

AD 742254

PREPARED FOR THE
DEPARTMENT OF THE ARMY
Contract DAHC19-69-C-0017

DISTRIBUTION STATEMENT
Approved for public release;
distribution unlimited.

RAC-TP-445
FINAL DRAFT
OF VOL II

JANUARY 1972

A Methodology for Optimal Planning over Time

Volume II Appendices A, B, C, D, and E in Support of Volume I

by Charles A. Allen
Beverly D. Causey
James E. Falk
Ronald G. Magee
Charles W. Mylander
Ronald New, *Project Director*
John D. Pearson
Philip D. Robers

FINAL DRAFT

U D C
RECEIVED
JAN 23 1972
LIBRARY

Copy 21 of 165

VOL #1 = AD742251

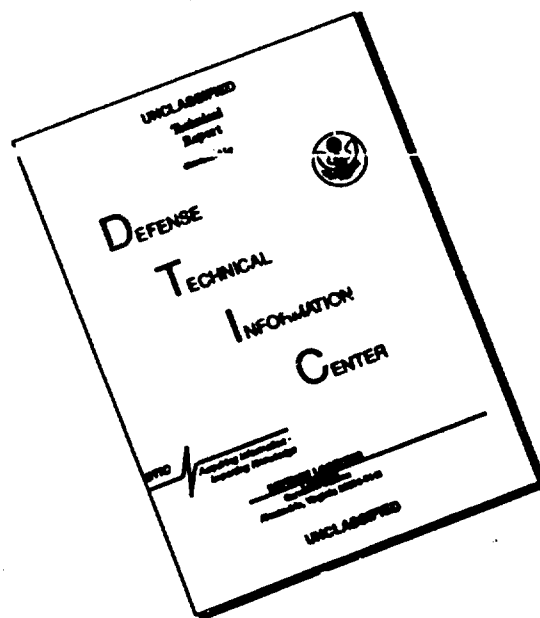
Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151



Research Analysis Corporation

283

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

ACCESSION FOR		
OFSTI	WHITE SECTION	<input checked="" type="checkbox"/>
DDC	BUFF SECTION	<input type="checkbox"/>
UNANNOUNCED		<input type="checkbox"/>
JUSTIFICATION		
BY		
DISTRIBUTION/AVAILABILITY CODES		
DIST.	AVAIL. AND	SPECIAL
A		

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D		
<i>(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)</i>		
1. ORIGINATING ACTIVITY (Corporate author) RESEARCH ANALYSIS CORPORATION		2a. REPORT SECURITY CLASSIFICATION none
		2b. GROUP N/A
3. REPORT TITLE A Methodology for Optimal Planning Over Time, Volume II, Appendices A,B,C,D,&E		
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Technical Paper		
5. AUTHOR(S) (First name, middle initial, last name) Charles A. Allen Ronald G. Magee Philip D. Robers Beverly D. Causey Ronald New, Project Director Charles W. Mylander James E. Falk John D. Pearson		
6. REPORT DATE January 1972	7a. TOTAL NO. OF PAGES 284	7b. NO. OF REFS 2
8a. CONTRACT OR GRANT NO. A. PROJECT NO. 011.310 c. d.		9a. ORIGINATOR'S REPORT NUMBER(S) TP-445, Final Draft of Vol II 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
10. DISTRIBUTION STATEMENT Approved for public release; distribution unlimited.		
11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY US Army, Combat Developments Command Combat Systems Group	
13. ABSTRACT This report describes a methodology which can be used to identify the most cost-effective plan for the phase-in and phase-out of vehicle systems -- a methodology for optimal fleet planning over time. Volume I provides a systematic development of the problem structure, a qualitative description of the solution procedure, and mathematical and operational descriptions of the algorithm. Volume II provides appendices containing a demonstration problem, subroutine descriptions, program flow charts, program listings, and error message descriptions.		

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
branch and bound						
fleet planning						
non-convex programming						
non-linear programming						
optimization over time						
vehicle systems						

UNCLASSIFIED

Security Classification

**A Methodology
for Optimal Planning over Time
Volume II
Appendices A, B, C, D, and E
in Support of Volume I**

by

Charles A. Allen
Beverly D. Causey
James E. Falk
Ronald G. Magee
Charles W. Mylander
Ronald New, *Project Director*
John D. Pearson
Philip D. Robers

FINAL DRAFT

DISTRIBUTION STATEMENT
Approved for public release;
distribution unlimited.

Research Analysis Corporation

McLean, Virginia 22101



Area Code 703
893-5900



DARD-ARS

DEPARTMENT OF THE ARMY
OFFICE OF THE CHIEF OF RESEARCH AND DEVELOPMENT
WASHINGTON, D.C. 20310

1. Volume I, "A Methodology for Optimal Planning Over Time" and Volume II, "Appendices A, B, C, D and E in Support of a Methodology for Optimal Planning Over Time - Volume I," were prepared by the Research Analysis Corporation for the Combat Systems Group, United States Army Combat Developments Command, and document RAC study 011.310, "Aircraft Systems Least Cost Phase-In." Copies of these reports are forwarded for your retention and use.
2. The methodology described in these volumes was developed to meet in part the need of the US Army to determine an optimal plan for phasing in new aircraft systems to meet its worldwide commitments yet remain within budgetary constraints. It provides to planners a tool for use in planning situations involving consideration of large numbers of alternative systems and combinations of tasks.
3. The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

FOR THE CHIEF OF RESEARCH AND DEVELOPMENT:

A handwritten signature in cursive script, appearing to read "Stanley R. Meeken".

STANLEY R. MEEKEN

Colonel, GS

Chief, Studies and Analyses Division

Published January 1972
by
RESEARCH ANALYSIS CORPORATION
McLean, Virginia 22101

CONTENTS

VOLUME I

SUMMARY	S-1
INTRODUCTION	I-1
CHAPTER 1 - PROBLEM STRUCTURE	1-1
CHAPTER 2 - A QUALITATIVE DESCRIPTION OF BRANCH AND BOUND	2-1
CHAPTER 3 - MATHEMATICAL DESCRIPTION OF PROBLEM AND ITS SOLUTION	3-1
THE CONSTRAINT SET	3-1
Material Balance Constraints - Consistency Constraints- Vintage Constraints - Master Variable Relation Constraints - Cost Constraints	
THE OBJECTIVE FUNCTION	3-6
Primary Cost Categories - Secondary Cost Categories	
THE SOLUTION PROCEDURE	3-10
Linear Envelopes - Bounding the Solution - Partitioning into Subsets - The Algorithm	
CHAPTER 4 - AN OPERATIONAL DESCRIPTION OF THE BRANCH AND BOUND SOLUTION PROCEDURE	4-1
PROGRAM INTERFACING	4-1
THE MATRIX GENERATOR	4-5
Program Logic - Core Allocation - User's Subroutine (YRCOST) - Input Formats - Output Description	
THE MAIN PROGRAM	4-21
Program Logic - Core Allocation - User's Subroutine (GMTPHE) - Input Formats - Output Description	
THE REPORT GENERATOR	4-34
Program Logic - Core Allocation - User's Subroutine (YRCOST) - Input Formats - Output Description	
REFERENCES	R-1
FIGURES: VOLUME I	
1-1 A Typical List of Alternative Vehicle Arrays Which Could Service a Mission Group	1-1
1-2 Eight Independent Groups Each with Twenty Equally-Effective Alternatives	1-3
1-3 The Mission Group Problem	1-5

FIGURES: VOLUME I (continued)

2-1	A Concave Cost Function	2-1
2-2	Partitioning the Total Solution Set Into Subsets	2-3
3-1	Cost Function	3-11
3-2	A Linear Envelope of a Concave Function with Discontinuity at Origin	3-13
3-3	The Algorithm	3-17
4-1a	Control Cards to Execute All Programs as a Single Job	4-3
4-1b	Control Cards to Execute Programs as Separate Jobs	4-4
4-2	System Macro Flowchart	4-6
4-3	Symbolic Naming Convention	4-7
4-4	Positive Integer Code	4-8
4-5	Basic Cards	4-13
4-6	Vehicle Table	4-14
4-7	Period Table	4-15
4-8	Task Table	4-16
4-9	Deck Structure for Matrix Generator	4-20
4-10	Input Data Formats	4-27
4-11	Deck Structure for Main Program	4-30
4-12	Deck Structure for Report Generator	4-37

CONTENTS
VOLUME II

APPENDIX A - SAMPLE PROBLEM	A-1
APPENDIX B - SUBROUTINE DESCRIPTIONS	B-1
APPENDIX C - FLOWCHARTS	C-1
APPENDIX D - PROGRAM LISTING	D-1
APPENDIX E - ERROR MESSAGES	E-1
GLOSSARY	G-1
REFERENCES	R-1

FIGURES: VOLUME II

A-1	Alternative Fleet Mixes for 1972	A-1
A-2	Capital Equipment (Truck) Requirements as a Function of Time	A-2
A-3	The Simplified Objective (cost) Function for the Sample Truck Problem	A-4
A-4	The Sample Problem Data Deck for the GENLCP Program	A-8
A-5	GENLCP Output For Sample Problem, Parts (a) - (f)	A-10
A-6	The Sample Problem Data Deck For the BECAV2 Program	A-17
A-7	A Branching Tree For the Sample Problem	A-19
A-8	The Sample Problem Data Deck For the REPGEN Program	A-20
A-9	REPGEN Output For Sample Problem, Parts (a) and (b)	A-21
A-10	A Sample Optimal Plan For the Ace Trucking Company	A-24

APPENDIX A

A SAMPLE PROBLEM

We have chosen a relatively small sample problem to illustrate an implementation of the branch-and-bound algorithm and its corresponding program. We will structure the problem from a user's viewpoint, formulate the objective function and the constraint set, illustrate the preparation of the data decks and user's subroutines, and explain the logic of the results. The problem to be described here has been discussed in a previous document.

The Ace Trucking Co., in planning for next year's workload, estimates that the company can serve its customers with a fleet of 67 light trucks and 16 cross-country trailers. An alternative fleet was also considered consisting of 43 light and 20 medium sized trucks, together with only 7 cross-country trailers. Finally, the only other practical alternative considered was a mix of 42 medium trucks and 12 of the big trailers. The medium trucks, however, are of a new design and will not be available for next year unless the company is willing to pay a substantial premium. At first it appeared that choosing one of these three alternative fleets (shown in table form below) was

	light trucks	medium trucks	trailers
Alternative #1	67	—	16
Alternative #2	43	20	7
Alternative #3	—	42	12

Figure A-1 ALTERNATIVE FLEET MIXES FOR 1972

the only decision issue. However, it soon became clear to the planning group at Ace Trucking Co. that the investment decision should also depend on the utilization of the trucks in subsequent years, in addition to that utilization planned for the next year. And furthermore, the existing fleet of trucks was far from obsolete, even though maintenance costs on some of the older vehicles were beginning to climb. Realizing these factors, the planning group estimated the workload for their trucks over the next three years (beyond which they could not be confident of their estimates), and then prepared a requirements table like that in Figure A-2 below.

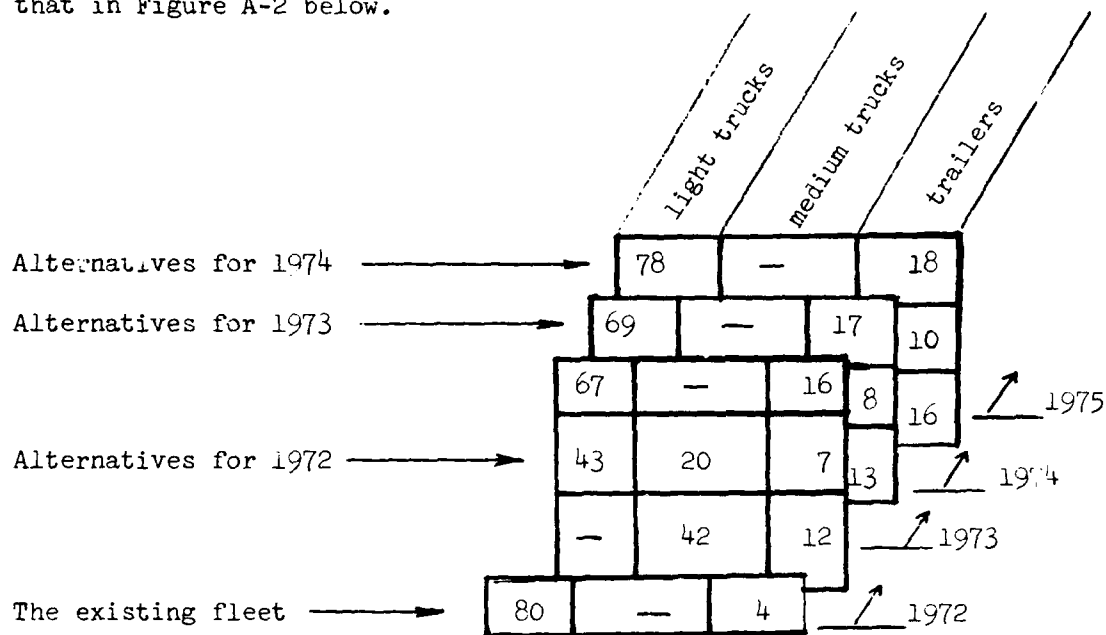


Figure A-2 CAPITAL EQUIPMENT (TRUCK) REQUIREMENTS AS A FUNCTION OF TIME

The existing fleet did not include any medium sized trucks; 60 light trucks were two years old, but the remaining 20 were purchased only last year. The inherited trailers were also two years old. Finally, based on the projected cash flow position of Ace for the next three years, it was decided that a limit (cost constraint) be placed on new truck procurements for each of the three years.* Decision time for Ace

* Costs constraints were not imposed on this problem in the previous referenced description; hence, a slightly different solution was obtained.

Trucking Co. is January 1972, and the question is: what is the optimal plan over the next three-year period?

Initially, one should ask: how many alternative plans exist? If we permute three alternatives per year with three years, we develop 27 alternative plans — but this number ignores all permutations of the existing fleet as well as all concern for the "welfare" of the inherited fleet from year to year. In addition, there are questions like "buy" or "lease," ... "sell," "salvage," or "store," etc., making the number of alternative plans very large indeed (literally thousands even in this trivial problem). In fact, for real-world problems, it would not be atypical to have millions of possible alternative plans confronting the decision-maker, each involving a maze of cost factors — making evaluation of each and every plan rather impractical. Problems like this can be solved efficiently, without evaluating each and every alternative, using the mathematical programming techniques embodied within the Palk-U-Tani Algorithm.²

The objective function for this problem is developed from the general form of Fig. 2-1 (page 2-11) plus projected estimates of the cost coefficients for each cost category. We have only three vehicles under consideration but we artificially introduce a fourth vehicle (and call it MEDIUM*) to account for the premium charge if we buy MEDIUM trucks for the first year (1972). That is, the program will treat the purchase of MEDIUM trucks and MEDIUM* trucks separately, as if they were distinct. In Figure A-3 we show the reduced form of the objective function. Note that,

- (1) There is only one (possible) R&D charge and that is associated with the MEDIUM* vehicles (called vehicle No. 1; the LIGHT-MEDIUM and TRAILERS are numbered 2, 3, and 4, respectively).
- (2) We use r_k increase in operating cost over time (for simplicity, hence, we drop the second subscript in the c coefficients as well as the summation over k).
- (3) We have introduced the specific values for the number of vehicles ($N=4$) and the length of the planning period ($T=3$) where they appear in Figure A-3.

$$\begin{aligned}
\varphi(x) = & U_1(x_1) + \sum_{j=1}^4 a_j x_j + \sum_{j=1}^4 \sum_{l=1}^3 (-d_{jl}) s_{jl} \\
\text{R\&D Costs} & \quad \text{Procurement Costs} & \quad \text{Savings due to mothballing of unneeded vehicles} \\
+ & \sum_{j=1}^4 \sum_{l=k}^0 c_j w_{jln} + \sum_{j=1}^4 \sum_{l=1}^3 \sum_{n=1}^3 c_j x_{jln} \\
& \quad \text{(inherited fleet)} & \quad \text{Operating and maintenance cost} \\
& & \quad \text{(purchased fleet)} \\
+ & \sum_{j=1}^4 \sum_{l=k}^0 \sum_{n=0}^2 (-e_{j,(n-l+1)}) w_{jln} + \sum_{j=1}^4 \sum_{l=1}^3 \sum_{n=1}^3 (-e_{j,(n-l+1)}) x_{jln} \\
& \quad \text{(inherited fleet)} & \quad \text{Savings resulting from vehicle salvage} \\
& & \quad \text{(purchased fleet)} \\
+ & \sum_{j=1}^4 \sum_{l=k}^0 (-f_{j,(4-l)}) w_{jls} + \sum_{j=1}^4 \sum_{l=1}^3 (-f_{j,(4-l)}) x_{jls} \\
& \quad \text{(inherited fleet)} & \quad \text{Savings due to crediting the value of fleet owned} \\
& & \quad \text{at the end of the planning period} \\
& & \quad \text{(purchased fleet)}
\end{aligned}$$

Fig A3 - The Simplified Objective (cost) Function For the Sample Truck Problem

The constraint set is developed from the general descriptions given in Chapter 3 (starting on p. 3-1).

The material balance constraints become:

$$\sum_{j=K_j}^3 w_{j\ell'm} + \sum_{\ell'=i}^{\ell} \sum_{m=\ell}^3 x_{j\ell'm} = \sum_{i \in M_\ell} \sum_{k=1}^{R_{i\ell}} u_{ijk\ell} P_{ik\ell} + s_{j\ell}$$

j = 1, 2, 3, 4
 \ell = 1, 2, 3

Note that we have assumed no attrition ($v_{\ell-\ell'} = 1$), and have assigned the $t_{i\ell}$ factor = 1 for all mission groups.

The consistency constraints reduce to:

$$\sum_{k=1}^{R_{i\ell}} P_{ik\ell} = 1; \ell = 1, 2, 3 \text{ and } i = 1, 2, 3$$

The vintage constraints for the light trucks and trailers are:

$$w_{j\ell} = \sum_{m=0}^{Q+\ell} w_{j\ell m}; j = 2, 4 \text{ and } \ell = -1, 0$$

where L_j has been replaced by 10 subperiods (years) for all vehicles and K_j by -1 for both inherited vehicles.

The master variable constraints are simply:

$$x_j = \sum_{\ell=1}^3 \sum_{m=\ell}^3 x_{j\ell m}; j = 1, 2, 3, 4$$

Finally, since we have chosen to introduce cost constraints, we have

$$H_{\ell} = \sum_{j=1}^4 a_j^0 \sum_{m=\ell}^3 x_{j\ell m} + P_{\ell} - P_{\ell-1}$$

We (i.e., Ace Trucking Co.) will set the cost constraint, $H_{\ell} = \$150,000.$, $\$250,000.$, and $\$300,000.$, for the three years of the planning period, respectively. The linear approximation a_j^0 , to the procurement cost function will be selected after inspection of the actual cost vs. quantity curve.

A careful inspection of the total constraint set for this problem will indicate a total of 24 constraints (recall that there is no material balance constraint for $[\ell=1, j=3]$ and no vintage constraint for $[j=4, \ell=0]$). Similarly, a count of the number of variables (being careful to delete those which must equal zero because of specific exclusions in this sample problem) will indicate a total of 60. The reader will note that the GENLCP Program automatically computes and prints these totals for use in the BBCAV 2 Program.

We are now ready to prepare the data decks and user subroutines. The user subroutine GETPHI is shown in the program listing on page D-32. It is here that we describe the R&D and procurement equations for the four vehicles in the sample problem. Note that vehicle No. 1 (MEDIUM*) has an R&D (premium) charge of $\$300,000.$ * — the other three vehicles have no R&D charge and their procurement costs are simply described by concave functions of the form ax^b . Of course, other forms of concave functions could have been used.

