5(/),5X,''*******NOT FEASIBLE*******'')')
                IFLAG(3)=1
                RETURN
              ENDIF
            ENDIF
4373      CONTINUE
          WRITE(1.'(5(/),5X,''*********NOT OPTIMAL*********'')')
        ELSE
          WRITE(1,'(5(/),5X,''********STILL OPTIMAL********'')')
      ENDIF
      PAUSE
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C                                                                         *
C  MODULE 4 UNIT44                                                        *
C    UNIT $USES: UNIT 47                                                  *
C                                                                         *
C  SUBROUTINE ADDCON                                                      *
C  USE: THIS SUBROUTINE ALLOWS AN ADDITIONAL CONSTRAINT OR VARIABLE       *
C       TO BE ADDED TO A PROBLEM WHICH HAS ALREADY BEEN SOLVED BY         *
C       MODULES 2 OR 3.  THE EFFECTS OF THE ADDITION ARE CALCULATED       *
C       AND THE NEW TABLEAU IS PUT INTO A BASIC SOLUTION.                 *
C                                                                         *
C  CALLED BY: PROGRAM MAINSA                                              *
C  CALLS    : SUBROUTINE CHECK2                                           *
C             SUBROUTINE CHECK                                            *
C                                                                         *
C  VARIABLES:                                                             *
C  USES     : V,K,VT,IFLAG,INEQ,MXMN,CB,NEG,NEC,NLC,NGC,AO,AF,BO,BF,      *
C             CO,CF,Z,INDEXL,BM                                           *
C  MODIFIES : INDEXL,V,VT,K,AF,BF,CF,Z,DELTAA,NLC,NGC,NEC                 *
C                                                                         *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$USES UCHECK2 IN UNIT47.CODE OVERLAY
      SUBROUTINE  ADDCON
      INTEGER I,V,K,J,COL,VT,ROW,IFLAG,INEQ,MXMN,NEC,NGC,NLC,
     .CONSTR,CB
      REAL AO,AF,BO,BF,CO,CF,Z,DELTAA(20),BM
      CHARACTER SELINP(2)*1,FN*10,SELOUT*1,P(10)*1
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
$INCLUDE COMVAR
      IF (V.EQ.20 .OR. K .EQ. 20 ) THEN
        WRITE(1,'(5(/),5X,''THIS PROBLEM IS TOO LARGE FOR'',/,4X
     .  ''ADDITIONS. '')')
        RETURN
      ENDIF
      WRITE(1,'(A1)') CHAR(12)
      WRITE(1,'(///,3X
     .,''THIS SEGMENT ALLOWS YOU TO ADD AN'',/,3X
     .,''ADDITIONAL CONSTRAINT OR VARIABLE TO '',)')
      WRITE(1,'(3X,''AN ALREADY SOLVED LINEAR PROGRAMMING '',/,3X
     .,''PROBLEM'',///)')
4400  WRITE(1,'(9X,''DO YOU WISH TO ADD A:'',//,14X
     .,              ''C)ONSTRAINT'',/,18X
     .,                 ''OR''        )')
      WRITE(1,'(14X,  ''V)ARIABLE''  ,//,10X
     .,              ''SELECT ''''C'''' OR ''''V'''' '',//)')
      READ(5,'(A1)') SELINP(1)
      IF (SELINP(1) .EQ. 'V') THEN
C       IF A VARIABLE IS ADDED, IT IS PLACED JUST AFTER THE LAST
C       VARIABLE IN THE ORIGINAL PROBLEM.  ALL VARIABLES TO THE RIGHT
C       ARE MOVED ONE COLUMN TO THE RIGHT.  ALL INDICES ARE RESET.
         VT=VT+1
```

```
          V=V+1
          INDEXL=INDEXL+1
          INDEXG=INDEXG+1
          INDEXE=INDEXE+1
          DO 4420 COL=VT,V+1,-1
            CF(COL)=CF(COL-1)
            DO 4410 CONSTR=1,K
              AF(CONSTR,COL)=AF(CONSTR,COL-1)
4410        CONTINUE
4420      CONTINUE
          DO 4485 CONSTR=1,K
            IF (CB(CONSTR) .GE. V )THEN
              CB(CONSTR)=CB(CONSTR)+1
            ENDIF
4485      CONTINUE
          WRITE(1,'(A1,///)') CHAR(12)
4411      WRITE(1,'(//,4X,''PLEASE ENTER THE COEFFICIENT FOR'',/,9X
     .    ''THE OBJECTIVE FUNCTION'',///)')
          IF(V .LT.10)THEN
              WRITE(1,'(/,8X,''C('',I1,'') = '',$)')V
            ELSE
              WRITE(1,'(/,7X,''C('',I2,'') = '',$)')V
          ENDIF
          READ(5,'(10A1)') (P(L),L=1,10)
C         THE INPUTS ARE CHECKED FOR VALIDITY.
          CALL CHECK(P,INVAL,CO(V))
          IF(INVAL .EQ. 1)THEN
            WRITE(1,'(A1)') CHAR(12)
            WRITE(1,'(//,,5X,''INVALID RESPONSE, PLEASE REENTER'')')
            GOTO 4411
          ENDIF
          WRITE(1,'(A1,///)') CHAR(12)
4412      WRITE(1,'(//,4X,''PLEASE ENTER THE COEFFICIENT FOR'',/,4X
     .    ,''EACH CONSTRAINT'',//)')
          DO 4430 CONSTR=1,K
            WRITE(1,'(6X,''A('',I2,'','',I2,'') = '',$)') CONSTR,V
            READ(5,'(10A1)') (P(L),L=1,10)
            CALL CHECK(P,INVAL,AO(CONSTR,V))
            IF(INVAL .EQ.1)THEN
              WRITE(1,'(A1)')CHAR(12)
              WRITE(1,'(//,4X,''INVALID RESPONSE, PLEASE REENTER'',//)')
              GOTO 4412
            ENDIF
            IF(NEG(CONSTR) .EQ. 1)THEN
              AO(CONSTR,V)=-AO(CONSTR,V)
            ENDIF
4430      CONTINUE
          CF(V)=-CO(V)
C         THE NEW ADDITIONS ARE MULTIPLIED BY B-INVERSE TO GET A
C         MODIFIED FINAL TABLEAU.
          DO 4495 CONSTR=1,K
            CF(V)=CF(V)+AO(CONSTR,V)*CF(CONSTR+INDEXL-1)
```

```
            DELTAA(CONSTR)=0.0
4495      CONTINUE
          DO 4497 CONSTR=1,K
            DO 4499 I=1,K
              DELTAA(CONSTR)=DELTAA(CONSTR)+AO(I,V)*AF(CONSTR,
     .        I+INDEXL-1)
4499        CONTINUE
4497      CONTINUE
          DO 4498 CONSTR=1,K
            AF(CONSTR,V)=DELTAA(CONSTR)
4498      CONTINUE
C         IF UNIT48 REMOVED THE BIG M VALUE, IT IS ADDED BACK.
          DO 4476 CONSTR=1,K
            IF(INEQ(K) .NE. 0)THEN
              CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)+BM
            ENDIF
4476      CONTINUE
        ELSEIF (SELINP(1) .EQ. 'C') THEN
C         IF A CONSTRAINT IS ADDED, IT IS PLACED AT THE BOTTOM.
C         INDICES ARE RESET AS REQUIRED.  THE COLUMN OR COLUMNS
C         REQUIRED FOR THE NEW CONSTRAINT ARE ADDED TO THE RIGHT OF THE
C         EXISTING COLUMNS.
          K=K+1
          WRITE(1,'(A1)') CHAR(12)
4413      WRITE(1,'(//,5X,''PLEASE ENTER THE NEW CONSTRAINT'')')
          DO 4440 COL=1,V
            WRITE(1,'(/,10X,''A('',I2,'','',I2,'') = ''$)') K,COL
            READ(5,'(10A1)') (P(L),L=1,10)
            CALL CHECK(P,INVAL,AF(K,COL))
            IF(INVAL .EQ. 1)THEN
              WRITE(1,'(A1)') CHAR(12)
              WRITE(1,'(//,4X,''INVALID RESPONSE.PLEASE REENTER'')')
              GOTO 4413
            ENDIF
4440      CONTINUE
          WRITE(1,'(A1,///)')CHAR(12)
4441      WRITE(1,'(//,4X,''IS CONSTRAINT OF THE FORM ''//)')
          WRITE(1,'(10X,''1) LESS THAN'',//,10X
     .    ''2) GREATER THAN'',//,10X
     .    ,9X,''OR'',//,10X,
     .    ''3) EQUALS'',/)')
          READ(5,'(I1)') SELINP(1)
          CALL CHECK2(SELINP,1,3,INVAL,INEQ(K))
          IF(INVAL .EQ. 1)THEN
            WRITE(1,'(A1)')CHAR(12)
            WRITE(1,'(//,4X,''INVALID RESPONSE, PLEASE REENTER'')')
            GOTO 4441
          ENDIF
          INEQ(K)=INEQ(K)-1
          WRITE(1,'(A1,///)') CHAR(12)
4414      WRITE(1,'(//,4X,''PLEASE ENTER THE RIGHT HAND SIDE '',//)')
          READ(5,'(10A1)')(P(L),L=1,10)
```

```
      CALL CHECK(P,INVAL,BF(K))
      IF(INVAL .EQ.1)THEN
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(//,4X,''INVALID RESPONSE, PLEASE REENTER'')')
        GOTO 4414
      ENDIF
      TEMP=0.0
      DO 4439  COL=V+1,VT
        AF(K,COL)=0.0
4439  CONTINUE
      DO 4444 CONSTR=1,K-1
        TEMP=TEMP+BF(CONSTR)*AF(K,CB(CONSTR))
4444  CONTINUE
      IF(INEQ(K) .EQ. 0)THEN
          IF(TEMP .GT. BF(K))THEN
            GOTO 4401
          ENDIF
        ELSEIF(INEQ(K) .EQ. 1    )THEN
          IF(TEMP .LT. BF(K))THEN
            GOTO 4401
          ENDIF
        ELSE
          IF(TEMP .NE. BF(K))THEN
            GOTO 4401
          ENDIF
      ENDIF
      WRITE(1,'(A1)') CHAR(12)
      WRITE(1,'(////,''THE NEW CONSTRAINT WAS SATISFIED BY THE'')')
      WRITE(1,'(/,11X,''ORIGINAL SOLUTION'')')
      IFLAG(2)=1
      PAUSE
      RETURN
4401  IF(INEQ(K) .EQ. 0) THEN
C       IF THE NEW CONSTRAINT IS A "LESS THAN", A SINGLE COLUMN IS
C       ADDED.
          VT=VT+1
          NLC=NLC+1
          DO 4450 CONSTR=1,K-1
            AF(CONSTR,VT)=0.0
4450      CONTINUE
          DO 4460 COL =V+1,VT-1
            AF(K,COL)=0.0
4460      CONTINUE
          AF(K,VT)=1.0
          CB(K)=VT
          DO 4477 CONSTR=1,K
            IF(INEQ(CONSTR) .NE. 0)THEN
              CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)+BM
            ENDIF
4477      CONTINUE
        ELSEIF(INEQ(K) .EQ. 1) THEN
C         IF THE NEW CONSTRAINT WAS A "GREATER THAN", TWO COLUMNS
```

```
C             ARE ADDED AND THE INDICES ARE ADJUSTED ACCORDINGLY.
              VT=VT+2
              NGC=NGC+1
              NEC=NEC+1
              DO 4480 COL=VT-1,VT
                CF(COL)=0.0
                DO 4470 CONSTR=1,K-1
                  AF(CONSTR,COL)=0.0
4470            CONTINUE
4480          CONTINUE
              DO 4481 COL=V+1,VT-2
                AF(K,CONSTR)=0.0
4481          CONTINUE
              AF(K,VT-1)=-1.0
              AF(K,VT)=1.0
              CB(K)=VT
              DO 4490 COL=1,VT
                CF(COL)=CF(COL)-AF(K,COL)*BM
4490          CONTINUE
              Z=Z-BF(K)*BM
              CF(VT)=BM
              DO 4478 CONSTR=1,K-1
                IF(INEQ(CONSTR) .NE. 0)THEN
                  CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)+BM
                ENDIF
4478          CONTINUE
            ELSEIF(INEQ(K) .EQ. 2) THEN
C             IF THE NEW CONSTRAINT IS AN "EQUALS", A SINGLE ARTIFICIAL
C             VARIABLE IS ADDED.
              VT=VT+1
              NEC=NEC+1
              DO 4405 COL=V+1,VT-1
                AF(K,COL)=0.0
4405          CONTINUE
              DO 4415 CONSTR=1,K-1
                AF(CONSTR,VT)=0.0
4415          CONTINUE
              AF(K,VT)=1.0
              CB(K)=VT
              DO 4425 COL=1,VT
                CF(COL)=CF(COL)-AF(K,COL)*BM
4425          CONTINUE
              Z=Z-BF(K)*BM
              DO 4479 CONSTR=1,K
                IF(INEQ(CONSTR) .NE. 0)THEN
                  CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)+BM
                ENDIF
4479          CONTINUE
            ELSE
              WRITE(1,'(A1)') CHAR(12)
              WRITE(1,'(///,5X,''IMPROPER RESPONSE'',///)')
              GOTO 4441
```