* We have chosen to scale all costs by 10^6 . This means that all final cost data should be multiplied by 10^6 .

The only other user subroutine YRCOST (see page D-19) is prepared for use in the GENLCP program, then duplicated for use in REPGEN. As described previously on p. 4-10 of chapter 4, YRCOST is used to calculate the operating, mothballing, salvage, and truncation cost coefficients, c_{jk} , $d_{j\ell}$, e_j , $(m-\ell+1)$ and f_j , $(Y-\ell+1)$ respectively. For our sample problem, we use no increase in operating cost overtime ($R=0.$); a mothballing savings factor of $R1 = 0.9$; a salvage savings factor of $\alpha = 0.5$; and truncation savings based upon a linear decay from an estimate of the purchase cost (input through the GENLCP data deck) and an assumed 10 year lifetime for each vehicle.

The data deck for the GENLCP program can now be prepared (see figure A-4).

In entry (card image) No. 1 we give the problem title, the first and last year of the planning period, and then specify the four vehicle tables, three task tables, and five period tables — the first two of which are inherited periods. Entry 2 is the VEHICLE header card for the first vehicle. Entry 3 describes the first vehicle as LIGHT, indicates an availability date of 1970 (i.e. an inherited vehicle) and finally a ten year vehicle lifetime. Entry 4 indicates that 60 light vehicles were purchased in 1970 and 20 light vehicles were purchased in 1971. Entry 5 indicates a \$3,000. purchase cost estimate for purposes of calculation of the truncation and salvage value, a ten year operating cost of \$120,000., zero R&D cost for the light vehicle, zero attrition for the light vehicle, and finally an estimated linear purchase cost coefficient of \$3,000., respectively. This linear purchase cost coefficient estimate was based upon a study of the corresponding non-linear procurement equation for the light vehicle. In general, one should choose the cost coefficient (slope of the straight line) such that the straight line intersects the non-linear curve at or about the estimated solution value. The consequences of a poor estimate will be described shortly. Entries 6 through 15 simply complete the vehicle tables. Entries 16 through 31 describe the period tables. The first two periods (1970 and 1971) are inherited periods. In period 1972 we indicate a cost constraint of \$150,000. in entry 21. Entry 22 specifies that there exists only one task in 1972 and its scale factor is 1.0.

1	SAMPLE	1972	1974	4	3	5
2	VEHICLE					
3	LIGHT	1970	10			
4		60	20			
5		.003	.12	0.0	1.0	.003
6	VEHICLE					
7	MEDIUM*	1972	10			
8		.006	.14	.300	1.0	.006
9	VEHICLE					
10	MEDIUM	1973	10			
11		.005	.14	0.	1.0	.006
12	VEHICLE					
13	TRAILER	1970	10			
14		4				
15		.01	.16	0.	1.0	.012
16	PERIOD					
17	1970	1971				
18	PERIOD					
19	1971	1971				
20	PERIOD					
21	1972	1972	.15			
22		1	1.00			
23		1	1.0			
24	PERIOD					
25	1973	1973	.25			
26		1	1.0			
27		2	1.0			
28	PERIOD					
29	1974	1974	.3			
30		1	1.0			
31		3	1.0			
32	TASK					
33		1	2	3		
34	LIGHT	MEDIUM*	TRAILER			
35	67.0	0.0	16.0			
36	43.0	20.0	7.0			
37	0.0	42.0	12.0			
38	TASK					
39		2	4	5		
40	LIGHT	MEDIUM*	MEDIUM	TRAILER		
41	69.0	0.0	0.0	17.0		
42	45.0	22.0	0.0	2.0		
43	0.0	45.0	0.0	12.0		
44	45.0	0.0	22.0	2.0		
45	0.0	0.0	45.0	12.0		
46	TASK					
47		3	4	5		
48	LIGHT	MEDIUM*	MEDIUM	TRAILER		
49	78.0	0.0	0.0	15.0		
50	48.0	24.0	0.0	10.0		
51	0.0	50.0	0.0	16.0		
52	48.0	0.0	24.0	10.0		
53	0.0	0.0	50.0	16.0		
54	ENDTABLE					

Reproduced from
best available copy.

Fig. A-4 The Sample Problem Data Deck for the GENLCP Program

The remaining entries in the period tables should be self explanatory. The task tables are input next. Entry 33 specifies that task No. 1 will have three vehicles and three alternatives. Entries 34 through 37 input the alternative set for the first year of the planning period (compare with the figure A-2). The alternative sets for the second and third years of the planning period follow. Note that because of the introduction of the artificial MEDIUM* vehicle, the complete set of permutations yield five distinct alternatives instead of the original three. This data deck ends with an ENDTABLE card in entry 54.

We now run (process) the GENLCP program and obtain the printout of Figure A-5; parts (a) through (f). Part (a) simply prints out some input information for checking purposes, and reorders the vehicles according to the magnitude of the R&D charge; note, in this regard, that the MEDIUM* truck is "called" vehicle No. 1 (X01) since it has the R&D charge.

In the first section of Part (b), a summary of the constraint equations for this sample problem is listed; the row type (E for equality and N for free), then the row name is printed in accordance with the symbolic naming convention of Fig. 4-3. The second section of part (b), and continuing in part (c), lists the variable, the columns in which it appears, and its corresponding coefficient. Similarly, the last section, labeled RHS, gives a summary of those rows (constraint equations) which have non-zero right-hand-sides.

Part (d) provides a cross-reference list of variable number versus variable name for use in the interpretation of the output from the B3CAV2 program. The last section of part (d) indicates that there are 25 rows and 61 columns in this sample problem. Note that in each case these are one more than was indicated previously because the cost row and the right hand side variable, respectively are now included. Finally the upper bounds, computed by the GENLCP program, are listed for the master variables; the minus sign here is superfluous.

Part (e) prints a cost summary on each vehicle along with the components of the inherited fleet. Then in the last section of part (e) and continuing in part (f), the task (alternative) tables are reproduced.

GENERATING THE MATRIX FOR THE LEAST COST PHASE-IN PROBLEM

FILENAME= SAMPLE STARTING YEAR = 1972 LAST YEAR = 1974
 WILL INPUT 4 VEHICLE TABLES, AND 3 TASK TABLE, AND 5 PERIOD TABLES.

READING IN A VEHICLE TABLE
 LIGHT 1970 10

READING IN A VEHICLE TABLE
 MEDIUM* 1972 10

READING IN A VEHICLE TABLE
 MEDIUM 1973 10

READING IN A VEHICLE TABLE
 TRAILER 1976 10

READING IN A PERIOD TABLE
 1970 1970

READING IN A PERIOD TABLE
 1971 1971

READING IN A PERIOD TABLE
 1972 1972

READING IN A PERIOD TABLE
 1973 1973

READING IN A PERIOD TABLE
 1974 1974

READING IN A TASK TABLE
 1 3 3

READING IN A TASK TABLE
 2 4 5

READING IN A TASK TABLE
 3 4 5

VEHICLE NAME VARIABLE NAME
 OPTIONAL P+D VEHICLES

MEDIUM* X01
 OTHER VEHICLES
 LIGHT X02
 MEDIUM X03
 TRAILER X04

Fig A-5(a) GENLCP Output For Sample Problem

NAME SAMPLE

ROWS

* E SUMX01
 * E SUMX02
 * E SUMX03
 * E SUMX04
 * E PC01
 * E PC02
 * E PC03
 * E IN02PM1
 * E IN02P10
 * E IN04PM1
 * E X01P01
 * E X02P01
 * E X04P01
 * E T01P01
 * E X01P02
 * E X02P02
 * E X03P02
 * E X04P02
 * E T02P02
 * E X01P03
 * E X02P03
 * E X03P03
 * E X04P03
 * E T03P03
 * N COST

COLUMNS

(PARTIAL LISTING)

* X01	SUMX01	-1.0000
* X02	SUMX02	-1.0000
* X03	SUMX03	-1.0000
* X04	SUMX04	-1.0000
* P01	PC01	1.0000
* P01	PC02	-1.0000
* P02	PC02	1.0000
* P02	PC03	-1.0000
* P03	PC03	1.0000
* W02M100	COST	-.0108
* W02M100	IN02PM1	1.0000
* W02M101	COST	.0116
* W02M101	IN02PM1	1.0000
* W02M101	X02P01	-1.0000
* W02M102	COST	.0238
* W02M102	IN02PM1	1.0000
* W02M102	X02P01	-1.0000
* W02M102	X02P02	-1.0000
* W02M103	COST	.0345
* W02M103	IN02PM1	1.0000
* W02M103	X02P01	-1.0000
* W02M103	X02P02	-1.0000
* W02M103	X02P03	-1.0000
* W02M100	COST	-.0015
* W02M000	IN02P00	1.0000
* W02M011	COST	.0112
* W02M011	IN02P00	1.0000
* W02M011	X02P01	-1.0000
* W02M012	COST	.0256
* W02M012	IN02P00	1.0000
* W02M012	X02P01	-1.0000
* W02M012	X02P02	-1.0000

Fig A-5(b) GENLCP Output For Sample Problem

*	W020003	COST	.0342
*	W020003	IW02P00	1.0000
*	W020003	X02P01	-1.0000
*	W020003	X02P02	-1.0000
*	W020003	X02P03	-1.0000
*	W04M100	COST	-.0025
*	W04M100	IW04PM1	1.0000
*	W04M101	COST	.0147
*	W04M101	IW04PM1	1.0000
*	W04M101	X04P01	-1.0000
*	W04M102	COST	.0314
*	W04M102	IW04PM1	1.0000
*	W04M102	X04P01	-1.0000
*	W04M102	X04P02	-1.0000
*	W04M103	COST	.0430
*	W04M103	IW04PM1	1.0000
*	W04M103	X04P01	-1.0000
*	W04M103	X04P02	-1.0000
*	W04M103	X04P03	-1.0000
*	P030103	X02P03	78.0000
*	P030103	X04P03	18.0000
*	P030103	T03P03	1.0000
*	P030203	X01P03	24.0000
*	P030203	X02P03	48.0000
*	P030203	X04P03	10.0000
*	P030203	T03P03	1.0000
*	P030303	X01P03	50.0000
*	P030303	X04P03	16.0000
*	P030303	T03P03	1.0000
*	P030403	X02P03	48.0000
*	P030403	X03P03	24.0000
*	P030403	X04P03	10.0000
*	P030403	T03P03	1.0000
*	P030503	X03P03	50.0000
*	P030503	X04P03	16.0000
*	P030503	T03P03	1.0000
*	X010303	SUMX01	1.0000
*	X010303	X01P03	-1.0000
*	X010303	PC03	.0060
*	X010303	COST	.0086
*	X020303	SUMX02	1.0000
*	X020303	X02P03	-1.0000
*	X020303	PC03	.0039
*	X020303	COST	.0093
*	X030303	SUMX03	1.0000
*	X030303	X03P03	-1.0000
*	X030303	PC03	.0060
*	X030303	COST	.0086
*	X040303	SUMX04	1.0000
*	X040303	X04P03	-1.0000
*	X040303	PC03	.0120
*	X040303	COST	.0070
*RHS			
*	RHS1	PC01	.1500
*	RHS1	PC02	.2500
*	RHS1	PC03	.3000
*	RHS1	IW02PM1	60.0000
*	RHS1	IW02P00	20.0000
*	RHS1	IW04PM1	4.0000
*	RHS1	T01P01	1.0000
*	RHS1	T02P02	1.0000
*	RHS1	T03P03	1.0000
*ENDATA			

Fig A-5(e) GENLCP Output for
Sample Problem

REFERENCE LIST FOR COLUMN NUMBERS AND NAMES

1	2	3	4	5	6	7	8	9	10	11
0000004	X01	X02	X03	X04	X03	X03	X03	X04	X04	X01
5	P02	P03	W02M100	W02M101	W02M100	W02M100	W02M100	W02M100	W02M102	W02M102
11	X02M103	K020000	W02J001	W02M102	W02J001	W02M100	W02M100	W02M100	W02M102	W02M102
15	W04M100	W04M101	W04M102	W04M103	W04M102	W04M102	W04M102	W04M102	W04M105	W04M105
21	P010201	P010001	P010001	P010101	P010001	P010001	P010001	P010101	X010102	X010102
25	S0101	X020101	X020101	X020102	X020101	X020101	X020101	X020102	X020103	X020103
31	X040101	X040102	X040102	X040103	X040102	X040102	X040102	X040103	S0401	S0401
35	P020202	P020202	P020202	P020202	P020202	P020202	P020202	P020202	P020202	P020202
41	X030203	X030203	X030203	X030203	X030203	X030203	X030203	X030203	X030203	X030203
45	X040202	X040202	X040202	X040202	X040202	X040202	X040202	X040202	X040202	X040202
51	S0402	P030103	P030103	P030103	P030103	P030103	P030103	P030103	P030103	P030103
55	P030503	X010203	X010203	X010203	X010203	X010203	X010203	X010203	X030303	X030303
61	RHS1									

IMPORTANT DATA ITEMS FOR INPUT TO B5CAVLP
 NUMBER OF ROWS (INCLUDING COST) IS 25
 NUMBER OF COLUMNS (INCLUDING RHS) IS 61
 UPPER BOUNDS FOR VEHICLES IN ORDER FROM X1 THRU XN ARE

- 157.0000
- 224.0000
- 95.0000
- 51.0000

Fig A-5 i) GENICP Output For Sample Problem

VEHICLE NAME	VARIABLE NAME	SAL/TRUNC COST	O AND M COST	R AND D COST	RETENTION RATE	YEAR FIRST AVAILABLE	LIFE IN YEARS
MEDIUM*	X01	.0060	.1400	.3000	1.0000	1972	10
LIGHT	X02	.0030	.1200	0.0000	1.0000	1970	10
MEDIUM	X03	.0060	.1400	0.0000	1.0000	1973	10
TRAILER	X04	.0100	.1600	0.0000	1.0000	1970	10

COMPONENTS OF THE INHERITED FLEET

	1970	1971
NUMBER OF X02	60	20
NUMBER OF X04	4	-0

TASKS REQUIRED IN PERIOD FROM 1972 THROUGH 1972

TASK 01 - PERFORMED BY 1.00 FORCE ELEMENT(S), WITH SCALE FACTOR EQUAL 1.000

* VARIABLE X01 X02 X04 X

ALTERNATIVE	1	2	3	67	16
	0	20	42	43	7
				0	12

Fig A-5(e) GENLCP Output For Sample Problem

The reader should note that very little new information is produced by the GENLCP program — for the most part, GENLCP merely formats, reorders, checks, and performs bookkeeping operations in preparation for entry to the BBCAV2-REPGEN algorithm.

The first data deck for the BBCAV2-REPGEN algorithm is prepared as illustrated in figure A-6. We first assign a solution name in entry 1. The first, second, fourth and fifth fields of entry 2 are omitted as described on page 4-26. The third field in entry 2 indicates that there are four concave cost functions. The zero in the sixth field suppresses printing of the subroutine calls; the 1 in field No. 7 prints a listing of the primal iterations of each linear program; and the 1 in field No. 8 prints the entire set of LP solutions. Fields 9 and 10 are the standard specifications for the size of the array BLIST. The 1 in field No. 11 prints the column numbers and their corresponding values for each node. The last field, set to 20, establishes the limit on the number of nodes that will be evaluated prior to termination.

Entry 3 has four fields which establish (1) a tolerance factor of 0.005 (i.e., the solution will be within one-half of one percent of the theoretical optimum), (2) a program time limit of 90.0 seconds prior to termination (the solution to the sample problem actually used only 48 central processor seconds), (3) that no initial solution (basis) will be input, and (4) that we wish to obtain a detailed output. The second data deck for the BBCAV2-REPGEN algorithm is prepared as illustrated in figure A-8. As discussed on p. 4-35, the REPGEN data deck is very easy to prepare since most cards are duplicates of the GENLCP data deck. After the title card, the vehicle tables are inserted with cards of type 2 deleted. The period tables come next using only the header cards and cards of type 1. Note that the period designators have been inserted on all cards of type 1 in columns 11 and 12. The ENDTABLE card in entry 24 ends the data deck.

When the BBCAV2 program in the BBCAV2-REPGEN packet is loaded and processed, the printed output gives complete information vis-à-vis the optimal solution as well as all intermediate nodal solutions. The printout is long and involved and is in a coded format. The REPGEN program in the BBCAV2-REPGEN packet will decode the BBCAV2 output and

```

1 SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL
2           4           0           1           1           25           131           1           20
3           0.005           90.0           0.0           1.0

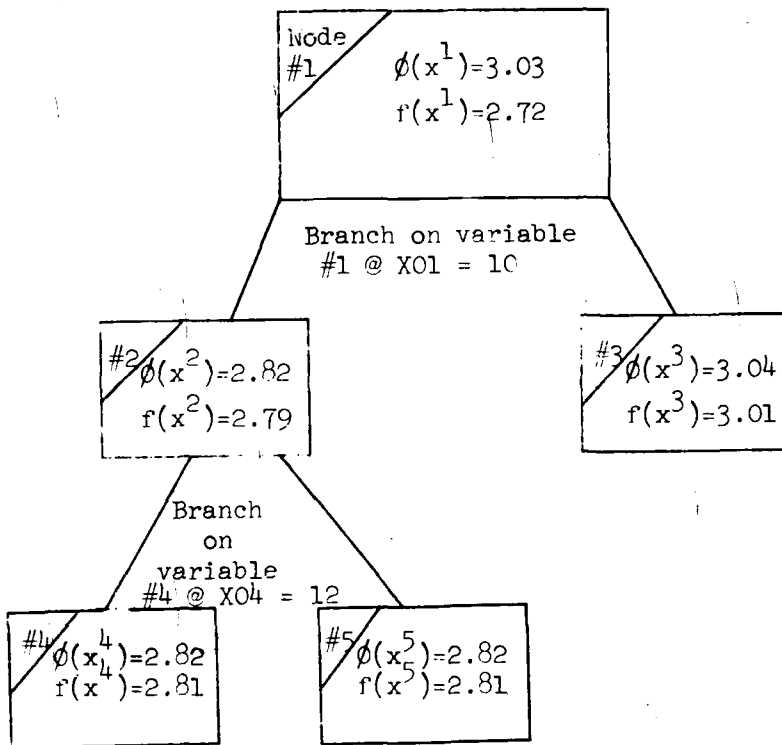
```

Fig. A-6 The Sample Problem Data Deck For the BBCAV2 Program

present all essential information, hence, we neither present nor discuss the BBCAV2 output. After some experience with these programs, the user may suppress all intermediate information if he so desires. We do present here a branching tree (like that of Fig. 2-2b) to lend further clarity to the intermediate solution.

The "tree" of intermediate solutions (see Fig. A-7) is illustrative of the iteration process of the branch-and-bound algorithm. Node #1 represents a complete linearization of the non-linear problem and establishes the first reference solution, $\phi(x^1) = 3.03$ million dollars. A lower bound to all solutions, $f(x^1)$, is also determined and equals 2.72 million dollars. The branching rule is then applied and variable #1 is selected, at the branching value of $X_{01} = 10$. The linear programs associated with node #'s 2 and 3 are then evaluated. Node #2 yields a better reference solution of $\phi(x^2) = 2.82$. Node #3 is found to contain no better solution than $\phi(x^2)$ because $f(x^3) > \phi(x^2)$, hence node #3 need no longer be considered. The next best branching variable is #4, at a value of $X_{04} = 12$. The linear solutions to nodes 4 and 5 yield identical results, indicating a solution at the bound. The process terminates here because the smallest lower bound is within one-half of one percent of the current reference solution. Detailed information regarding the optimal (and final) solution is obtained through the REPGEN program.