```
          ENDIF
C         COEFFICIENTS IN THE NEW CONSTRAINT WHICH REPRESENT COLUMNS
C         WITH VARIABLES IN THE BASIS ARE DRIVEN TO ZERO, AND THE FULL
C         CONSTRAINT AND RHS ARE ADJUSTED ACCORDINGLY.
          DO 4435 CONSTR=1,K-1
            TEMP=AF(K,CB(CONSTR))
            DO 4445 COL=1,VT
              AF(K,COL)=AF(K,COL)-AF(CONSTR,COL)*TEMP
4445        CONTINUE
            BF(K)=BF(K)-BF(CONSTR)*TEMP
4435      CONTINUE
        ELSE
          WRITE(1,'(A1)') CHAR(12)
          WRITE(1,'(///,11X,''IMPROPER RESPONSE'',////)')
          GOTO 4400
      ENDIF
      J=0
      IFLAG(3)=0
C
C     IF ALL OBJECTIVE FUNCTION COEFFICIENTS AND ALL RHS ARE
C     NON-NEGITVE AND AN ARTIFICIAL VARIABLE IS IN THE BASIS AT
C     A POSITIVE LEVEL, THEN THE PROBLEM IS INFEASIBLE
C
      DO 4465 CONSTR=1,K
        IF(BF(CONSTR) .LT. -.0001)THEN
          J=1
        ENDIF
4465  CONTINUE
      DO 4475 COL=1,VT
        IF(CF(COL) .LT. -.0001)THEN
          J=1
        ENDIF
4475  CONTINUE
      DO 4493 CONSTR=1,K
        IF(J.EQ.0.AND.CB(CONSTR).GE.INDEXE)THEN
          IF(BF(CONSTR) .GT..0001)THEN
            IFLAG(3)=1
          ENDIF
        ENDIF
4493  CONTINUE
      RETURN
      END
```

```
C****************************************
C                                       *
C  MODULE 4 UNIT45                      *
C    UNIT $USES: UNIT 47                *
C                                       *
C  SUBROUTINE SOLVE                     *
C  USE: THIS SUBROUTINE ACCEPTS BASIC TABLEAUS FROM SUBROUTINES   *
C       MULCNG AND ADDCON AND DIRECTS OTHER SUBROUTINES TO DETERMINE *
C       THE FINAL SOLUTION AND THEN DISPLAY AS DIRECTED.        *
C                                       *
C  CALLED BY: SUBROUTINE SELECT         *
C  CALLS    : SUBROUTINE OPTB           *
C             SUBROUTINE WORK           *
C             SUBROUTINE TDISPL         *
C                                       *
C  VARIABLES:                           *
C  USES     : IFLAG(7),INFP,OPTS        *
C  MODIFIES : NONE                      *
C                                       *
C****************************************
$USES UCHECK2 IN UNIT47.CODE OVERLAY
      SUBROUTINE SOLVE
      INTEGER PK,PR,OPTS,V,VT,CB
      REAL AO,AF,BO,BF,CO,CF,Z,BM
      CHARACTER SELOUT,FN*10
      COMMON/P1/OPTS,KFA,PK,PR
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
      COMMON/THREE/INFP
$INCLUDE CONVAR
C     SUBROUTINE OPTB IS CALLED TO DETERMINE THE CONDITION OF THE
C     TABLEAU.  IF THE TABLEAU IS EITHER UNBOUNDED OR INFEASIBLE, THE
C     PROGRAM RETURNS TO THE MAIN MENU.  IF IT IS NEITHER OF THESE
C     BUT IS OPTIMAL, THE SUBROUTINE TDISPL IS CALLED TO DISPLAY THE
C     RESULTS.  OTHERWISE, SUBROUTINE WORK IS CALLED TO COMPLETE A
C     BASIS CHANGE.
4500  CALL OPTB
      IF(OPTS .EQ. 1 .OR. IFLAG(7) .EQ. 1)THEN
        IF(IFLAG(7) .EQ. 1 .OR. INFP .EQ. 1)THEN
          RETURN
        ENDIF
      ENDIF
      IF(OPTS .EQ. 1) THEN
          CALL TDISPL
          RETURN
        ELSE
          CALL WORK
          GOTO 4500
      ENDIF
      RETURN
      END
```

```
C ***************************************
C                                        *
C  MODULE 4 UNIT45                       *
C    UNIT $USES: NONE                    *
C                                        *
C  SUBROUTINE: OPTB                      *
C  USE: THIS SUBROUTINE DETERMINES IF A TABLEAU IS OPTIMAL,     *
C       UNBOUNDED, INFEASIBLE, MULTIPLE OPTIMAL, OR DEGENERATIVE.  *
C       IF IT IS NONE OF THESE, OTHER PARAMETERS ARE DETERMINED TO  *
C       PERFORM A BASIS CHANGE.          *
C                                        *
C  CALLED BY: SUBROUTINE SOLVE           *
C  CALLS    : NONE                       *
C                                        *
C  VARIABLES:                            *
C  USES     : V,NGC,NLC,AO,AF,BO,BF,CO,CF,Z,CB           *
C  MODIFIES : IFLAG,INFP,GNEG,KFA,PK,PR,OPTS,SPR         *
C                                        *
C ***************************************
      SUBROUTINE OPTB
      INTEGER PK,PR,OPTS,V,VT,CB
      CHARACTER SELOUT,FN*10
      COMMON/ONE/SELOUT,FN
      COMMON/P1/OPTS,KFA,PK,PR
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
      COMMON/THREE/INFP
$INCLUDE COMVAR
      IFLAG(4)=0
      IFLAG(6)=0
      IFLAG(7)=0
      IFLAG(8)=0
      IFLAG(9)=0
      OPTS=0
      INFP=0
      GNEG=0.0
      KFA=V+NGC+NLC+1
      GOTO 300
C     THE PIVOT COLUMN IS FOUND.
110   DO 130 J=1,VT
        IF (CF(J) .GE. GNEG)THEN
           GOTO 130
          ELSE
           GNEG=CF(J)
           PK=J
        ENDIF
130   CONTINUE
C     OPTIMALITY IS DETERMINED.
      IF (ABS(GNEG) .LT. 0.0001) THEN
        OPTS=1
      ENDIF
C     INFEASIBILITY IS DETERMINED.
      DO 150 I=1,K
```

```
      IF (CB(I) .LT. KFA) THEN
        IF(BF(I) .LT. -0.00001)THEN
          INFP=1
        ENDIF
      ELSEIF(BF(I) .LT. -0.00001)THEN
        GOTO 150
      ELSE
        INFP=1
      ENDIF
150   CONTINUE
C     THE LEAVING BASIC VARIABLE IS FOUND.
      IF(OPTS .EQ. 0)THEN
        SPR=10.E8
        DO 190 I=1,K
          IF(AF(I,PK) .LE. .0001) THEN
            GOTO 190
          ELSEIF (BF(I)/AF(I,PK) .GE. SPR) THEN
            GOTO 190
          ELSE
            SPR=BF(I)/AF(I,PK)
            PR=I
          ENDIF
190     CONTINUE
        IF(SPR .GE. 10.E6)THEN
          IFLAG(7)=1
        ENDIF
      ENDIF
      GOTO 500
C     DUAL PIVOTS ARE USED UNLESS A NEGATIVE OBJECTIVE FUNCTION
C     COEFFICIENT IS FOUND.
300   DO 320 J=1,VT
        IF(CF(J) .LT. -0.00001)THEN
          GOTO 110
        ENDIF
320   CONTINUE
C     THE PIVOT ROW IS FOUND.
      DO 340 I=1,K
        IF(BF(I) .GE. GNEG)THEN
          GOTO 340
        ELSE
          GNEG=BF(I)
          PR=I
        ENDIF
340   CONTINUE
      IF(ABS(GNEG) .LT. 0.0001) THEN
        OPTS=1
        GOTO 500
      ELSE
        SPR=-10.E8
        DO 370 J=1,VT
          DO 360 I=1,K
            IF(CB(I) .EQ. J)THEN
```

```
            GOTO 370
          ENDIF
360     CONTINUE
        IF(AF(PR,J) .GE. -.0001)THEN
          GOTO 370
        ELSEIF(CF(J)/AF(PR,J) .LE. SPR)THEN
          GOTO 370
        ELSE
          SPR=CF(J)/AF(PR,J)
          PK=J
        ENDIF
370   CONTINUE
    ENDIF
    IF(SPR .LE. -10.E6)THEN
      IFLAG(7)=1
    ENDIF
500 IF(OPTS .EQ. 1)THEN
      IF(INFP .EQ. 1) THEN
        GOTO 600
      ENDIF
      DO 540 J=1,VT
        IFLAG(8)=0
        DO 520 I=1,K
          IF(CB(I) .EQ. J) THEN
            IFLAG(8)=1
          ENDIF
520     CONTINUE
        IF (IFLAG(8) .EQ. 0)THEN
          IF(ABS(CF(J)) .LT. .0001 )THEN
            IFLAG(4)=1
          ENDIF
        ENDIF
540   CONTINUE
    ENDIF
    IF(IFLAG(7) .EQ. 1)THEN
      GOTO 600
    ENDIF
    DO 560 I=1,K
      IF(ABS(BF(I))  .LE. .0001)THEN
        IFLAG(6)=1
      ENDIF
560 CONTINUE
C   THE CONDITION OF THE TABLEAU IS PRINTED.
600 IF (SELOUT .EQ. 'S' .OR. SELOUT .EQ. 'B') THEN
      IF (OPTS .EQ. 1 .OR. IFLAG(7) .EQ. 1)THEN
          WRITE(1,'(10X,''FINAL TABLEAU - '',$)')
        IF(INFP .EQ. 1)THEN
            WRITE(1,'(''INFEASIBLE'')')
          ELSEIF(IFLAG(7) .EQ. 1)THEN
            WRITE(1,'(''UNBOUNDED'')')
          ELSE
            WRITE(1,'(''OPTIMAL'')')
```

```
          ENDIF
          IF(IFLAG(6) .EQ. 1)THEN
            WRITE(1,'(26X,''DEGENERATE'')')
          ENDIF
          IF(OPTS .EQ. 1 .AND. IFLAG(4) .EQ. 1)THEN
            WRITE(1,'(5X,''MULTIPLE OPTIMAL SOLUTIONS EXIST'')')
          ENDIF
          PAUSE
        ENDIF
      ENDIF
      IF (SELOUT .EQ. 'P' .OR. SELOUT .EQ. 'B') THEN
        IF (OPTS .EQ. 1 .OR. IFLAG(7) .EQ. 1)THEN
          WRITE(6,'(////,10X,''FINAL TABLEAU - '',$)')
          IF(INFP .EQ. 1)THEN
            WRITE(6,'(''INFEASIBLE'')')
          ELSEIF(IFLAG(7) .EQ. 1)THEN
            WRITE(6,'(''UNBOUNDED'')')
          ELSE
            WRITE(6,'(''OPTIMAL'')')
          ENDIF
          IF(IFLAG(6) .EQ. 1)THEN
            WRITE(6,'(26X,''DEGENERATE'')')
          ENDIF
          IF(OPTS .EQ. 1 .AND. IFLAG(4) .EQ. 1)THEN
            WRITE(6,'(5X,''MULTIPLE OPTIMAL SOLUTIONS EXIST'')')
          ENDIF
          WRITE(6,'(//)')
        ENDIF
      ENDIF
      IFLAG(9)=1
      RETURN
      END
```

```
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C                                                                             *
C  MODULE 4 UNIT45                                                            *
C    UNIT $USES: NONE                                                         *
C                                                                             *
C  SUBROUTINE: WORK                                                           *
C  USE: THIS SUBROUTINE PERFORMS A PIVOT ON A BASIC TABLEAD                   *
C       ACCORDING TO PARAMETERS DETERMINED BY SUBROUTINE OPTB.                *
C                                                                             *
C  CALLED BY: SUBROUTINE SOLVE                                                *
C  CALLS    : NONE                                                            *
C                                                                             *
C  VARIABLES:                                                                 *
C  USES     : PK,PR,OPTS,V,VT,CB                                              *
C  MODIFIES : AF,BF,CF,Z                                                      *
C                                                                             *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE WORK
      INTEGER PK,PR,OPTS,V,VT,CB,KFA
      REAL PELE,HOLD
      COMMON/P1/OPTS,KFA,PK,PR
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MYMN,BM
$INCLUDE CONVAR
C     THIS SUBROUTINE PERFORMS A BASIS CHANGE WITH A PIVOT ROW AND PIVOT
C     COLUMN DETERMINED BY SUBROUTINE OPTB.
      PELE=AF(PR,PK)
      DO 200 J=1,VT
        AF(PR,J)=AF(PR,J)/PELE
200   CONTINUE
      BF(PR)=BF(PR)/PELE
      CB(PR)=PK
      DO 300 I=1,K
        IF(I .EQ. PR)THEN
          GOTO 300
        ENDIF
        HOLD=AF(I,PK)
        DO 250 J=1,VT
          AF(I,J)=AF(I,J)-HOLD*AF(PR,J)
250     CONTINUE
        BF(I)=BF(I)-HOLD*BF(PR)
300   CONTINUE
      HOLD=CF(PK)
      DO 350 J=1,VT
        CF(J)=CF(J)-HOLD*AF(PR,J)
350   CONTINUE
      Z=Z-HOLD*BF(PR)
      RETURN
      END
```