REPGEN in the BBCAV2-REPGEN algorithm is then processed resulting in the output of figure A-9, parts (a) and (b). Part (a) contains an overall COST INFORMATION summary together with a breakdown of the number of PURCHASED RESOURCES (vehicles). Part (b) illustrates a breakdown of the STORED (mothballed) RESOURCES and the TOTAL RESOURCES USED by period. From these tabulated results, we have constructed a bar chart in figure A-10 to better illustrate the optimal solution. In this display, the results of a cost minimization over time are illustrated by a bar for each year and each vehicle. The height of the bar is a measure of how



$\phi(x^i)$ = the actual (non-linear) solution for node i .

$f(x^i)$ = the lower bound (linear) solution for node i .

Figure A-7 A Branching Tree for the Sample Problem

1	SAMPLE	1972	1974	4	3	5	
2	VEHICLE						
3	MEDIUM*	1972	10				
4		.006	.14	.30	1.0	.006	
5	VEHICLE						
6	LIGHT	1972	10				
7		.003	.12	.20	1.0	.003	
8	VEHICLE						
9	MEDIUM	1973	10				
10		.006	.14	0.	1.0	.006	
11	VEHICLE						
12	TRAILER	1972	10				
13		.010	.16	0.	1.0	.012	
14	PERIOD						
15	1970 1972	01.15					
16	PERIOD						
17	1971 1971	02.25					
18	PERIOD						
19	1972 1972	01.15					
20	PERIOD						
21	1973 1973	02.25					
22	PERIOD						
23	1974 1974	03.3					
24	PERIOD						

Figure A-8 The Sample Problem Data Deck For The REPGEN Program

many vehicles were selected - the color black indicates vehicles existing or retained - a dotted section indicates vehicles purchased - a blank section indicates vehicles stored (mothballed) for later use. One can observe the trend toward a fleet of only medium trucks and trailers (perhaps because of the high labor costs of operating so many light trucks). The MEDIUM* trucks are not chosen for 1972 because of the high purchase cost for early delivery. The slack is taken up by a large purchase of trailers in this first year; the trailers are needed in the later years anyway. Storage of a few trailers is indicated in 1973.

SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

COST INFORMATION

	* R. AND D *	* PROCUREMENT *	* OPERATING *	* SALVAGE *	* TOTAL *
PERIOD 01		.112	1.060	.019	1.152
PERIOD 02		.199	.856	.026	1.029
PERIOD 03		.034	.956	.001	.989
TOTAL	0.000	.345	2.872	.047	3.170

TRUNCATION VALUE FOR RESOURCES = .348

SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

PURCHASED RESOURCES

	* MEDIUM*	* LIGHT *	* MEDIUM *	* TRAILER *
PERIOD 01	0.000	0.000	0.000	12.000
PERIOD 02	0.000	0.000	42.667	0.000
PERIOD 03	0.000	0.000	7.333	0.000
TOTAL	0.000	0.000	50.000	12.000

Fig. A - 3(a) REPGEN Output For Sample Problem

SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

STORED RESOURCES

	* MEDIUM*	* LIGHT	* MEDIUM	* TRAILER
PERIOD 01	0.000	0.000	0.000	0.000
PERIOD 02	0.000	0.000	0.000	2.793
PERIOD 03	0.000	0.000	0.000	0.000

SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

TOTAL RESOURCES USED

	* MEDIUM*	* LIGHT	* MEDIUM	* TRAILER
PERIOD 01	0.000	67.000	0.000	16.000
PERIOD 02	0.000	3.578	42.667	13.207
PERIOD 03	0.000	0.000	50.000	16.000

Fig. A - 2(b) REPGEN Output For Sample Problem

12
11
10
9
8
7
6
5
4
-

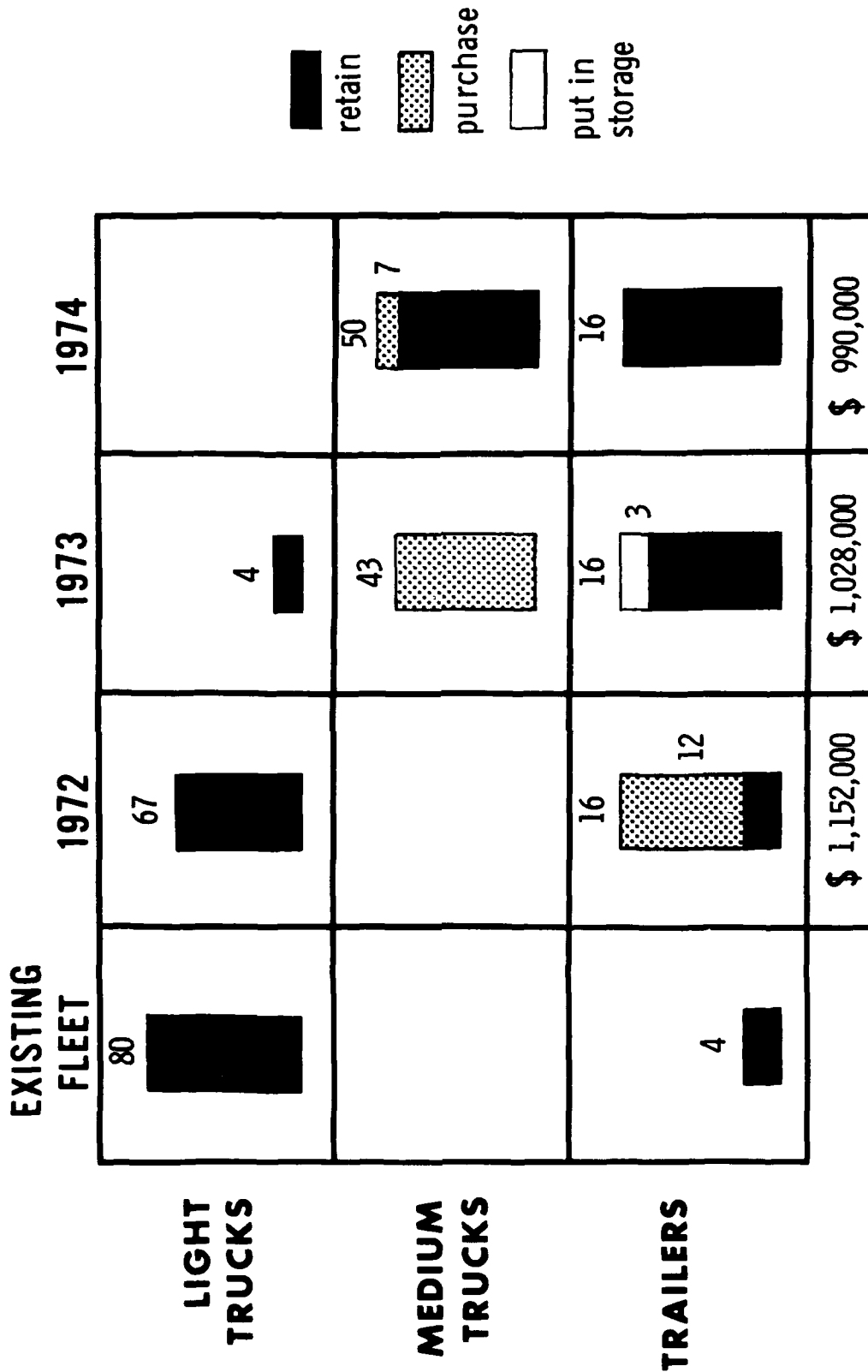
(perhaps a surprising result but, under the circumstances, a reasonable one since buying medium trucks or retaining a larger number of small trucks for this sub-period are very costly alternatives). Finally, one can observe the relatively high start-up costs due to the purchase of the entire trailer fleet in 1972, and then most of the medium truck fleet in 1973. Nevertheless, these high start-up costs yield the least total cost over the planning period.

The COST INFORMATION summary indicates a total real cost (that which must be allocated) of 3.17 million dollars. This differs from the branch-and-bound solution value of 2.82 million dollars by the truncation value; i.e., it is deemed proper to include the truncation credit for purposes of optimization, but this dollar credit (unlike salvage) is not considered to be available for other purposes. From the procurement cost summary, it would appear that none of the procurement cost constraints were binding; however, the reader should recall from p. 4-36 that these data were calculated from the linear procurement estimates and, upon close inspection of the BBCAV2 print-out, one could observe a binding cost constraint in the second period. As previously intimated on page A-7, this relatively poor correlation between linear estimate and actual value (in this case in the medium vehicles) can give erroneous results.* For cases in which the discrepancy is excessively large (judged not so in this sample problem) the programs should be reprocessed with a better linear estimate. The restriction of available funds in period two is probably the cause of the retention of the light vehicles, in lieu of the purchase of the full complement of medium vehicles for that period. Finally, one should observe that the RESOURCE summaries in general contain non-integer values — a consequence of the continuum of solutions to linear programming problems. It is incumbent upon the user to provide a physically meaningful interpretation to such results — as we have done for this sample problem in figure A-10.

* The difficulties introduced here due to a poor linear estimate can be avoided in the future by employing a recently developed modification to the current algorithm.

Figure A-10

A Sample Optimal Plan for the Ace Trucking Company



APPENDIX B
SUBROUTINE DESCRIPTIONS

APPENDIX B

MATRIX GENERATOR ROUTINE DESCRIPTIONS

GENLCP - reads and analyzes input data, creates column names and row names, determines non-zero values of the matrix of coefficients, creates the MPS360 file, and outputs the documentation listing.

YRCOST (J) - has parameter J, which is vehicle number, and determines for this vehicle all cost information (see Chapter 4 for detailed description).

YINTERP (NVR, NTR, NRY) - determines for all tasks (NTR) which of the NVR vehicles will not have been developed by the year NRY, and eliminates from those tasks all alternatives in which the "non-existent" vehicles are accomplishing something which could be done by an existing vehicle.

MATFILL (N, M) - creates from the MPS360 file the file for BBCAV2 which is an N-row by M-column matrix of coefficients, and also creates the reference list for matching column numbers and names.

MAIN PROGRAM ROUTINE DESCRIPTIONS

BBCAV2 - is the programmed implementation of the main logic structure of the branch and bound algorithm; selects node from branching tree, branches on it defining two new nodes, and determines when optimality has been achieved.

BCX1 - defines the initial node of the branching tree and establishes the problem framework in which the main program will iterate.

INITA (NCF, N, M) - reads the matrix file from tape and stores it on disk in the form acceptable to the LP; the parameters N and M define the number of columns and rows in the matrix, and the parameter NCF is the number of columns having nonlinear cost functions.

GETPHI (KFX, XPHI, PHI, SUMPHI) - evaluates the nonlinear cost functions (see Chapter 4 for a detailed description).

TABOUT (IRT) - outputs the general information concerning the node being evaluated, and if IRT = 1, also prints the nodes on the branching list.

READIN - transfers the input basis to the file which the LP uses for storing its current basis on.

NXB (XT, SIGMAT, NXB) - determines the best branching candidate, NXB, given the present X-vector, XT, and the node information, SIGMAT.

TIMEC - determines how long the program has been running, prints the time or interrupts depending on whether or not this time is less than the input maximum.

GETASQ (NOES, ELM, JSQ) - orders the elements of the vector ELM, of length NOES, in ascending sequence, keeping the index variable, JSQ, associated with the vector in the corresponding sequence.

GETC (KCX, BLT, ULT, CT) - determines the slope of a straight line on the cost function from the lower bound, BLT, to the upper bound, ULT, for the KCX variable and stores this in the cost vector, CT.

PRESET - initializes core storage at beginning of the algorithm.

SET (TMMAX) - initializes the time limit which is used by the routine TIMEC by adding the limit, TMMAX, to the clock time.

PARAMS - reads and stores the information on the parameter and bound cards of the input deck.

LP

LP is the linear programming system driving subroutine which directs the overall solution stages through:

SETUP - which initializes all data for the A matrix files and the solution bookkeeping.

MAPIN - which introduces any prior solution known for the problem.

INVERT - which solves the problem equations to generate the current solution represented by the basis inverse and the values of the basic and key variables in the current solution, or an artificial solution.

PRIMAL - which solves the linear programming problem.

MAPOUT - which stores the solution found and loads it into the output vectors IX and X.

LP begins with the overall common definitions for the LP system, and must be loaded first since the common statements /A/, /B/, /CORE/, /ROWTYP/, /LXX/, /XX/, /NAMES/ overwrite the smaller dimensions specified in later subroutines which can remain unchanged regardless of problem size.

AJ contains the operating core columns and the complete basis inverse B at $AJ(IORG) = B(IORG)$, see below.

The first stage is to specify the A matrix files IA1 - used for the A matrix less GUB rows, INPUT - used as the source of the unpacked A matrix by columns, written in binary, one column per record, and IMAP - used by MAPOUT, MAPIN and INMAP as a file for the BCD MAP cards defining the basis, initial and/or final.

Files IA1 and IA2 are specified as blocked. Meaning that each physical disc write or read is of as many columns as the buffer sizes for IA1 and IA2 will allow, and not just one column, as actual written in FORTRAN. This considerably reduces the disc access time denoted as PP time on the CDC 6400.

NWAJ, the number of words in AJ is used to compute the number of columns available in core.

The second step is to initialize the LP calling parameters with the LP calling arguments. The matrix generator locates the cost row

ICOST as the last row MROWS which is INPUTM for the linear program common/INPUT/. The right hand side is JRHS, placed as the last column NCOLS which is INPUTN for the common/INPUT/. The number of bounds NBDS is the number of changes NCHGS, the calling parameter, and the first NCHGS columns are bounded by BBCAV convention. The values of the bounds are in UBS.

The second stage ends with specification of LP system print and termination controls.

LP Cut-Off

STATUS terminates the program if any linear program takes longer than TMAX seconds or K5 iterations ITRN. The termination causes a MAPOUT allowing restart of the linear programming system but not necessarily BBCAV.

Diagnostic Snapshots

A snapshot of the current solution and column format can be had from XCHECK by setting K4 to

$$K4 = 1000 \times N1 + N2$$

giving a snapshot of iterations N1 through N2 inclusively. K4 = 0 suppresses XCHECK. The format is explained in the XCHECK subroutine writeup.

Print Control

This is achieved by K3.

K3 = 0 prints everything

K3 = 1 prints no LP system output except error messages.

Other values of K3 give a selective print of all or some of the respective outputs according to the prime factors of K3.

Specifically K3 should be a product of primes and

$$K3 = 2 \times 3 \times 5 \times 7 \times 11 \times 13$$

gives all print options. To obtain selective control:

1. To print the MAPIN cards as read K3 has factor 3.
2. Inversion diagnostic data from INVERT on infeasibilities of the current solution as found, the columns rejected during

an inversion or reinversion and the final infeasibility if any, K3 has factor 13.

3. MESSG prints linear programming system verb entry names and entry times and several messages if K3 has factor 7.
4. STATUS prints the status of the PRIMAL iterations at beginning and end of K3 has factor 11.
5. MAPOUT places a basis inverse B, IBASIS, KEYS and BETA representing Restart data sufficient to avoid an initial invert on file INPUT if K3 has factor 2.
6. MAPOUT prints the solution status in packed format when called if K3 has factor 5.

Thus to obtain printouts of inversion diagnostics, a mapout, verb entry times and status data set $K3 = 5 \times 7 \times 11 \times 13$.

The third stage of the program LP calls the system verbs listed earlier.

The basis inverse is at B(IØRG) where IØRG is M^2 words down from the end of AJ, i.e. NWAJ. The remaining space in AJ is allocated to columns of which NCRMAX can be fitted in. There must be at least 5 columns slots available, three for CHECK to retain columns and two for work space. Fifty columns are recommended

Finally MAPOUT moves the current variable state to file IMAP, the solution to INPUT, the packed variable values to IXX and XX. IXX has the indices in ascending order of non-zero variables, and XX has the corresponding values. There will be between INPUTM and INPUTM + NBDS non-zero values followed by zeros.

Variable Lengths

In /IXX/, /XX/ IXX, XX should be set to 100 or INPUTM + NBDS if larger.

In /CORE/ AJ should be big enough to take the basis inverse $(INPUTM+1-L)^2$ words plus 10 to 100 columns at $(INPUTM+1-L)$ words each where L is the number of GUB rows.

In /ROWTYP/ IROWTP should be 100 or INPUTM+1 if larger than 100.

In /NAMES/ NAME should be 100 or INPUTM+1+S if larger, where $S \leq INPUTM$ is the number of inequalities and free rows.

In /A/ ALPHA should be 100 or INPUTM+1 if larger.

In /B/ BETA should be 100 or INPUTM+1 if larger.

All other commons are correctly sized in LP.

EXISTS

All exists from primal are via the subroutine EXISTS for the purpose of user parameter settings.

SETUP

This subroutine is the system verb which has the task of initiating the LP system when starting from scratch.

SETUP first of all initializes all LP system parameters, then examines the row types constructing a logical or slack column for each nonequality row and writes these to disc using calls to OUT of IO. An extra free row is incorporated for the phase 1 cost row and the logical columns for free rows are marked basic.

SETUP then reads the A matrix columns from the binary INPUT file and writes then out to disc using calls to OUT of IO. For each column the NAME vector is set to record the column type (free/null), the column GUB packet number or zero and the column bound index or zero. The right hand side vector is recorded in core in RHS.

Finally, SETUP rewrites IBASIS and the RHS vector to place the GUB row elements at the end. The count of GUB rows is recorded in L and the actual row count is reduced by L. The cost row marker ICOST is reset to its new position in the rearranged rows.

IO

This subroutine handles all disc to core transactions and keeps track of column bookkeeping.

OUT writes two files of columns of the A matrix writing one column in each file per call. The first file IA1 contains columns less their GUB elements. The second file IA2 contains columns less their GUB elements and any zero elements and is written in a packed format.

IN reads file IA1 cyclically up to NT times, in search of a particular column rewinding when appropriate. It is normally accessed for sequential columns by CHECK but INVERT uses it to locate basis, at-bound and key columns marked in the NAME vector.

INPCKD reads file IA2 cyclically up to NT times in search of a column rewinding when appropriate. It is only accessed in random forward increments searching for key columns and thus uses a packed file.

After NT reads, sufficient to locate any column, both entries cause an error message and dump.

Once IN or INPCKD have located the required column and read it to a slot in AJ(), the column index is loaded to the corresponding position in JA, its reject memory in JAREJ is cleared and its mnemonic (unused) is placed in JAK.

Thus it is not possible to read a column into core without adjusting the bookkeeping of what is in core.

MAPIN

This subroutine is the system verb which sets the bookkeeping of the column status and allows a restart from a previous status. It is designed to read a file IMAP generated by MAPOUT and loaded by INMAP to file IMAP.

MAPIN reads settings of NULL, BASIC, KEY and ATBND designated at random one type per card up to 4 columns per card for each type. Each type sets the column status marker in NAME appropriately.

Restarts

MAPOUT writes the LP system status onto the end of the INPUT tape file when MAPOUT is called, and provides an INVERSE card for MAPIN use. The INVERSE card causes MAPIN to check for an inverse plus bookkeeping data and the solution status on the INPUT tape, and read it if present.

INMAP

Is the entry designed to read the input card stream for MAPIN cards and load them onto file IMAP. It is terminated either by an end of file or end EMD card.

Manual preparation of MAP cards is possible and extensive checks in MAPIN will detect and avoid most errors.

INVERT

This is the system verb which inverts or reinverts the current basis as defined in NAME records and completes the basis with artificials.

When INVERT is called, an inversion occurs only if the current iteration exceeds ITNINV. When it does ITNINV is increased by INVF, and an "INVERT" message is printed.