```
C ****************************************
C                                        *
C  MODULE 4 UNIT45                       *
C    UNIT $USES: NONE                    *
C                                        *
C  SUBROUTINE: TDISPL                    *
C  USE: THIS SUBROUTINE DISPLAYS THE FINAL TABLEAU ACCORDING TO THE  *
C       FORMAT AND CONDITIONS SET BY THE USER.                       *
C                                        *
C  CALLED BY: SUBROUTINE SOLVE           *
C  CALLS    : NONE                       *
C                                        *
C  VARIABLES:                            *
C  USES     : AF,BF,CF,Z,SELOUT.FMT      *
C  MODIFIES : NONE                       *
C                                        *
C ****************************************
      SUBROUTINE TDISPL
      CHARACTER P(10)*1,OBJN*10,SELOUT,FN*10
      INTEGER PK,PR,OPTS,V,VT,CB,DUAL,T,FMT
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
      COMMON/P1/OPTS,KFA,PK,PR
$INCLUDE COMVAR
C     THIS SUBROUTINE DISPLAYS THE FINAL TABLEAU IN VARIOUS FORMATS ON
C     SCREEN, PRINTER, OR BOTH SIMULTANEOUSLY.
      WRITE(1,'(A1)') CHAR(12)
100   WRITE(1,'(5(/),4X,''DO YOU WANT THE OUTPUT IN '')')
      WRITE(1,'(/,10X,''1)  E FORMAT '')')
      WRITE(1,'(/,16X,''OR'')')
      WRITE(1,'(/,10X,''2)  F FORMAT '')')
      READ(5,'(A1)') P(1)
      CALL CHECK2(P,1,2,INVAL,FMT)
      IF (INVAL .EQ. 1) THEN
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(//,5X,''IMPROPER RESPONSE,PLEASE REENTER'')')
        GOTO 100
      ENDIF
      WRITE(1,'(A1)') CHAR(12)
110   FORMAT(A)
      T=(VT/5)+1
      IF(SELOUT .EQ.'S'.OR.SELOUT .EQ.'B')THEN
        DO 470 N=1,T
          WRITE(1,'(13X,$)')
          DO 290 J=(N*5)-4,N*5
            IF(J .GT. VT)THEN
              GOTO 290
            ENDIF
            WRITE(1,280)J
 280        FORMAT(5X,'X(',I2,')',3X,$)
 290      CONTINUE
          IF(T .EQ. 1 .OR. N .EQ. T)THEN
```

```
          WRITE(1,300)
300       FORMAT(6X,'RHS')
         ELSE
          WRITE(1,'('' '')')
        ENDIF
        WRITE(1,'(''OBJ FUNCTION'',1X,$)')
        DO 320 J=(N*5)-4,N*5
         IF(J .GT. VT)THEN
          GOTO 320
         ENDIF
         IF(FMT .EQ. 1)THEN
            WRITE(1,'(1PE12.5,1X,$)')CF(J)
          ELSE
            WRITE(1,'(F12.5,1X,$)')CF(J)
         ENDIF
320     CONTINUE
        IF(T .EQ.1 .OR. N .EQ. T)THEN
         IF(FMT .EQ. 1) THEN
          WRITE(1,'(''= '',1PE12.5,1X)')Z
         ELSE
          WRITE(1,'(''= '',F12.5,1X)')Z
         ENDIF
        ELSE
         WRITE(1,'('' '')')
        ENDIF
        WRITE(1,'(''CN  NAME  VAR'',2X,65(''*''))')
        DO 400 L=1,K
         WRITE(1,'(I2,7X,$)')L
         WRITE(1,'(1X,I2,1X,$)')CB(L)
         DO 370 J=(N*5)-4,N*5
          IF(J .GT. VT)THEN
           GOTO 370
          ENDIF
          IF(FMT .EQ. 1)THEN
             WRITE(1,'(1PE12.5,1X,$)')AF(L,J)
           ELSE
             WRITE(1,'(F12.5,1X,$)')AF(L,J)
          ENDIF
370      CONTINUE
         IF(T .EQ. 1 .OR. N .EQ. T)THEN
           P(2)='='
           IF(FMT .EQ. 1)THEN
              WRITE(1,'(A1,1X,1PE12.5)')P(2),BF(L)
             ELSE
              WRITE(1,'(A1,1X,F12.5)')P(2),BF(L)
           ENDIF
          ELSE
           WRITE(1,'('' '')')
         ENDIF
400     CONTINUE
        PAUSE
        WRITE(1,110)CHAR(12)
```

```
  470   CONTINUE
        WRITE(1,110)CHAR(12)
        DO 580 I=1,K
          IF(FMT .EQ. 1)THEN
              WRITE(1,'(10X,''X('',I2,'') = '',1PE12.5)')CB(I),BF(I)
            ELSE
              WRITE(1,'(10X,''X('',I2,'') = '',F12.5)')CB(I),BF(I)
          ENDIF
  580   CONTINUE
        IF(FMT .EQ. 1)THEN
            WRITE(1,'(/14X,''Z = '',1PE12.5)')Z
          ELSE
            WRITE(1,'(/14X,''Z = '',F12.5)')Z
        ENDIF
      ENDIF
      IF(SELOUT .EQ.'F'.OR.SELOUT .EQ.'B')THEN
        DO 1470 N=1,T
          WRITE(6,'(13X,$)')
          DO 1290 J=(N*5)-4,N*5
            IF(J .GT. VT)THEN
              GOTO 1290
            ENDIF
            WRITE(6,1280)J
1280        FORMAT(5X,'X(',I2,')',3X,$)
1290      CONTINUE
          IF(T .EQ. 1 .OR. N .EQ. T)THEN
              WRITE(6,1300)
1300          FORMAT(6X,'RHS')
            ELSE
              WRITE(6,'('' '')')
          ENDIF
          WRITE(6,'(''OBJ FUNCTION'',1X,$)')
          DO 1320 J=(N*5)-4,N*5
            IF(J .GT. VT)THEN
              GOTO 1320
            ENDIF
            IF(FMT .EQ. 1)THEN
                WRITE(6,'(1PE12.5,1X,$)')CF(J)
              ELSE
                WRITE(6,'(F12.5,1X,$)')CF(J)
            ENDIF
1320      CONTINUE
          IF(T .EQ.1 .OR. N .EQ. T)THEN
            IF(FMT .EQ. 1) THEN
              WRITE(6,'(''= '',1PE12.5,1X)')Z
            ELSE
              WRITE(6,'(''= '',F12.5,1X)')Z
            ENDIF
          ELSE
            WRITE(6,'('' '')')
          ENDIF
          WRITE(6,'(''CN  NAME  VAR'',2X,65(''*''))')
```