INVERT first clears the basis records and the GUB packet basis column count, sets up a unit basis and for each GUB row without a key chooses the first valid GUB packet column as key. It then cycles the column status records in NAME until it locates a basic, key or at-bound column, which is retrieved by IN. Key and at-bound columns are accumulated in GAMMA, scaled by their packet righthand sides (if keys) or their bounds. If basic columns are in a GUB packet, the key is located by INPCKD, subtracted from the column and the result transformed and pivoted into the basis in a row determined by PIVOT.

When all NAME records have been checked and the columns incorporated or rejected by PIVOT, the basis record is completed with logicals or if necessary with artificials. The artificials are then constructed in DELTA transformed and pivoted into the basis. Finally, FEASCH is called to construct and check the solution feasibility.

NB. The MAPIN used can be partial, complete, redundant or nonsense.

FEASCH

FEASCH is called by INVENT to compute $\beta = B^{-1}\gamma$ given γ in GAMMA and B^{-1} in B. The resulting β elements in BETA are checked for feasibility and the basis is adjusted if infeasible until the resulting BETA is feasible and the phase I PHASE is 1 or 2.

The method is to cycle each element of BETA from 1 to M, compute it, check if it exceeds a bound then check if it is positive. If it exceeds a bound, the basic column is set "at-bound," and the bound is subtracted from that BETA(I) which then becomes negative and infeasible. If it is negative i.e., infeasible, its sign is reversed and the column is replaced by its negative artificial*, to pick up the infeasibility directly, (the artificial need not be transformed) and pivoted into the basis in place of the old basic column. Finally, if CUB rows are present the last L entries in BETA are filled with the values of the key variables.

Feasibility of the keys is maintained by calling KEYCH to move the infeasible key (essential packet) to a basic position in a non GUE row and processing it as above as an infeasible variable.

If any infeasibilities have been encountered, or the resulting Phase 1 cost is larger than CTOL, Phase 1 initiates otherwise Phase 2.

* The negative artificial of a column A_i is $-A_i + e_m$ is the mth column of the identity matrix. The negative artificial of an artificial $e_i + e_m$, is $-e_i + e_m$.

PRIMAL

This subroutine is the main LP verb which solves the LP problem phases 1 and 2.

PRIMAL notes its entry and time using MESSG, then picks up the cost row for its current phase 1 or 2, the appropriate π row in the inverse and sets the phase 1 row to free in phase 1 or equality in phase 2.

The basic solution cycle is counted by ITRN. If ITRN exceeds K5 or if CP time exceeds TMAX a MAPOUT is called by STATUS followed by EXIT.

The solution cycle proceeds with COLUMN to find an in-core column JCOL. If JCOL = 0 no column is found and the phase terminates. If the cost is zero in phase 1 this is the feasible solution termination, if phase 2 this is the optimal solution, if non-zero in phase 1 there is no feasible solution.

Next ROW is called for a pivotal row IROW. If IROW = 0 no row is found and the problem is unbounded.

Next the pivotal element is checked for size and degeneracy. If it is too small NREJ is indexed. If 5 bad columns have occurred an INVERT is called to check the inverse. If more than 100 bad columns have occurred the problem terminates either in phase 2 as optimal, or with a dump. If the pivotal element is okay, the cost change THETA * DJ(JCOL) is checked. If this is smaller than CTOL, NDEG is indexed. If more than NDEGLM degenerate columns have occurred and there are no more good columns the column is accepted. Otherwise in either case the old column is rejected and a new column is selected by COLUMN ignoring the previous selections.

Next the step is saturated to the bound on the column. If it exceeds the bound and the column is not at bound, the column is set ATBND. If the column is ATBND, the column is set free. In either case there is no pivot and the solution is corrected for the bound change but there is no basis change correction and NREJ = 1 to suppress pricing in the next iteration.

If the step is within the bounds, a basis change will be made. The rejected column is located first in the basis of $IROW \leq M$, then in the keys of $IROW > M$. If $IROW \leq M$ there is no key change, the new

column is pivoted in by PIVOT at IROW. If the new column is AT BND the step is off the bound and the new column value EPSI is corrected to the bound value less the step. Then the new column is made basic, the rejected column is made free, and the solution step made and the new basic column value set to EPSI.

If $IROW > M$ there is to be a key change. If the GUB packet is essential, it has other basic columns and the key is changed for one of these using KEYCH, then $IROW \leq M$ and the previous case follows.

If the GUB packet is not essential it has no basic columns, so the key is changed to the new column and the old key is dropped. The new key value $EPSI = THETA$ and a normal step is made as before without a pivot and pricing in the next iteration is suppressed.

After every pivot the rejected columns are cleared.

At the end of the iteration cycle STATUS reports the solution change.

After every row and column selection, or at any optimality stage, XCHECK is called for a debug which occurs if $K4$ is set > 0 . See LP for details.

STATUS

STATUS prints out the status of PRIMAL iterations every cycle under the headings.

PHASE = (IPHASE) - the LP phase 1 or 2.

ITER = (ITRN) - the LP iterations count.

TRY = (NTRY) - the number of iterations with the same set of columns in core + 100 x maximum number of tries.

VAL OBJECTIVE = (BETA(IC)) - the solution value of the current cost row.

NDJS = (NDJS) - current count of negative DJ's an estimate of non-optimality of the current core columns.

NARTS - the current number of artificial vectors present.

VALUE DJ IN = (DJ(JCOL)) - the value of the DJ for the column chosen to enter.

COL IN = (JPØS) - the internal number of the column chosen to enter.

CODE = (NAME(JPØS)) - the status of this column.

COL OUT = (JOUT) - the column rejected.

CODE = (NAME(JOUT)) - the status of the column.

NSCAN - the current number of rewinds of file IAL the A matrix plus the number of columns active in core, or columns read on disc.

Note: If JOUT is zero, no column was rejected and its code is zero.

If JCOL is zero or IROW is zero, these are taken as termination markers and the NOTE obtained in the STATUS call is printed e.g. PRIMAL--END, etc...

If JOUT > NT artificial code is constructed equal to the $10^9 \times$ IROW.

ROW

This subroutine is called by PRIMAL to locate the pivot row IROW in the selected column JCOL.

ROW first transforms the in-core column JCOL to the current basis representation in ALPHA, reconstructing the complete column including GUB elements which occupy the last L positions.

The row selection depends upon whether the column is at-bound or not, for if at bound the column represents the slack vector and the step is negative. For either case the minimum THETA is found which

(i) drives the resulting solution to zero or

(ii) drives the rejected column out at bound, depending upon the sign of the potential pivot element ALPHA(I).

These are case 2 and 3 for a normal column and cases 3 and 2 for an at bound column. Upon exit THETA is the step in row IROW, core-column JCOL, (JP~~OS~~ on disc) and ITYPE is 2 or 3 for the type of step.

If no row is found IROW = 0, ITYPE = 1 indicating an unbounded step and THETA = 1.E35.

COLUMN

This subroutine locates a potential column entry JCOL from those in-core, or calls CHECK to search all or part of the disc for more columns and uses these.

COLUMN counts NTRY selections with the current columns. If more than NCRMAX, or no columns exist in core it locates up to NCRMAX new columns with a call to DISC. If no columns are found the problem is optimal and JCOL = 0 at exit.

The in-core columns are then priced out, unless no pivot has occurred (NREJ or NDEG \neq 0) because of column rejection or DISC has just been called. PIKEY is always set to the current key price for the packet recorded in JPKTO. If a column is in a packet, its price is adjusted for the key price. All columns are priced apart from rejected columns.

The best unrejected column is now found by searching the DJ values. At bound columns have DJ reversed as they correspond to the slack column, and the number of negative DJ's is counted in NDJS.

If no good column is found, i.e. the best DJ is above the DJTOL threshold, DISC is called to search for more columns unless these columns are new, denoted by NTRY = 0 (no selections with these columns).

Upon exit JCOL is the in-core location of the best column found in-core (or from disc) called JPOS, or JCOL = 0 denoting no column. If JCOL = 0, JNCORE is reset to the number of columns in core (because DISC has deleted the count of columns that were there) in order to try to save a disc read in the next phase if any.

DISC

This subroutine checks the disc for more columns and selects those which are currently "not bad."

First DISC calls INVERT to see if the iteration count ITRN has exceed the next invert point and inverts if necessary.

DISC reads the columns in batches of NBCH columns serving 1 column/ batch. If fewer than NCRMAX columns are actually used these are read directly into core where they stay, once and for all. Alternatively the current file IAL position JNT is found by INPOS and DISC examines the columns starting at JNT + 1, proceeding cyclically, changing batch every NBCH columns. If the new packet number PKT is different and nonzero the new key is located and read over any unused old key, or into the next vacant AJ slot, by INPCKD.

The column type is found in JTYPE and null (0), basic (2) and key (4) columns are skipped. Free (1) and at-bound (3) columns are read by IN to the next location JORG. The new column is priced out correcting for its packet if $\neq 0$, and if the new price DJNEW is worse (\geq) than DJOLD (the best of the current batch) the column is skipped. Otherwise this column is preserved as IORG in the batch records and the best batch column DJ as DJOLD.

Every NBCH column, column IORG is saved if it is better than DJTOL and IORG is reset to the next vacant column.

DISC will work if the packets are disjoint, (separated by zero packet columns), and also if the packets are mixed up, (alternate columns in different packets) but with much loss of efficiency due to multiple key searches and rewinds of file IA2.

DISC always pulls in the key of each packet first for each packet, using the packed file IA2 regardless of where the key is located in the packet.

Subroutine KEYCH

Changes the key for an essential GUB packet, to one of its basic columns in the packet, selecting the first one. The basis inverse, B, solution in BETA and current column in ALPHA are corrected for this rearrangement.

Subroutine SETBND, SETBNB, SETNNN, SETKEY

Sets and unsets the state of a column J in NAME (J), to either free (1), null (0), basic (2), at-bound (3) or key (4), respectively.

Function DOT, DOTS

Computes the inner product x'y in either double and single precision, respectively.

Subroutine MAPOUT

Writes the states of the null, basic, key and at-bound columns onto BCD cards, placed on file IMAP, and also places the current inverse B solution BETA and basis bookkeeping IBASIS and KEY onto the end of input tape INPUT to allow instant RESTART.

Function BOUND

Returns the value of a column bound, if bounded, or 10^{70} if unbounded, or artificial.

PIVOT

This subroutine pivots a new column ALPHA into the basis inverse B at row IROW. If IROW is zero the best pivotal row is found.

IROW is zero the basis is checked for empty slots or slots containing the column disc index JPOS. If the latter is found, this row is used as IROW since this is SETUP's method of fixing logicals for free rows. For null basis entries PIV and IROW track the largest ALPHA element and its row, and this is used as the pivot element unless it is less than PIVTOL where upon the column is dropped with IROW = 0 as a marker.

If IROW is non-zero, the pivot ALPHA (IROW) is checked against the PIVTOL for possible errors. If the pivot is not unity, the inverse row IROW is normalized by the pivot. Then for every non-zero ALPHA entry at I, that multiple of the inverse pivot row is subtracted from the Ith inverse row (skipping the pivot row).

KEYFND - function

KEYFND find the location in core of the key column for the specified packet. If none is found in core it returns a value zero.

If the calling argument is zero KEYFND locates any key in core which has no associated GUB packet columns. If no key is found in-core it returns a value zero.

Otherwise the value of KEYFND is the column location 1 to NCRMAX.

ESCAPE

ESCAPE causes termination with a snapshot of the working core followed by a call to file 0. This will generate an abort condition suitable to generate a system dump. If it is desired to do this, use:

```
DEBUG.  
IGO.  
EXIT.  
DMP (LP, ESCAPE)  
7  
8  
9
```

This will dump the core using the labelled system dump from subroutines LP to ESCAPE, which should be first and last respectively.

Consult the variable list for a definition of the global variables.

All calls to ESCAPE are preceded by an ERRØR message of explanation of the fault condition.

XCHECK

XCHECK delivers a core snapshot if ITRN lies between N_1 and N_2 where $K4 = 1000 N_1 + N_2$. ($K4 = 0$ suppresses XCHECK.)

XCHECK prints using the following format.

- (a) Col 1 indexes the normal and GUB rows respectively.
- (b) Col 2 prints the basis IBASIS and KEYS respectively.
- (c) Col 3 is the current column representation ALPHA of JCOL.
- (d) gives the pivot position IROW.
- (e) Col 4 gives the current basic and key variables respectively.
- (f) step is THETA the proposed step, before bounding.
- (g) the column bound.
- (h) the selected column disc index.
- (i) is the list of core-column disc indices.
- (j) Col 5-14 is a list of 10 columns around the selected column in their current basis representation.
- (k) is a list of the column name codes at the XCHECK instant.

REPORT GENERATOR ROUTINE DESCRIPTIONS

REPGEN - the main program acts principally as a control program calling other routines to perform specific functions; determines if all solutions have been interpreted and initializes storage for each solution.

SETUP - reads input deck and reference list file into core storage.

INSOLN - interprets the meaning of each column in the solution and stores its value in the appropriate array(s).

YRCOST (J) - same as in matrix generator.

VALUES (N, ISTART, IEND, VAL) - determines cost information associated with each "X" or "W" type column in the solution; N is the number of the vehicle type, ISTART is its first year of existence, IEND is its last year of existence, and VAL is the number of those vehicles.

CINFO - this routine organizes, tabulates and outputs the table of cost information.

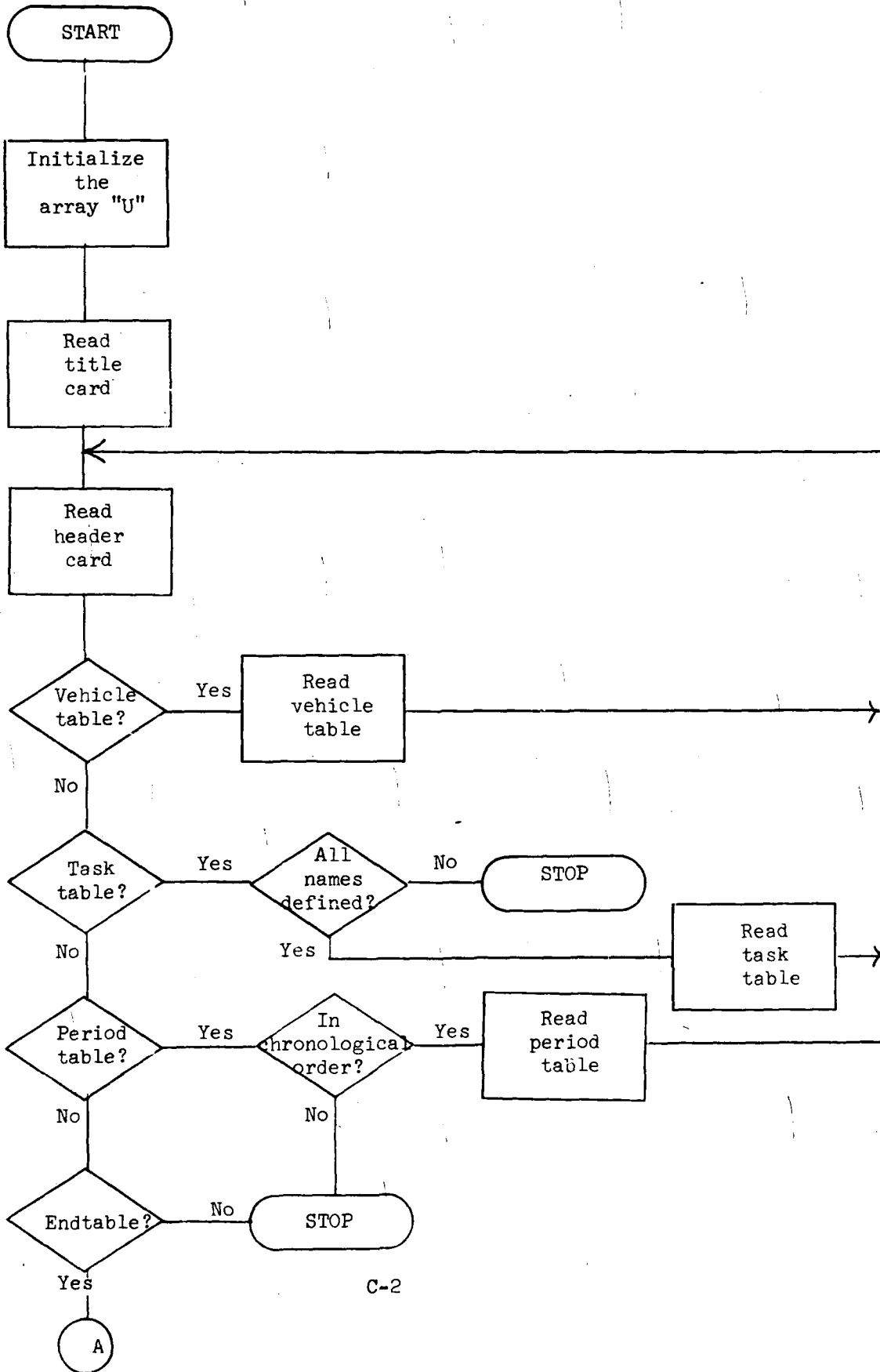
PINFO - this routine merely formats and outputs the last three tables of information; purchased resources, stored resources, and total resources used.

APPENDIX C

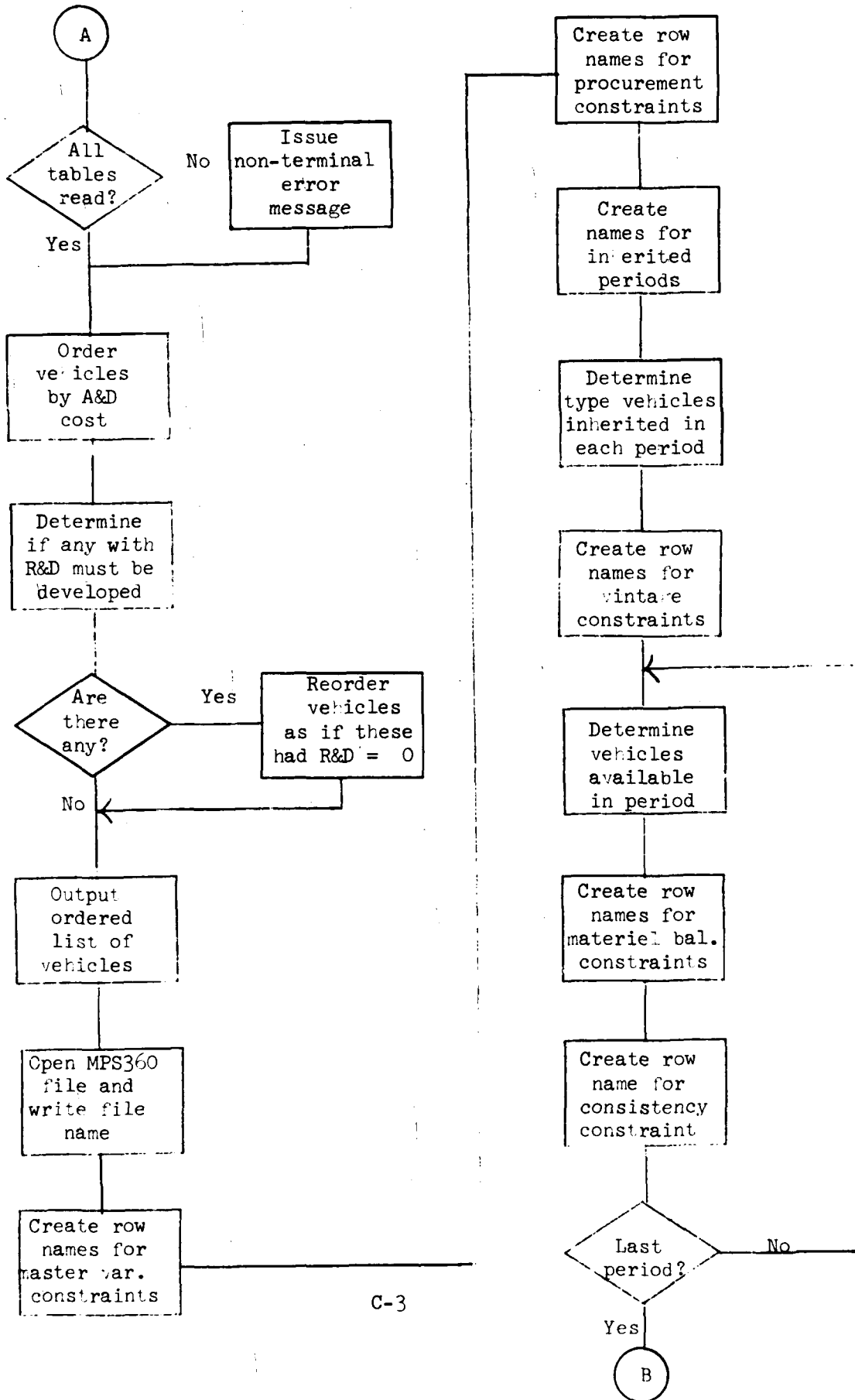
FLOWCHARTS

GENLCP.....	C-2
BBCAV2.....	C-12
REPGEN.....	C-82

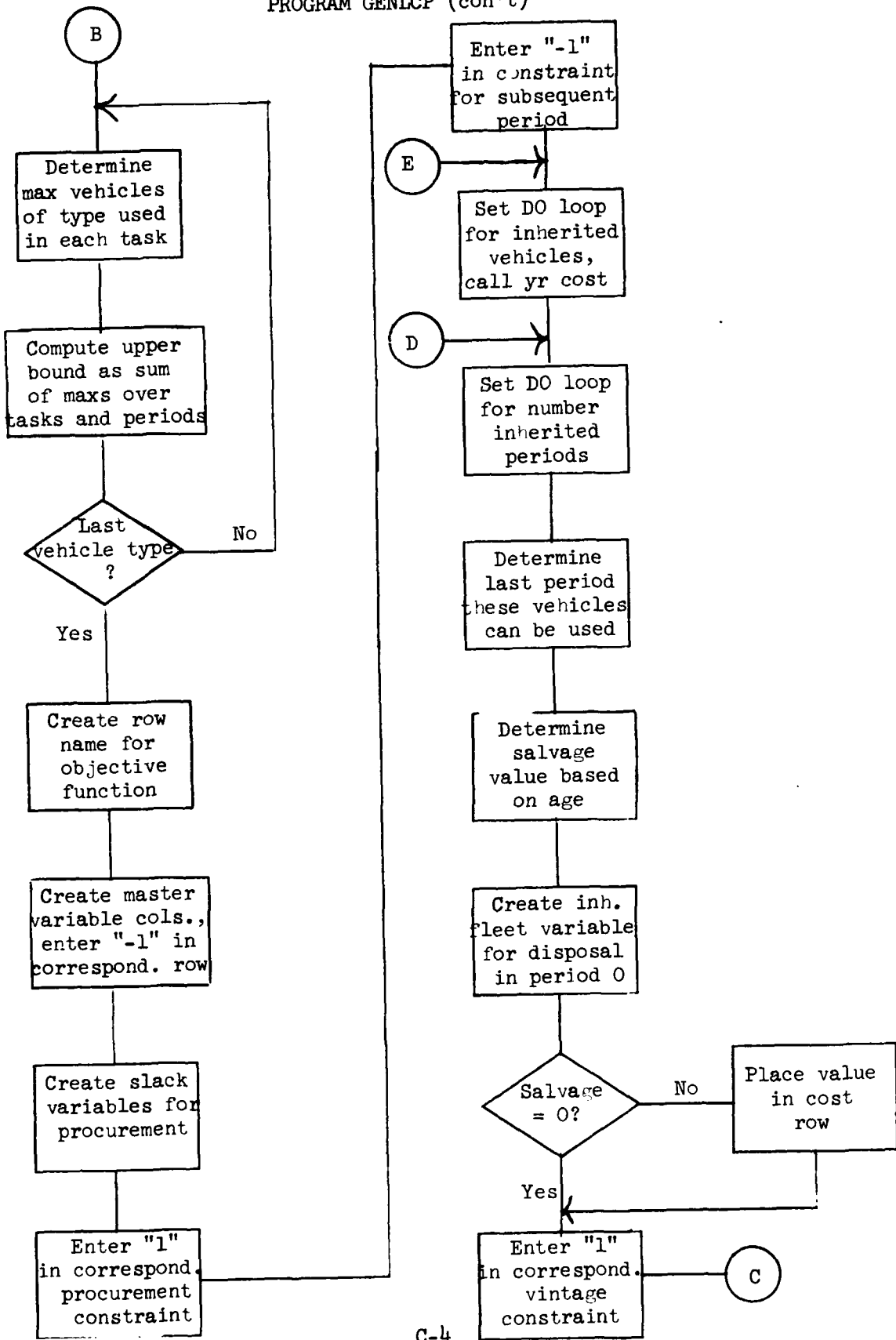
PROGRAM GENLCP



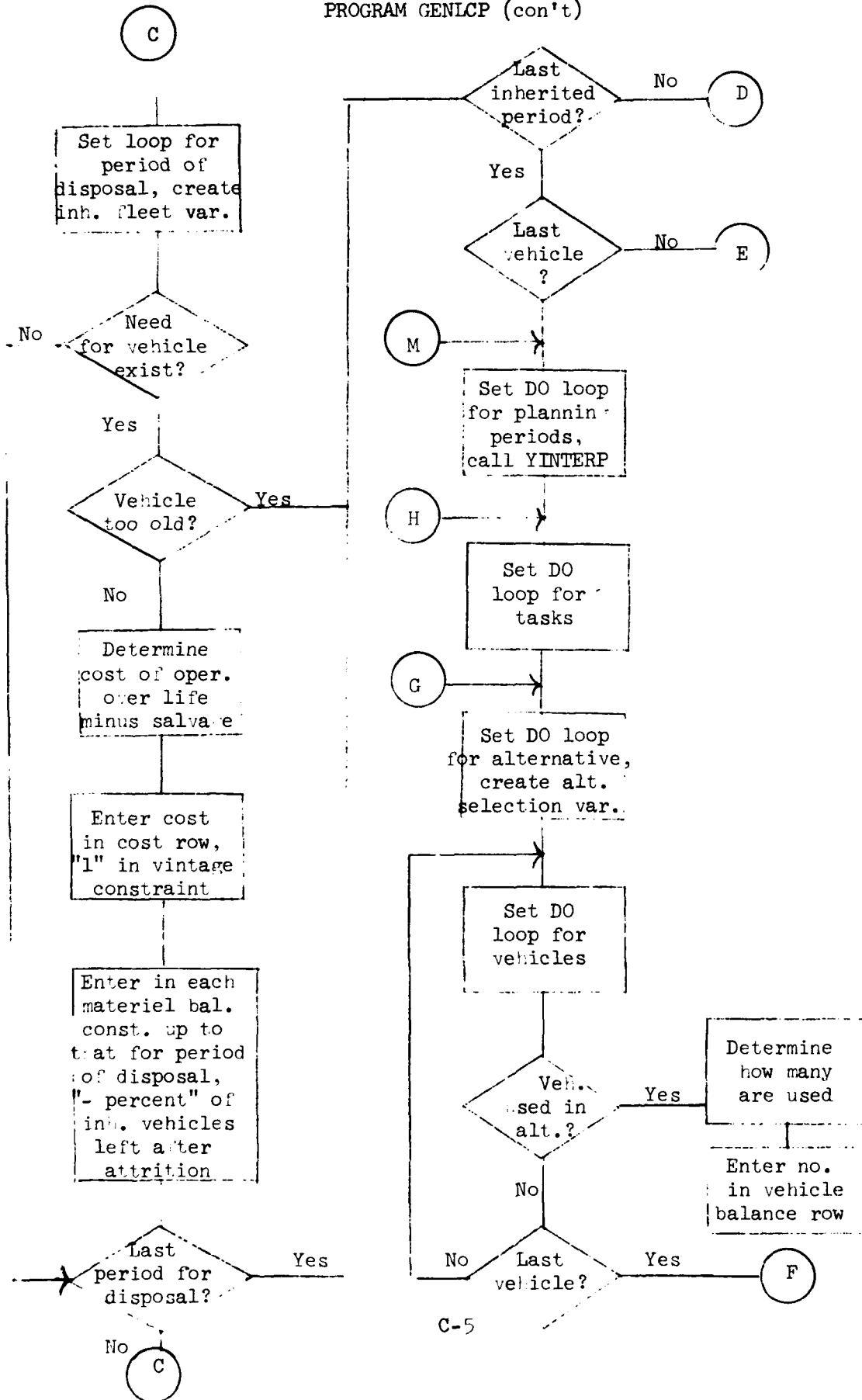
PROGRAM GENLCP (con't)



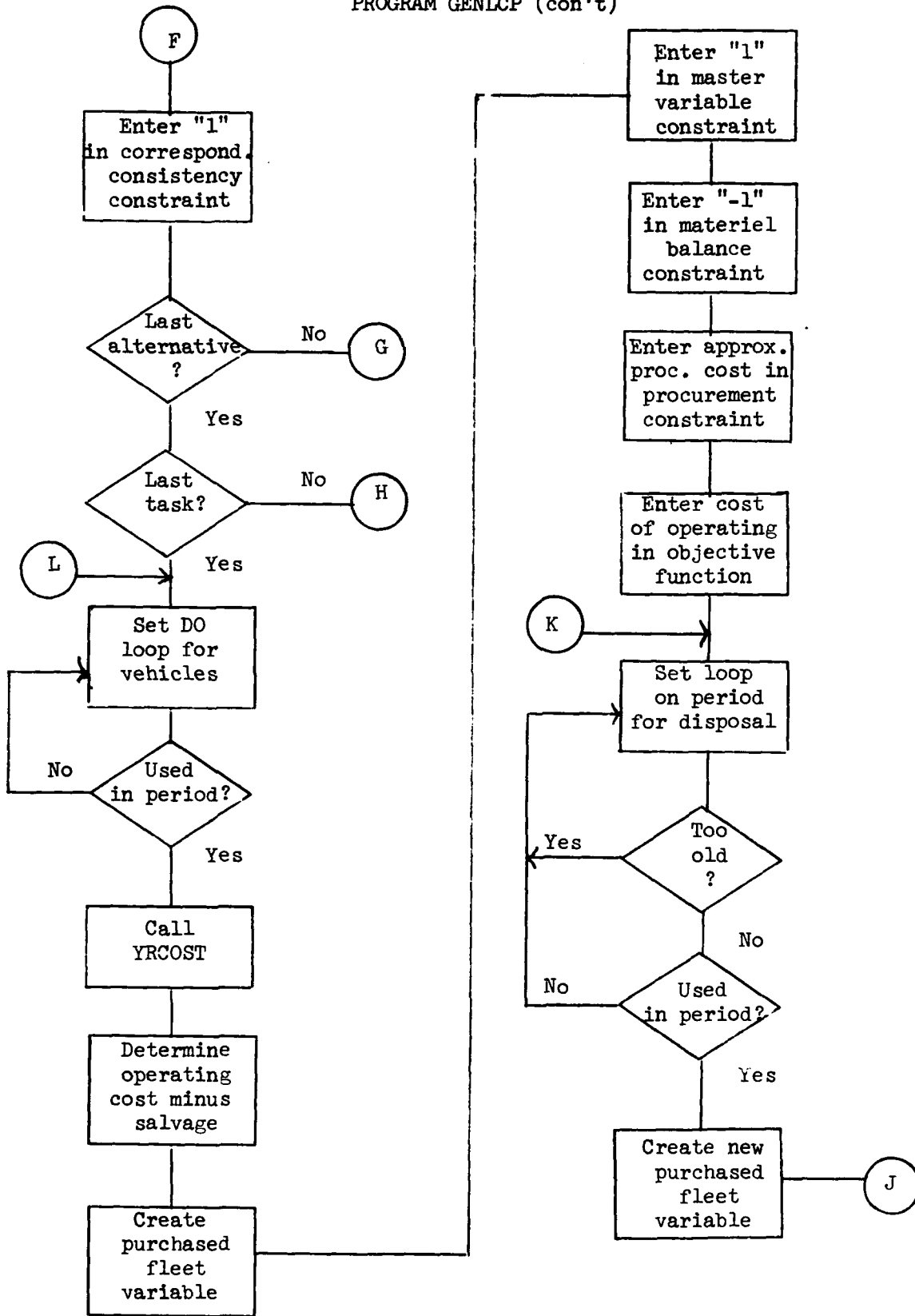
PROGRAM GENLCP (con't)



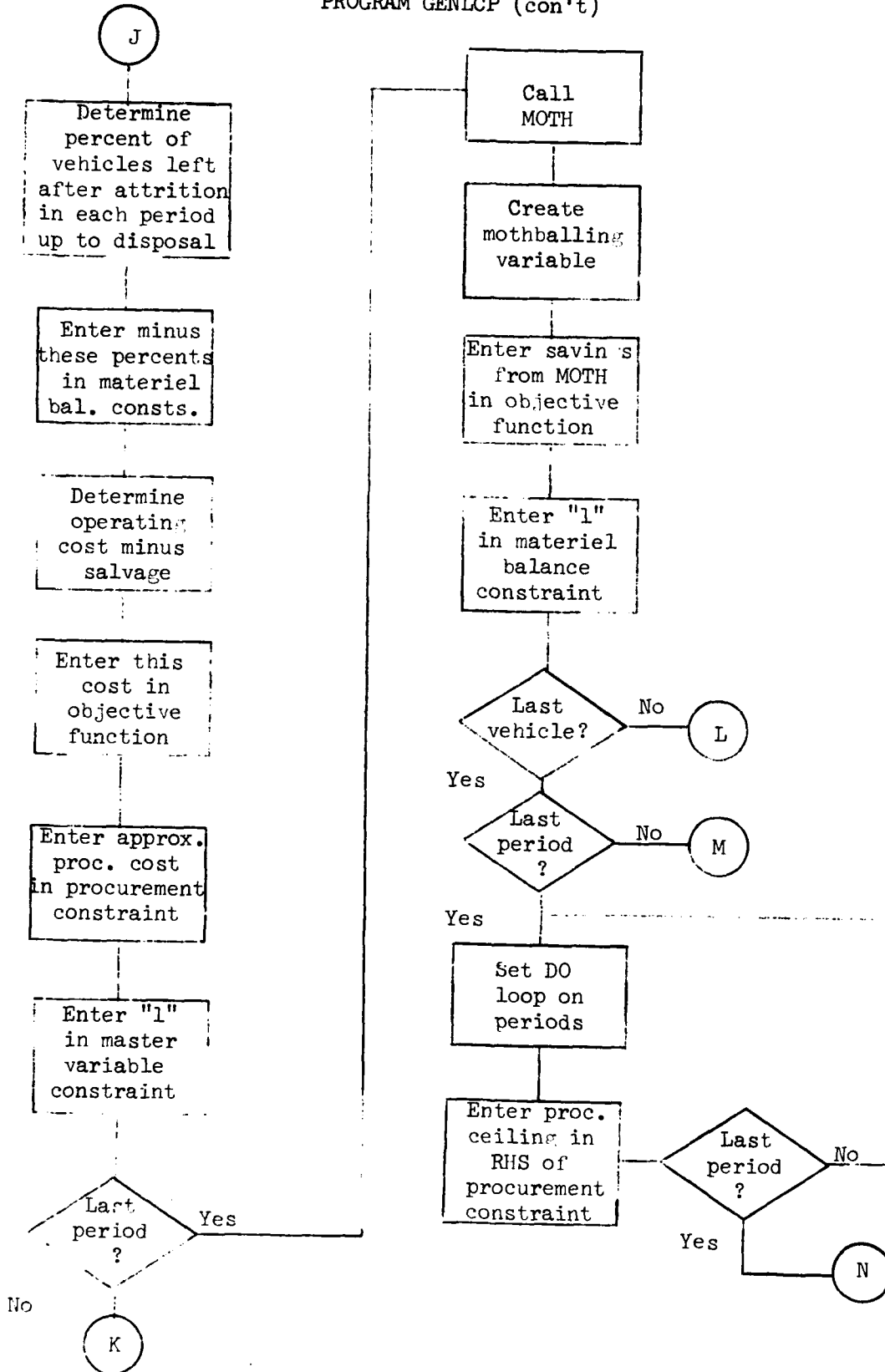
PROGRAM GENLCP (con't)



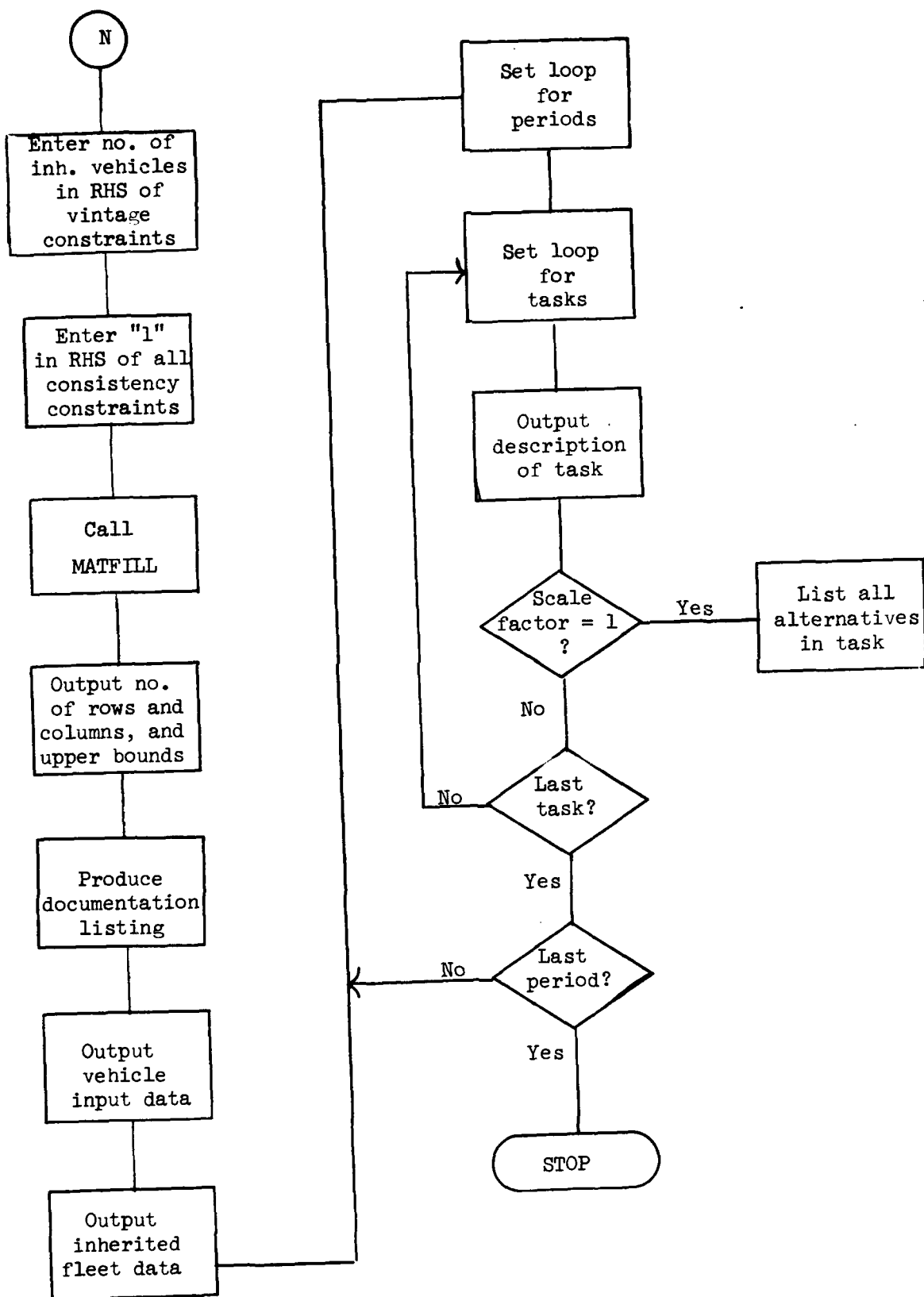
PROGRAM GENLCP (con't)