```
          DO 1400 L=1,K
            WRITE(6,'(I2,7X,$)')L
            WRITE(6,'(1X,I2.1X,$)')CB(L)
            DO 1370 J=(N*5)-4,N*5
              IF(J .GT. VT)THEN
                GOTO 1370
              ENDIF
              IF(FMT .EQ. 1)THEN
                  WRITE(6,'(1PE12.5,1X,$)')AF(L,J)
                ELSE
                  WRITE(6,'(F12.5,1X,$)')AF(L,J)
              ENDIF
1370        CONTINUE
            IF(T .EQ. 1 .OR. N .EQ. T)THEN
                P(2)='='
                IF(FMT .EQ. 1)THEN
                    WRITE(6,'(A1,1X,1PE12.5)')P(2),BF(L)
                  ELSE
                    WRITE(6,'(A1,1X,F12.5)')P(2),BF(L)
                ENDIF
              ELSE
                WRITE(6,'('' '')')
            ENDIF
1400      CONTINUE
          WRITE(6,'(////)')
1470    CONTINUE
        DO 1580 I=1,K
          IF(FMT .EQ. 1)THEN
              WRITE(6,'(10X,''X('',I2,'') = '',1PE12.5)')CB(I),BF(I)
            ELSE
              WRITE(6,'(10X,''X('',I2,'') = '',F12.5)')CB(I),BF(I)
          ENDIF
1580    CONTINUE
        IF(FMT .EQ. 1)THEN
            WRITE(6,'(/14X,''Z = '',1PE12.5)')Z
          ELSE
            WRITE(6,'(/14X,''Z = '',F12.5)')Z
        ENDIF
        WRITE(6,'(A1)')CHAR(12)
      ENDIF
      PAUSE
      RETURN
      END
```

```
C ****************************************
C                                        *
C  MODULE 4 UNIT47                       *
C    UNIT #USES: NONE                    *
C                                        *
C  SUBROUTINE CHECK2                     *
C  USE: THIS SUBROUTINE ACCEPTS KEYBOARD NUMERIC INPUTS AS   *
C       "CHARACTERS" AND RETURNS THE INTEGER EQUIVALENT IF THE *
C       INPUT IS OF THE CORRECT TYPE.    *
C                                        *
C  CALLED BY: SUBROUTINE MULCNG          *
C             SUBROUTINE ADDCON          *
C  CALLS     : NONE                      *
C                                        *
C                                        *
C                                        *
C  VARIABLES:                            *
C  USES      : E,D,HVAL                  *
C  MODIFIES : INVAL,INEW                 *
C                                        *
C ****************************************
      SUBROUTINE CHECK2(E,D,HVAL,INVAL,INEW)
      CHARACTER ALLOW(11)*1,E(10)*1
      INTEGER D,HVAL
      DATA ALLOW/'1','2','3','4','5','6','7','8','9','0',' '/
      INEW=0
      INVAL=0
      DO 4710 I=1,D
C       EACH CHARACTER IS CHECKED FOR VALIDITY.
        DO 4700 J=1,10
          IF(E(I) .EQ. ALLOW(11) )THEN
C           SPACES ARE IGNORED.
              GOTO 4710
            ELSEIF(E(I) .EQ.ALLOW(J))THEN
C             IF THE CHARACTER IS A NUMBER, THE VALUE OF THE NUMBER
C             BEING ASSEMBLED IS MULTIPLIED BY 10 AND THE VALUE OF THE
C             NEW CHARACTER IS ADDED.
              INEW=INEW*10+ICHAR(E(I))-48
              GOTO 4710
            ELSEIF(J .EQ. 10)THEN
C             IF THE CHARACTER DOES NOT FIT ONE OF THE ABOVE
C             DESCRIPTIONS, AN INVALID FLAG IS SET, AND THE PROGRAM
C             RETURNS TO GET A NEW INPUT.
              INVAL=1
              INEW=0
              RETURN
          ENDIF
4700    CONTINUE
4710  CONTINUE
      IF(INEW .EQ. 0 .OR. INEW .GT. HVAL)THEN
        INVAL=1
        INEW=0
```

```
  RETURN
ENDIF
RETURN
END
```

```
C ****************************************
C                                        *
C  MODULE 4 UNIT47                       *
C    UNIT #USES: NONE                    *
C                                        *
C  SUBROUTINE CHECK                      *
C  USE: THIS SUBROUTINE ACCEPTS KEYBOARD NUMBERIC INPUTS AS   *
C       "CHARACTERS" AND RETURNS THE REAL VARIABLE EQUIVALENT *
C       IF THE INPUT IS OF THE CORRECT TYPE.                  *
C                                        *
C  CALLED BY: SUBROUTINE MULCNG          *
C             SUBROUTINE ADDCON          *
C  CALLS    : NONE                       *
C                                        *
C  VARIABLES:                            *
C  USES     : E                          *
C  MODIFIES : INVAL,RNEW                 *
C                                        *
C ****************************************
      SUBROUTINE CHECK(E,INVAL,RNEW)
      CHARACTER ALLOW(14)*1,E(10)*1
      REAL M
      INTEGER DECIMA
      DATA ALLOW/'1','2','3','4','5','6','7','8','9','0','+','-','.',
     .' '/
      RNEW=0.0
      M=.1
      INVAL=0
      DECIMA=0
      NEGAT=0
      DO 4740 I=1,10
        IF(E(I) .EQ. ALLOW(14))THEN
          GOTO 4740
        ENDIF
        DO 4730 J=1,13
          IF(E(I) .EQ. ALLOW(J))THEN
C           IF VALID SPECIAL CHARACTERS ARE PRESENT, FLAGS ARE SET
C           ACCORDINGLY.  IF THE CHARACTER IS A NUMBER, THE NUMBER
C           WHICH IS BEING ASSEMBLED IS MULTIPLIED BY 10, AND THE VALUE
C           OF THE NEW CHARACTER IS ADDED.
              IF(DECIMA .EQ. 1)THEN
                GOTO 4710
              ELSEIF(E(I) .EQ. '-')THEN
                NEGAT=1
                GOTO 4740
              ELSEIF(E(I) .EQ. '.')THEN
                DECIMA=1
                GOTO 4740
              ELSE
                RNEW=RNEW*10+ICHAR(E(I))-48
                GOTO 4720
             ENDIF
```

```
4710        RNEW=RNEW+(ICHAR(E(I))-48)*M
            M=M*.1
            GOTO 4720
          ELSEIF(J .EQ.13)THEN
            INVAL=1
            RNEW=0.0
            RETURN
        ENDIF
        GOTO 4730
4720    J=13
4730   CONTINUE
4740  CONTINUE
C     IF THE NUMBER IS NEGATIVE, THE VALUE OF THE ASSEMBLED REAL NUMBER
C     IS SWITCHED.
      IF(NEGAT .EQ. 1)THEN
        RNEW=-RNEW
      ENDIF
      RETURN
      END
```

```
C ****************************************
C                                        *
C  MODULE 4 UNIT48                       *
C    UNIT $USES: NONE                    *
C                                        *
C  SUBROUTINE RETRIV                     *
C  USE: THIS SUBROUTINE READS ALL REQUIRED SENSITIVITY ANALYSIS DATA  *
C       AND PARAMETERS FROM A DISK FILE.  IT REARRANGES THE DATA AND  *
C       COMPUTES OTHER PARAMETERS TO ASSIST IN THE ANALYSIS.          *
C                                        *
C  CALLED BY: PROGRAM MAINSA             *
C  CALLS    : NONE                       *
C                                        *
C  VARIABLES:                            *
C  USES     : V,K,VT,INEQ,NEG,BO,IFLAG(10),IFLAG(5)       *
C  MODIFIES : INEQ,NEG,AO,AF,BO,BF,CF,BM                  *
C                                        *
C ****************************************
      SUBROUTINE RETRIV
      INTEGER I,V,K,J,COL,ROW,IFLAG,INEQ,VT,CB,CONSTR,NEG
     .,SLACK,ARTVAR
      REAL AO,AF,BO,BF,CO,CF,Z,TEMPA(20,20),TEMPC(20),BM,
     .TEMP(20)
      CHARACTER PN*20,FN*10,SELINP(10)*1,SELOUT
      COMMON/ONE/SELOUT,FN
      COMMON/TWO/VT,INDEXG,INDEXL,INDEXE,NGC,NLC,NEC,NEG(20),MXMN,BM
$INCLUDE CONVAR
      WRITE(1,'(A1)') CHAR(12)
      WRITE(1,'(5(/),5X,''ENSURE DISK LP2: IS AVAILABLE ''.5(/))')
      PAUSE
C     THE NAME OF THE FILE WITH THE DATA IS READ.
      OPEN(7,FILE='LP2:LPDATAN',STATUS='OLD',FORM='UNFORMATTED')
      READ(7) FN
      WRITE(1,'(A1)') CHAR(12)
C     THE USER IS ALLOWED TO CHANGE THE DATA FILE NAME IF DESIRED.
4800  WRITE(1,'(5(/),8X,''THE CURRENT DATA FILE IS ''//,14X,A10,///)')FN
      WRITE(1,'(5X,''DO YOU WISH TO USE THIS TABLEAU'',5(/))')
      READ(5,'(A1)') SELINP(1)
      IF (SELINP(1) .EQ. 'N') THEN
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(5(/),2X,''PLEASE ENTER THE NEW VOLUME:FILENAME''/)')
        WRITE(1,'(13X,''EG. LP2:TEST1''//)')
        READ(1,'(A10)') FN
        REWIND 7
        WRITE(7)FN
      ELSEIF (SELINP(1) .NE. 'Y') THEN
        WRITE(1,'(A1)') CHAR(12)
        WRITE(1,'(5X,''INVALID RESPONSE, TYPE ''''Y'''' OR ''''N'''' ''
     . ,///)')
        GOTO 4800
      ENDIF
      CLOSE(7,STATUS='KEEP')
```

```
      WRITE(1,'(A1)') CHAR(12)
      WRITE(1,'(5(/),7X,''ENSURE THE DISK CONTAINING'')')
      WRITE(1,'(/,12X,A10)') FN
      WRITE(1,'(//,12X,''IS AVAILABLE''///)')
      PAUSE
C     THE DATA FILE IS READ.
      OPEN(3,FILE=FN,STATUS='OLD',FORM='UNFORMATTED')
      READ(3)PN,MXMN,K,V,IFLAG(5)
      DO 4810 I=1,K
        READ(3) INEQ(I),BO(I)
        DO 4820 J=1,V
          READ(3) AO(I,J)
4820    CONTINUE
4810  CONTINUE
      DO 4830 J=1,V
        READ(3) CO(J)
4830  CONTINUE
      READ(3)IFLAG(10),VT
      DO 4840 I=1,K
        READ(3) BF(I),CB(I)
        DO 4850 J=1,VT
          READ(3) AF(I,J)
4850    CONTINUE
4840  CONTINUE
      DO 4860 J=1,VT
        READ(3) CF(J)
4860  CONTINUE
      READ(3) Z
      CLOSE(3,STATUS='KEEP')
      WRITE(1,'(A1)')CHAR(12)
      WRITE(1,'(5(/),8X,''ENSURE LP2: IS AVAILABLE''/////)')
      PAUSE
      NGC=0
      NLC=0
      NEC=0
      DO 4861 CONSTR=1,K
        NEG(CONSTR)=0
4861  CONTINUE
C     EACH CONSTRAINT IS ALTERED IF THE PROBLEM WAS A MINIMIZATION AND
C     THE CONSTRAINT WAS A "GREATER THAN" OR IF THE ORIGINAL CONSTRAINT
C     HAD A NEGATIVE RIGHT-HANAD SIDE (NOT MINIMIZATION).
      DO 4870 CONSTR=1,K
        IF(IFLAG(10) .EQ.1)THEN
             IF(INEQ(CONSTR) .EQ.1)THEN
               INEQ(CONSTR) = 0
               NEG(CONSTR)  = 1
               BO(CONSTR)=-BO(CONSTR)
               DO 4805 COL=1,V
                 AO(CONSTR,COL)=-AO(CONSTR,COL)
4805           CONTINUE
             ENDIF
           ELSE
```