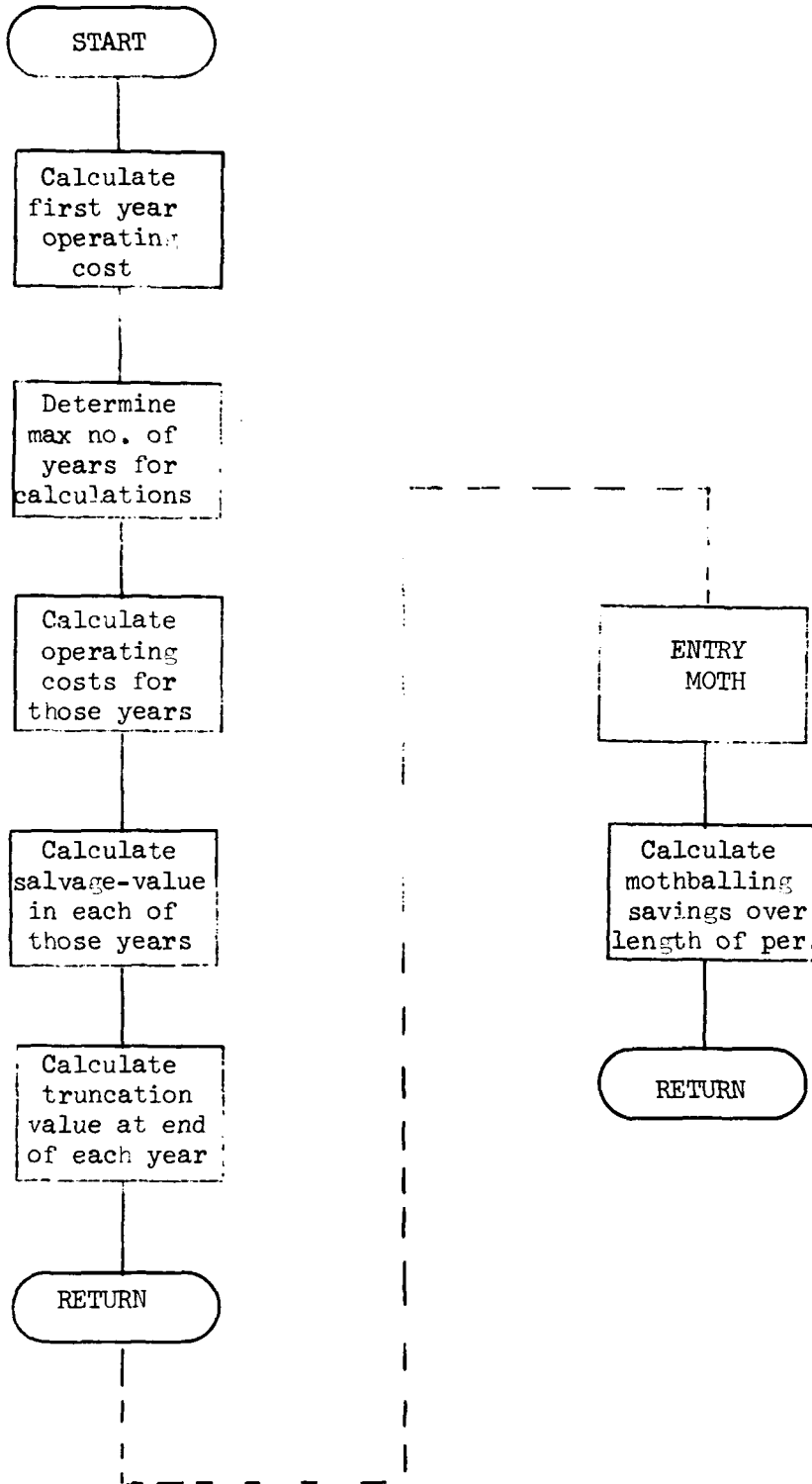
PROGRAM GENLCP (con't)