```
      IF (BO(CONSTR) .LT. -0.00001 ) THEN
        NEG(CONSTR)=1
        BO(CONSTR) = - BO(CONSTR)
        DO 4200 J=1,V
          AO(CONSTR,J) = -AO(CONSTR,J)
4200    CONTINUE
        IF (INEQ(CONSTR) .EQ. 0) THEN
            INEQ(CONSTR) = 1
          ELSEIF (INEQ(CONSTR) .EQ.1 ) THEN
            INEQ(CONSTR) = 0
        ENDIF
      ENDIF
    ENDIF
C   DETERMINE INDICES
    IF (INEQ(CONSTR) .EQ. 0) THEN
        NLC = NLC +1
      ELSEIF (INEQ(CONSTR) .EQ.1 ) THEN
        NGC = NGC +1
    ENDIF
4870 CONTINUE
    INDEXG = V + 1
    INDEXL = INDEXG + NGC
    INDEXE = INDEXL + NLC
    NEC=K-NGC-NLC
    SLACK  = 0
    ARTVAR = 0
    DO 111 CONSTR=1,K
      TEMP(CONSTR)=0
111  CONTINUE
    DO 4899 CONSTR=1,K
C     FIND THE COLUMN ASSOCIATED WITH THE CONSTRAINT
      IF(INEQ(CONSTR) .EQ.0)THEN
          COL=INDEXL + SLACK
          SLACK = SLACK + 1
        ELSE
          COL=INDEXE + ARTVAR
          ARTVAR = ARTVAR + 1
      ENDIF
C
C     REARRANGE THE LAST K COLUMNS TO PUT THEM IN THE B INVERSE ORDER
C     TEMPORARILY HOLD THE VALUES OF C(J) AND A(I,J) IN TEMP UNTIL
C     THEY ARE SORTED OUT
C
      DO 4890 ROW = 1,K
        TEMPA(ROW,CONSTR)=AF(ROW,COL)
        TEMPC(CONSTR)=CF(COL)
        IF(CB(ROW).EQ.COL)THEN
          TEMP(ROW)=CONSTR+INDEXL-1
        ENDIF
4890  CONTINUE
4899 CONTINUE
C
```

```
C     PUT THE VALUES WHICH WERE HELD IN TEMP BACK INTO THE CORRECTED
C     C(J) AND A(I,J) COLUMNS
C
      DO 4898 COL=1,K
        CF(COL+INDEXL-1)=TEMPC(COL)
        DO 4897 CONSTR=1,K
          AF(CONSTR,COL+INDEXL-1)=TEMPA(CONSTR,COL)
          IF(TEMP(CONSTR).GT.0)THEN
            CB(CONSTR)=TEMP(CONSTR)
          ENDIF
4897    CONTINUE
4898  CONTINUE
      BM=0.0
      DO 4866 J=1,V
        IF(ABS(CO(J)) .GT. BM)THEN
          BM=ABS(CO(J))
        ENDIF
4866  CONTINUE
      BM=ANINT(BM)*10
      IF(BM .LT. 10)THEN
        BM=10.0
      ENDIF
      DO 4877 CONSTR=1,K
      IF(INEQ(CONSTR) .NE. 0)THEN
          CF(CONSTR+INDEXL-1)=CF(CONSTR+INDEXL-1)-BM
        ENDIF
4877  CONTINUE
      RETURN
      END
```

```
      COMMON/VAR1/AD(20,20),AF(20,60),BD(20),BF(20),CD(20),CF(60),Z
     .,K,V,IFLAG(10),INEQ(20),CB(20)
```

VITA

Theodore R. E. Fraley was born in Greenville, Ohio on 13 April , 1943. After graduation from high school in Atascadero, California, he atttended California State Polytechnic College before joining the Air Force in 1964. He received a degree in Aeronautical Engineering from Oklahoma State University in 1968. Following pilot training, he has been involved in tactical flight operations. He entered the Air Force Institute of Technology in June, 1981.

Permanent address: 6843 Santa Lucia
Atascadero, California
93422

Dale A. Kem was born in Huntington, Indiana on 25 February, 1953. He graduated from high school in Huntington, Indiana in 1971 and attended Purdue University from which he received a Bachelor of Chemical Engineering degree in 1975. Following graduation, he received a commission in the U.S. Army. He was assigned to Fort Sill, Oklahoma until 1977. He attended the Defense Language Institute of Monterey, California for one year of language training and subsequently was assigned to TUSLOG, Group 67, Turkey. Upon his return from overseas, he attended the Ordnance Advance Course at Redstone Arsenal, Alabama and in June, 1981 entered the Air Force Institute of Technology.

Permanent address: 9065 S. 900 W 35
Lafontaine, Indiana
46940

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GOR/OS/82D-4	2. GOVT ACCESSION NO.: AD-A124 804
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): FORTRAN BASED LINEAR PROGRAMMING FOR MICROCOMPUTERS	5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Theodore R.E. Fraley, Major USAF; Dale A. Kem, Captain USA	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology(AFIT/EN) Wright Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: December 1982
	13. NUMBER OF PAGES: 429
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17.
LYNN E. [illegible]
Dean for Research and Professional Development
Air Force [illegible]
Wright-Patterson [illegible]

19 JAN 1983

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

COMPUTER PROGRAM, LINEAR PROGRAMMING, MULTIPLE OPTIMAL, DEGENERATE, SENSITIVITY ANALYSIS, INFEASIBLE, MICROCOMPUTER, EDUCATIONAL, UNBOUNDED, OPTIMALITY, SIMPLEX

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Linear programming is an analytical technique used in decision analysis. This paper describes the development and use of a highly interactive, non-programmer oriented, linear programming software package implemented on a microcomputer. This software, written in FORTRAN and supported by the UCSD Pascal Operating System, has allowed increased portability while providing the capability of solving moderate-sized LP models. Also available are extensive postoptimal sensitivity analysis capabilities.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

The modularly implemented package provides interactive, instructional sessions with user input LP models. The user is guided through tableau formulation and pivot element selection to an optimal solution by a series of option displays and user selections. This module also provides instructors the ability to rapidly demonstrate the application of the simplex algorithm.

A separate module provides a more rapid problem solution with minimal interaction. Options allow either primal or dual problem solution with screen-oriented output to either a monitor or printer. The sensitivity analysis capabilities include right-hand-side, cost coefficient, and constraint ranging. Also provided is the ability to add constraints and variables to the original model.

FILME

8