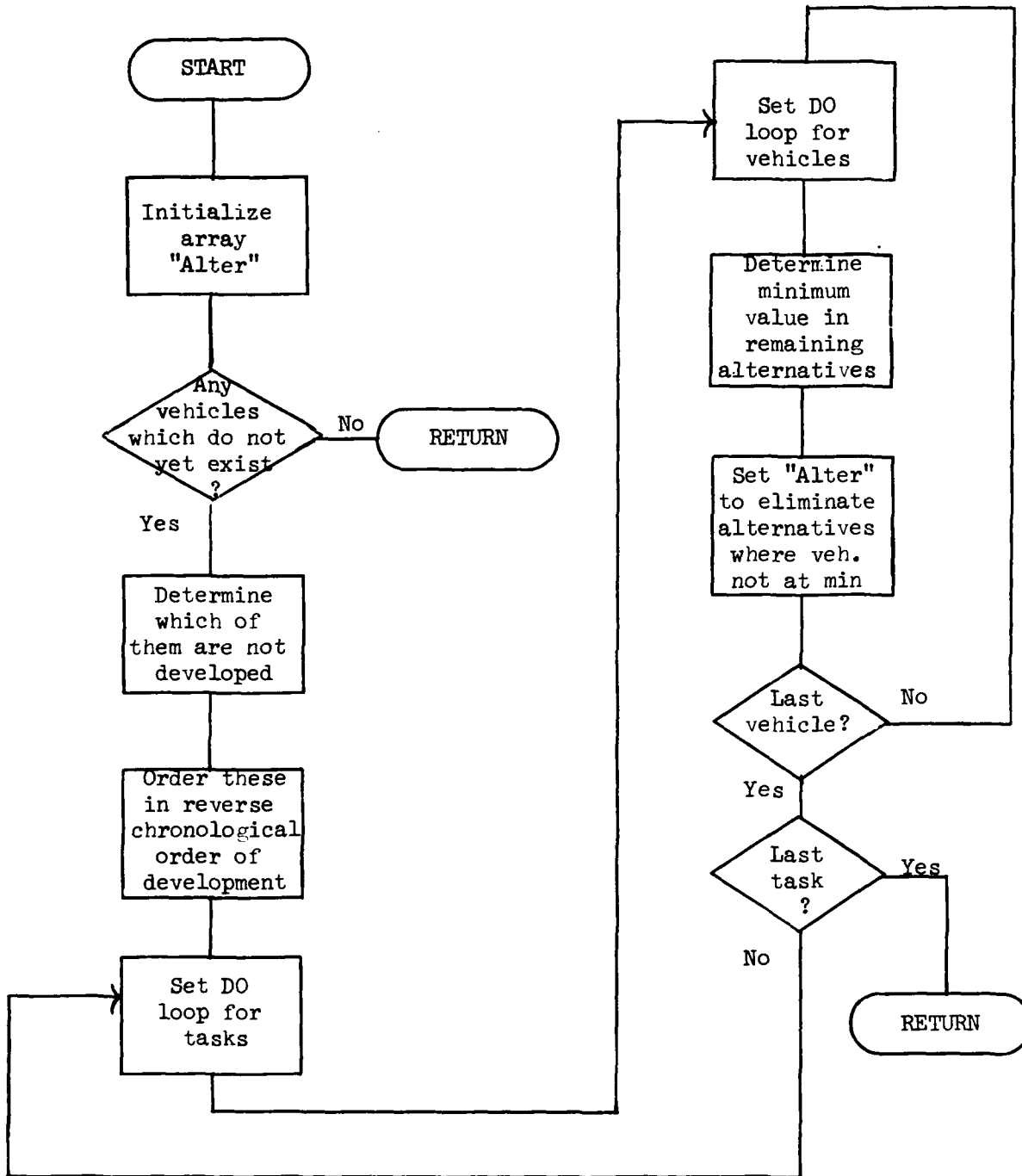
PROGRAM GENLCP



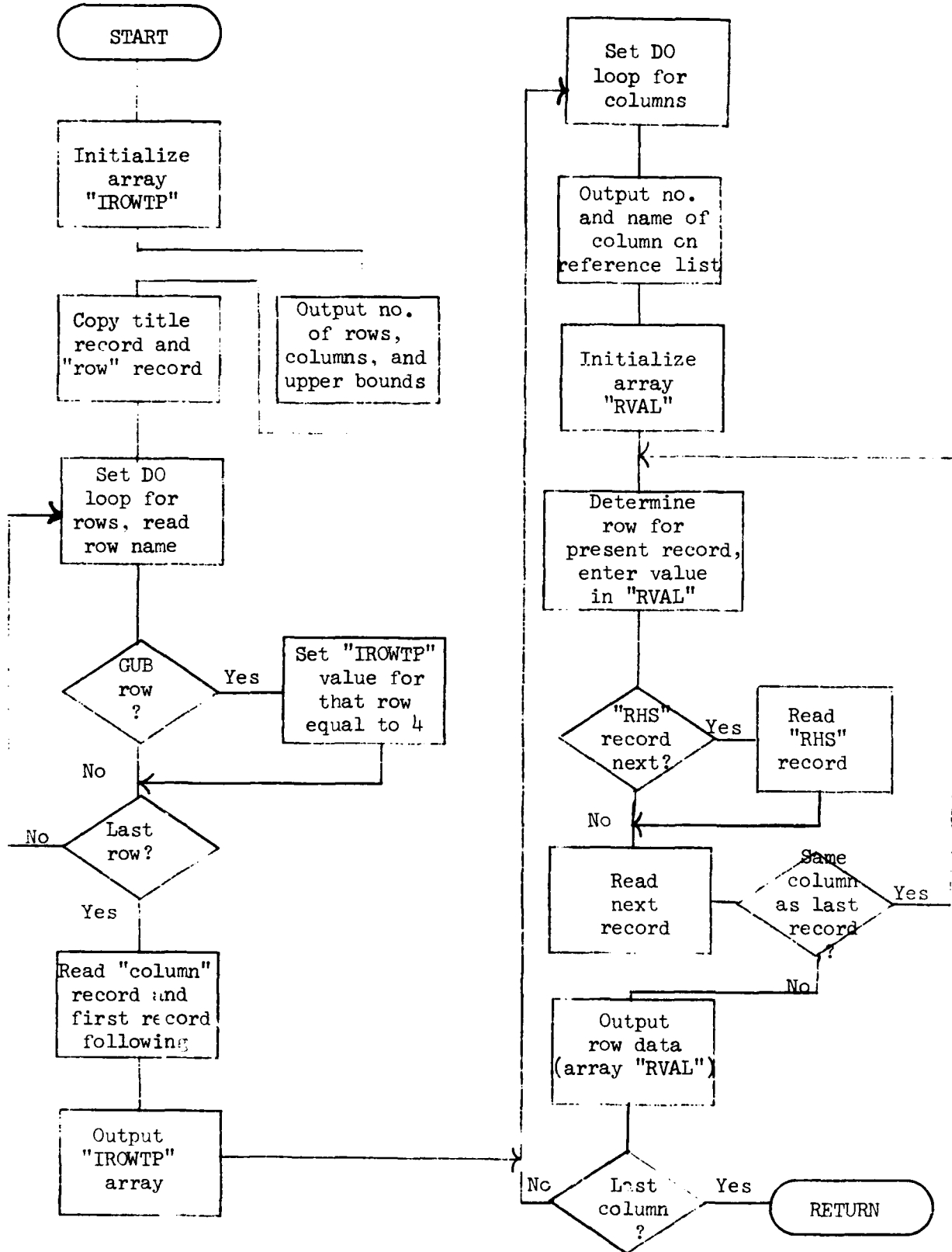
SUBROUTINE YRCOST



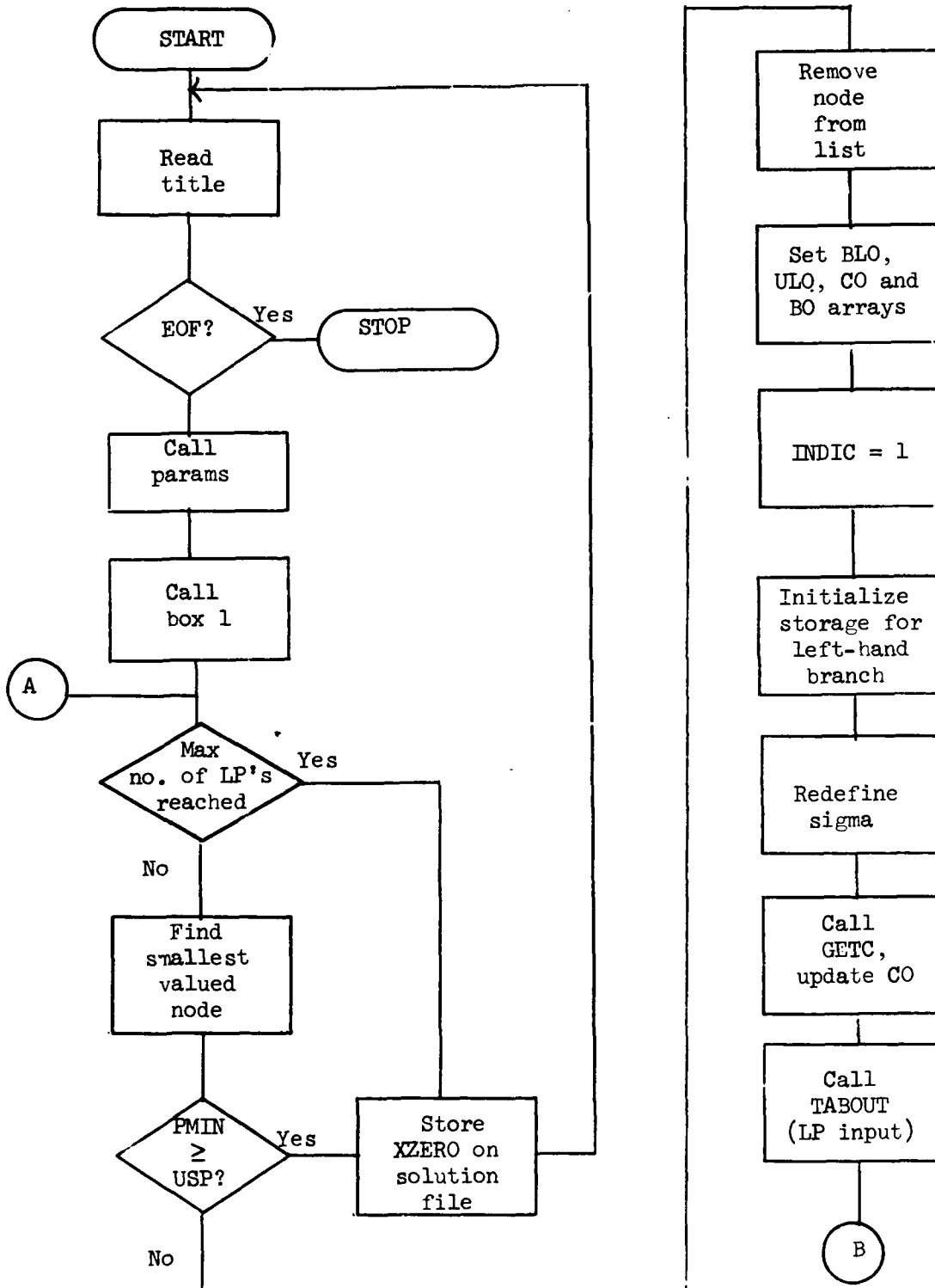
SUBROUTINE YINTERP



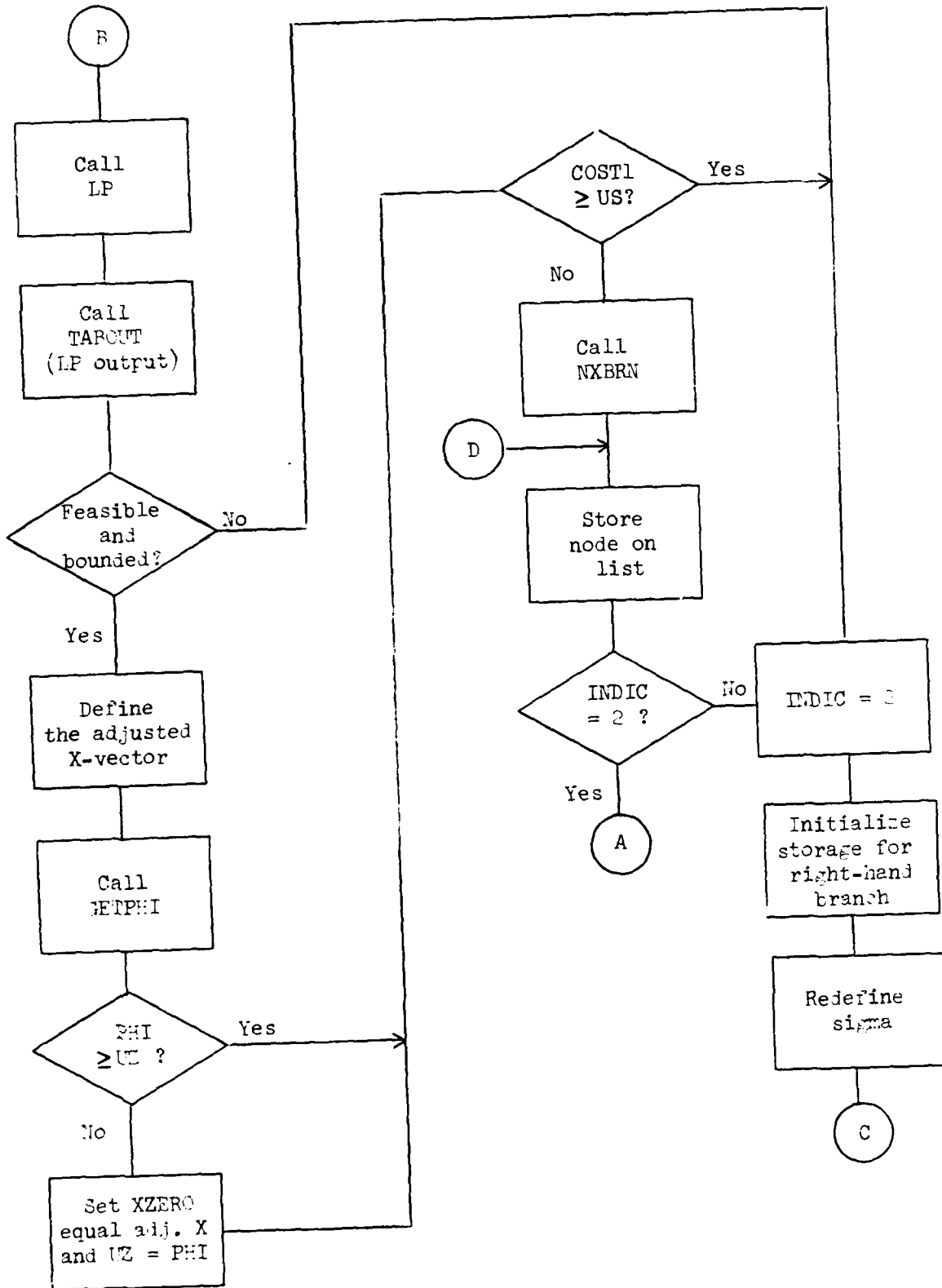
SUBROUTINE MATFILL



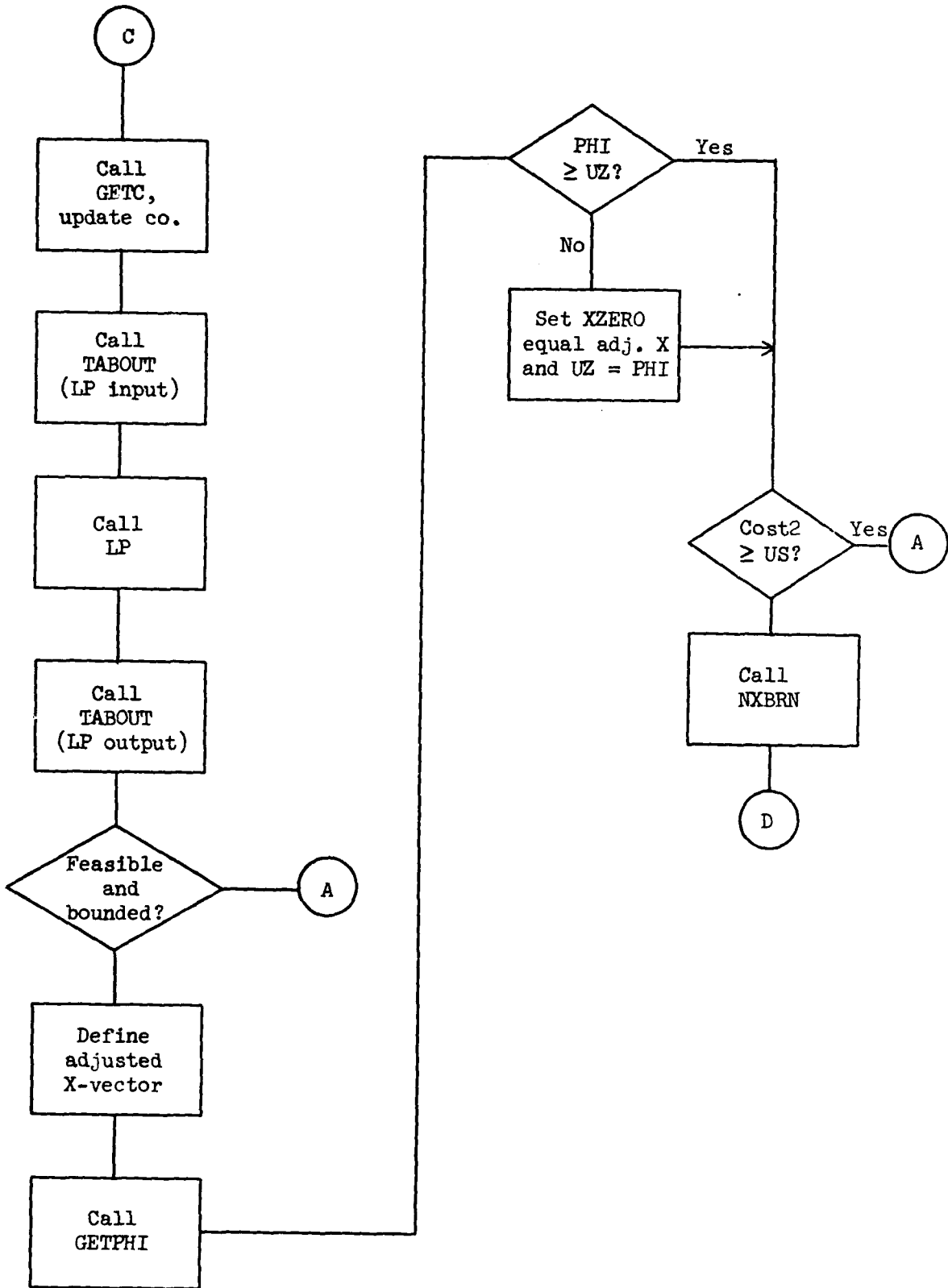
PROGRAM BBCAV2



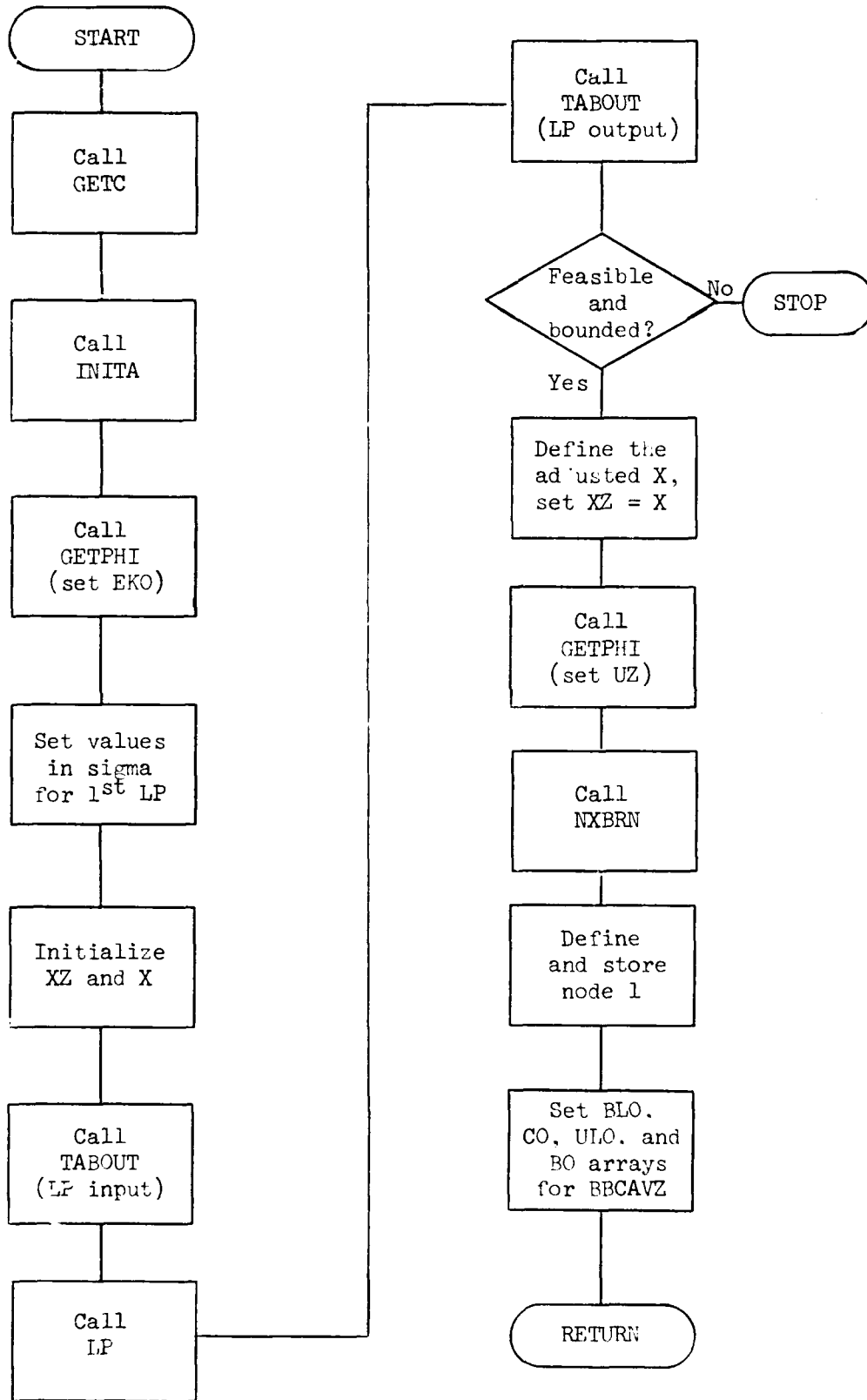
PROGRAM BBCAV2 (con't)



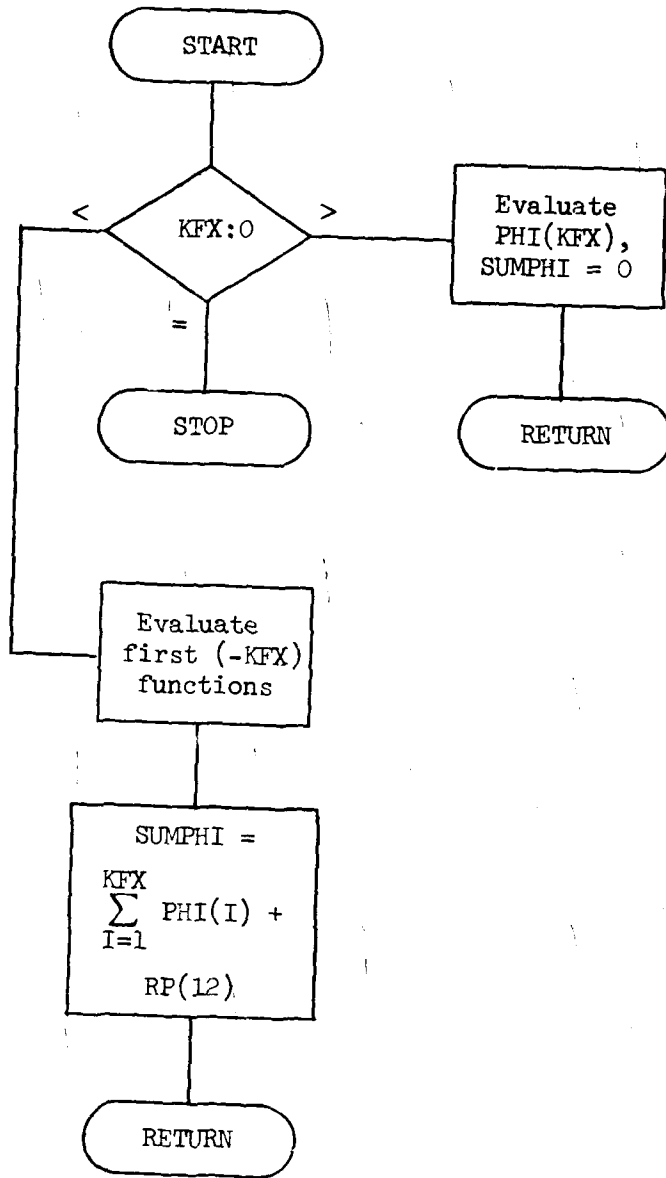
PROGRAM BBCAV2 (con't)



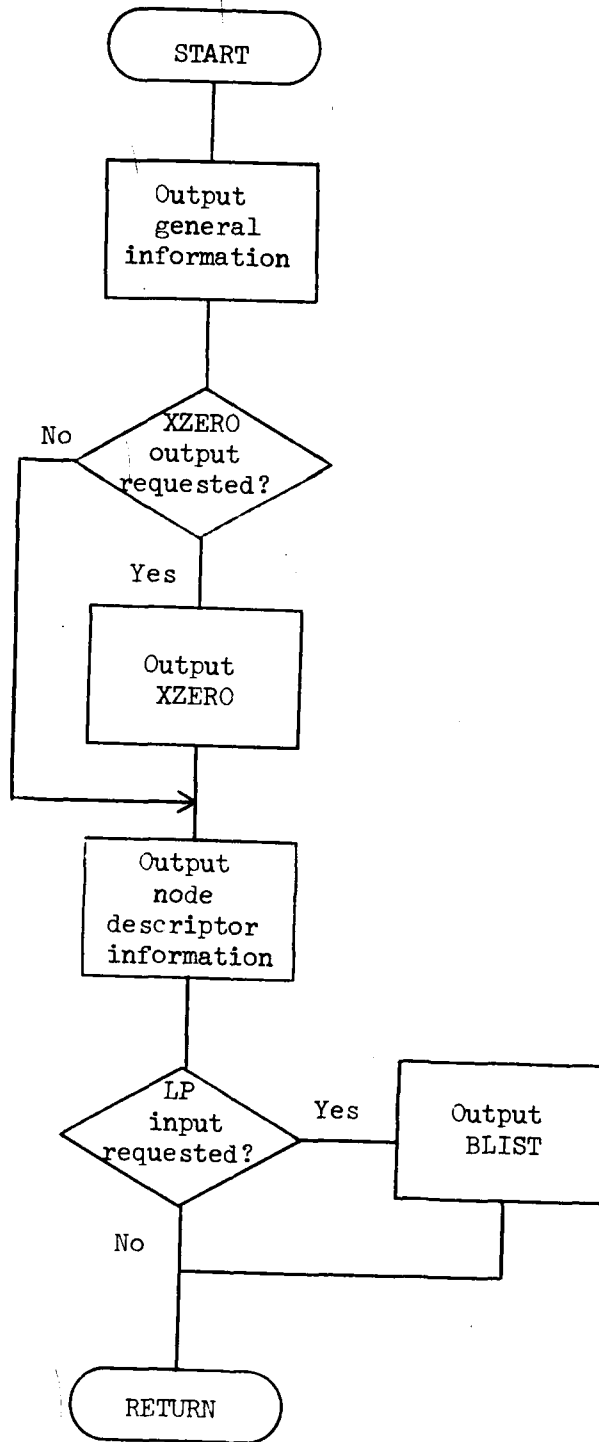
SUBROUTINE BOX1



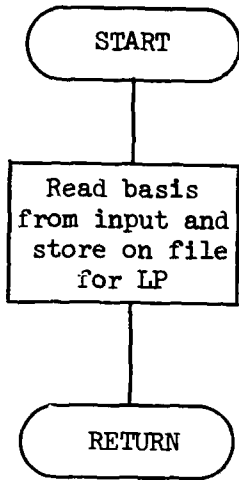
SUBROUTINE GETPHI



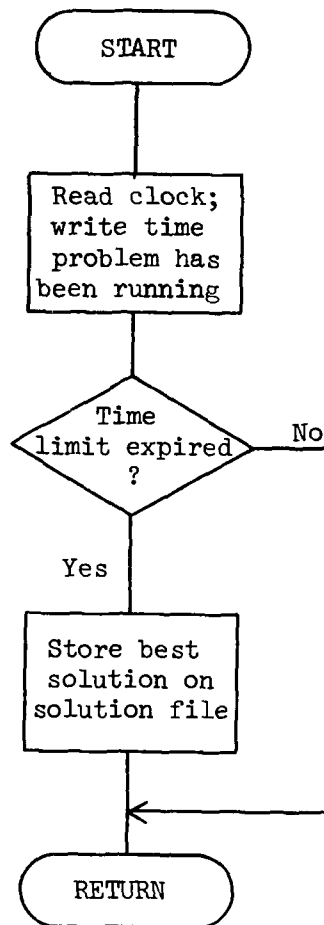
SUBROUTINE TABOUT



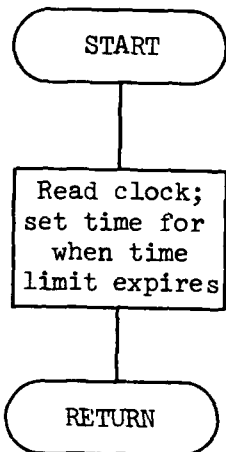
SUBROUTINE READIN



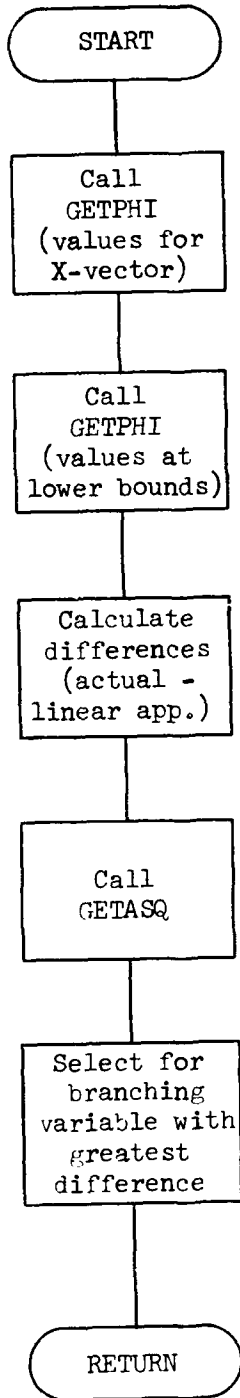
SUBROUTINE TIMEC



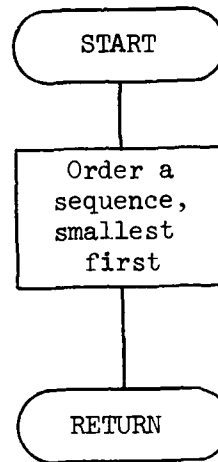
SUBROUTINE SET



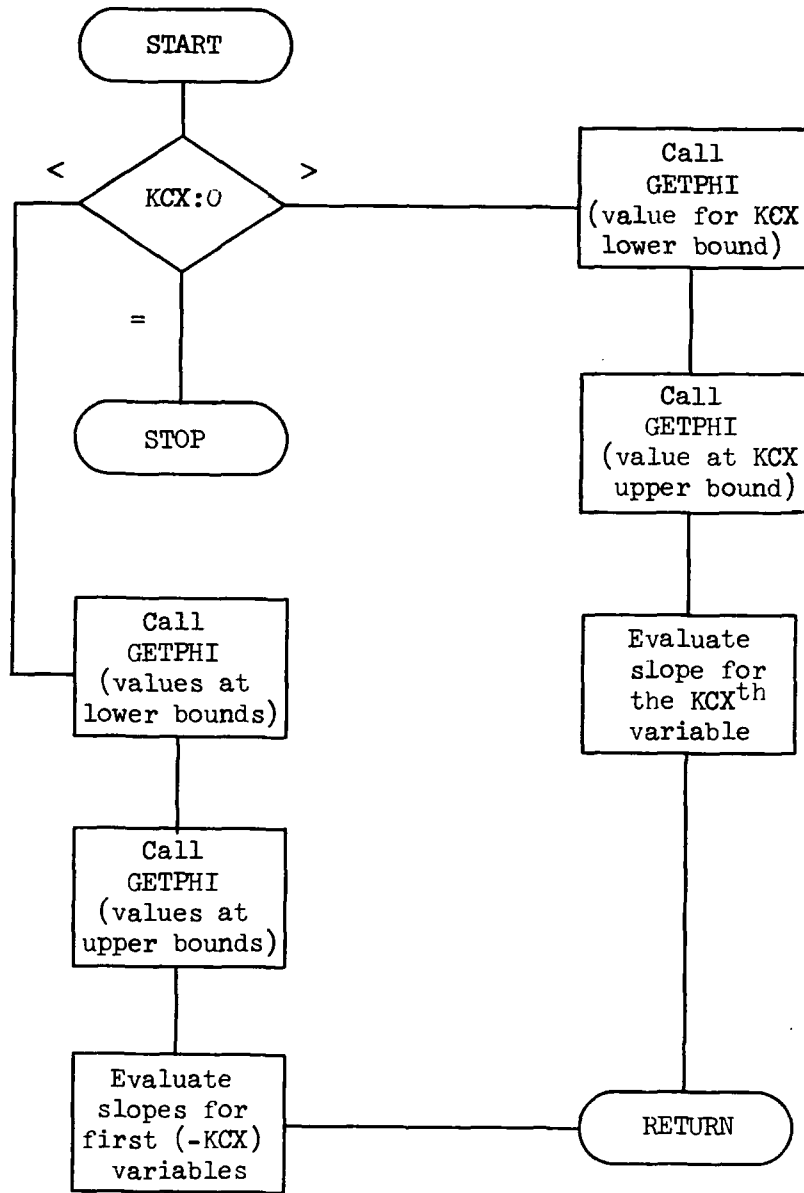
SUBROUTINE NXBRN



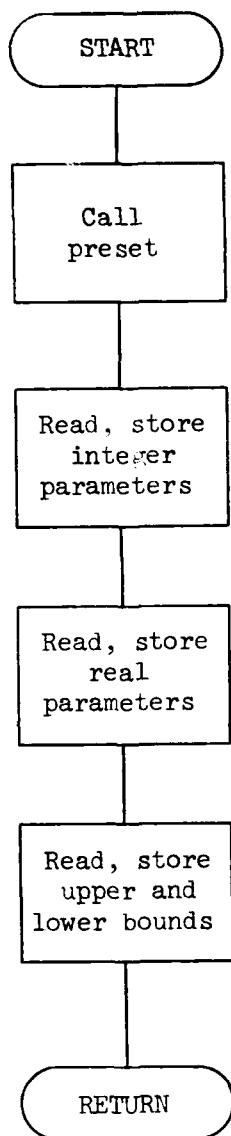
SUBROUTINE GETASQ



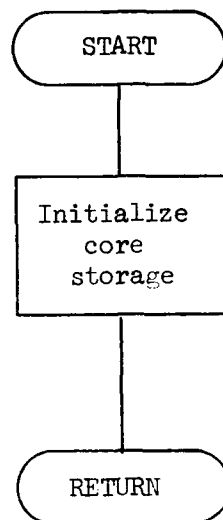
SUBROUTINE GETC



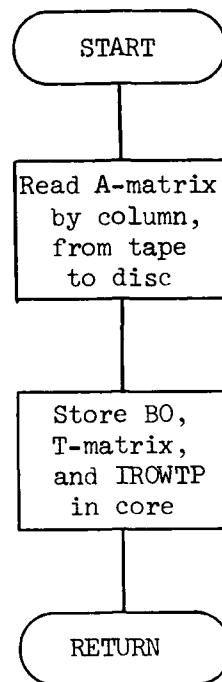
SUBROUTINE PARAMS

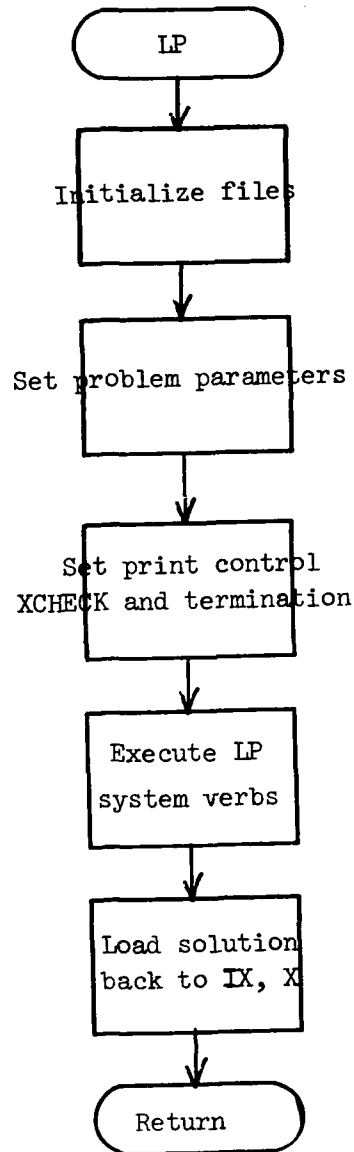


SUBROUTINE PRESET



SUBROUTINE INITA



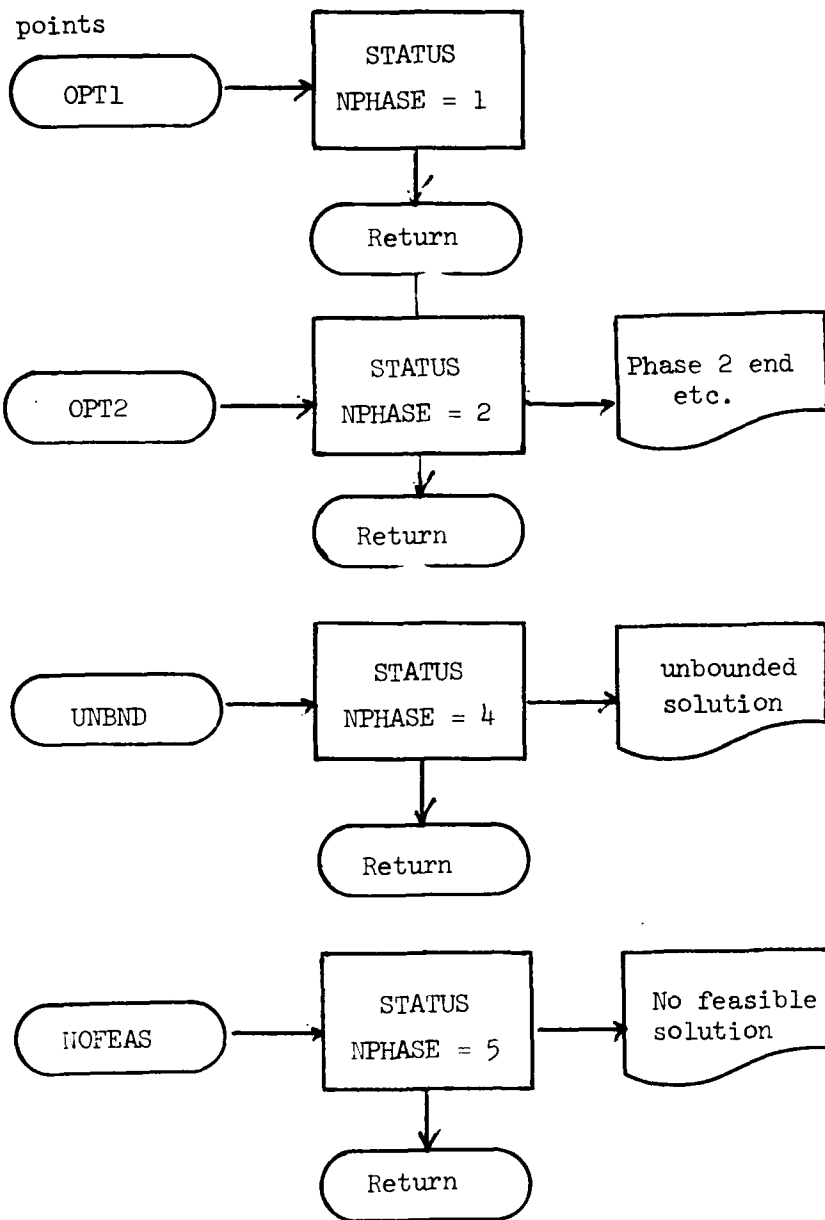


Subroutine EXITS

EXITS

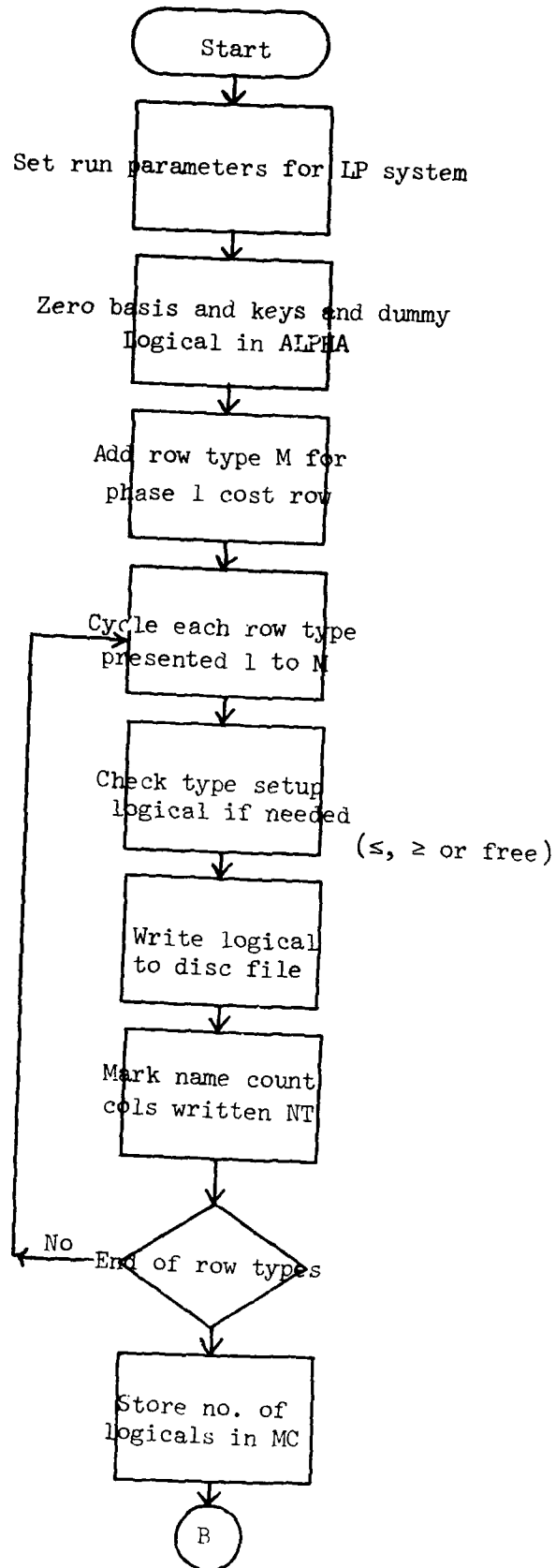
User changable EXITS program called after all control points in PRIMAL.

entry points

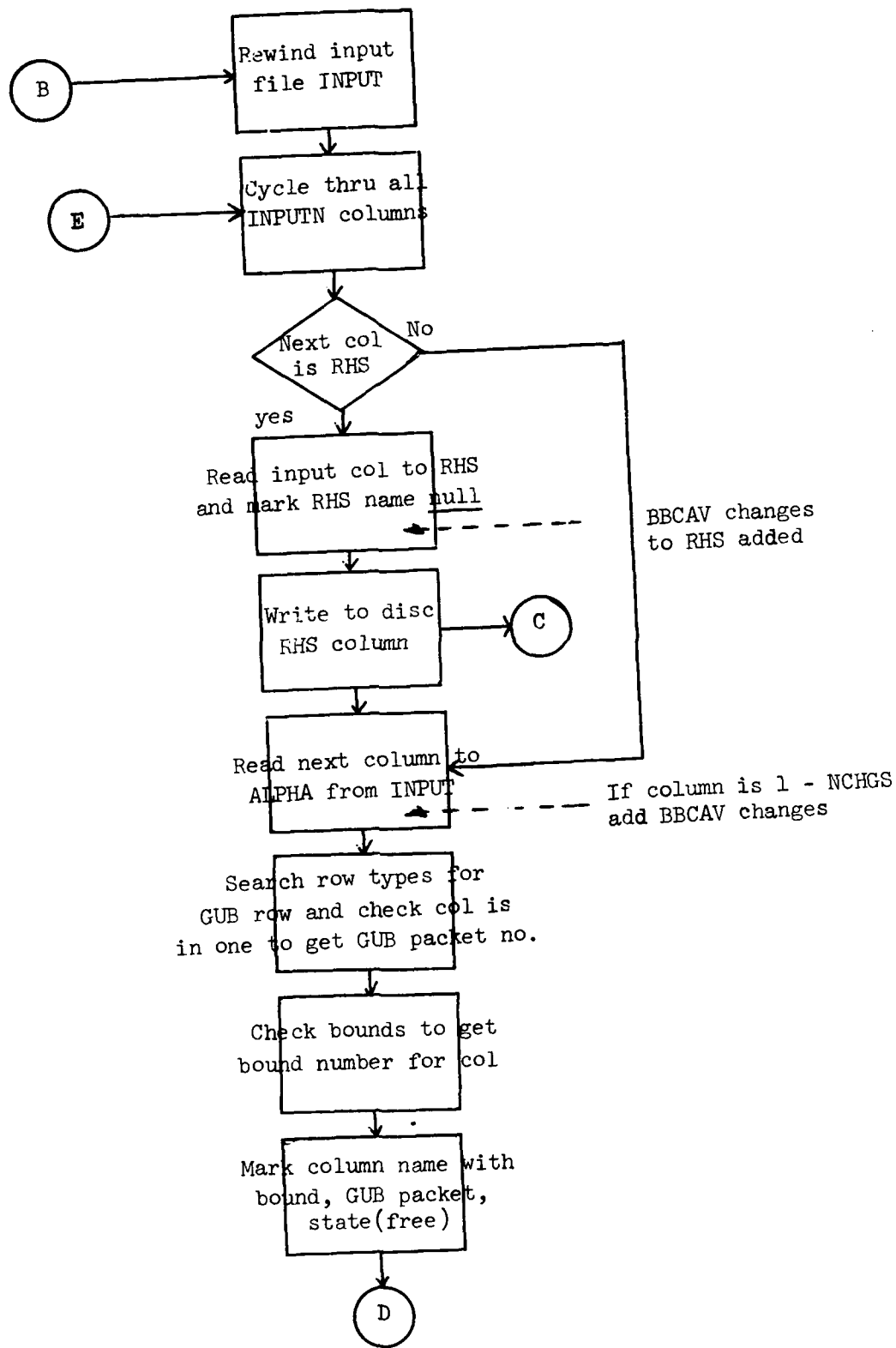


Subroutine SETUP

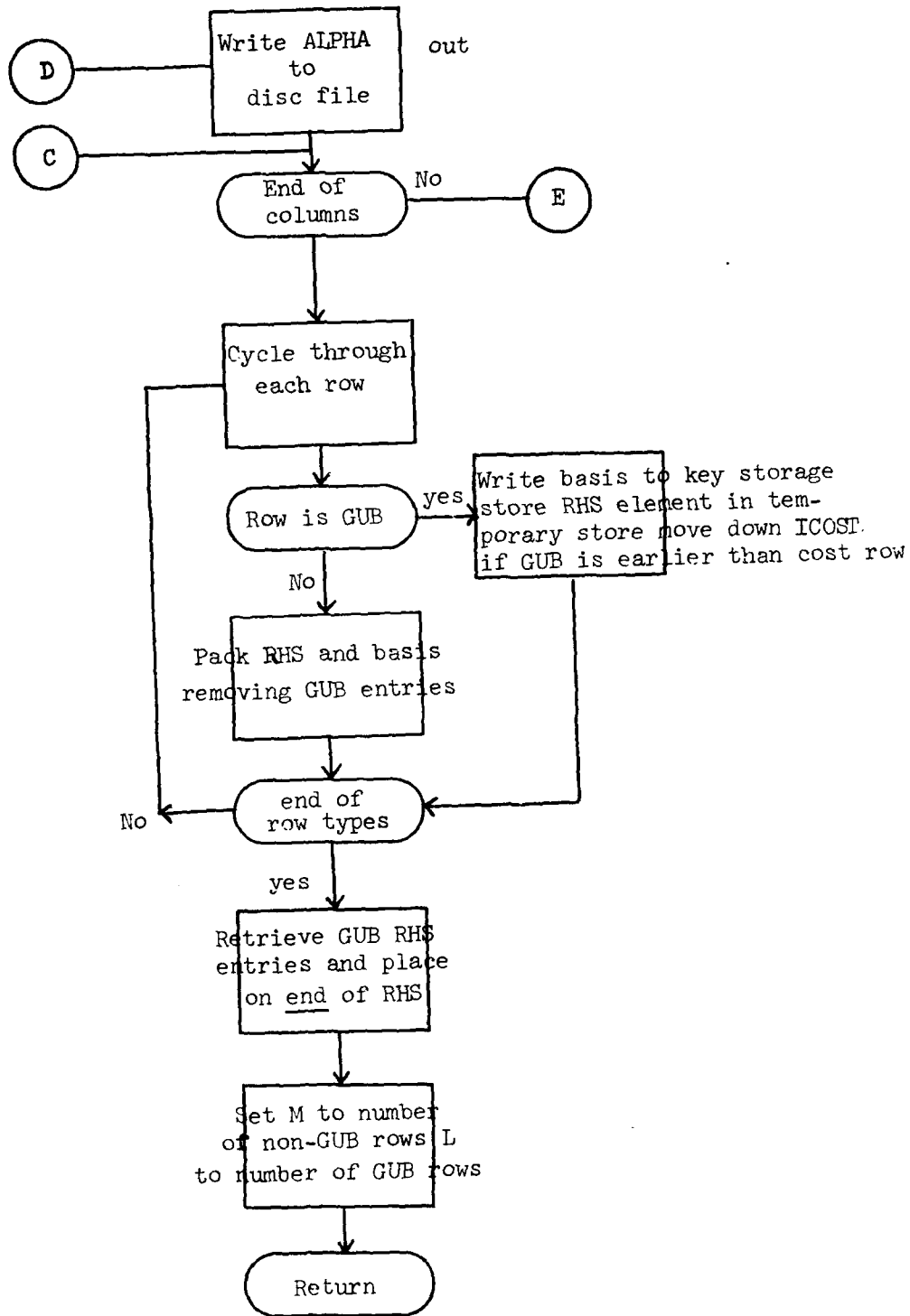
SETUP 1.



SETUP 2.

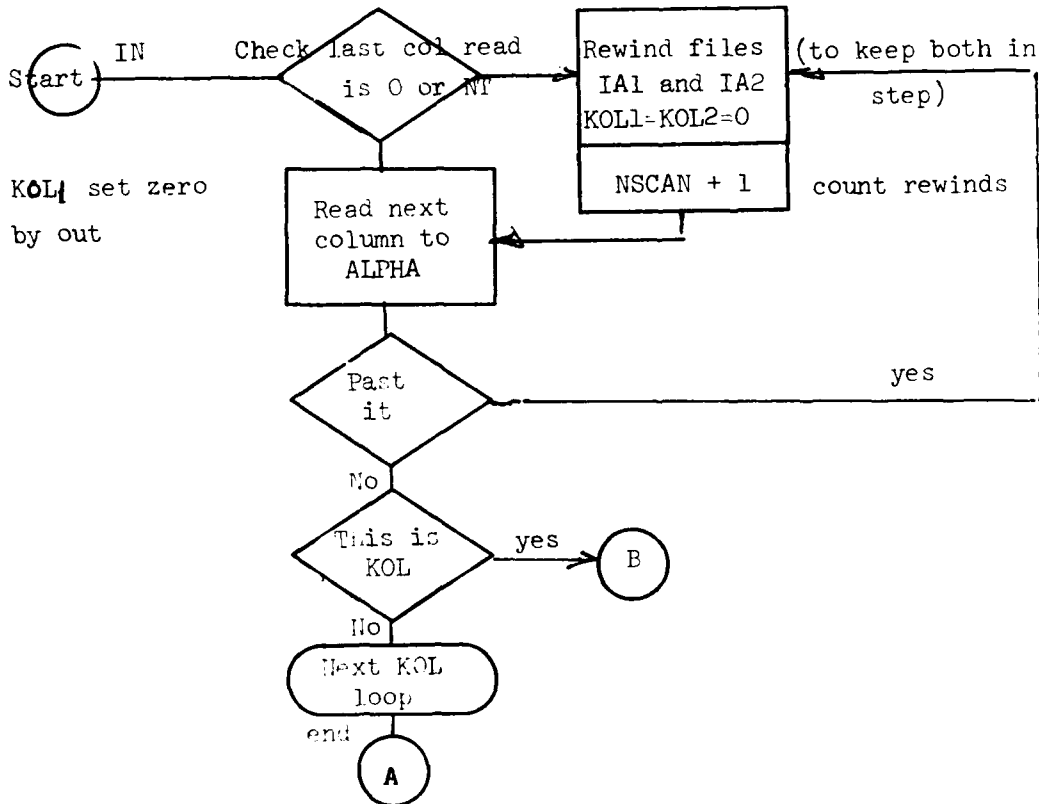
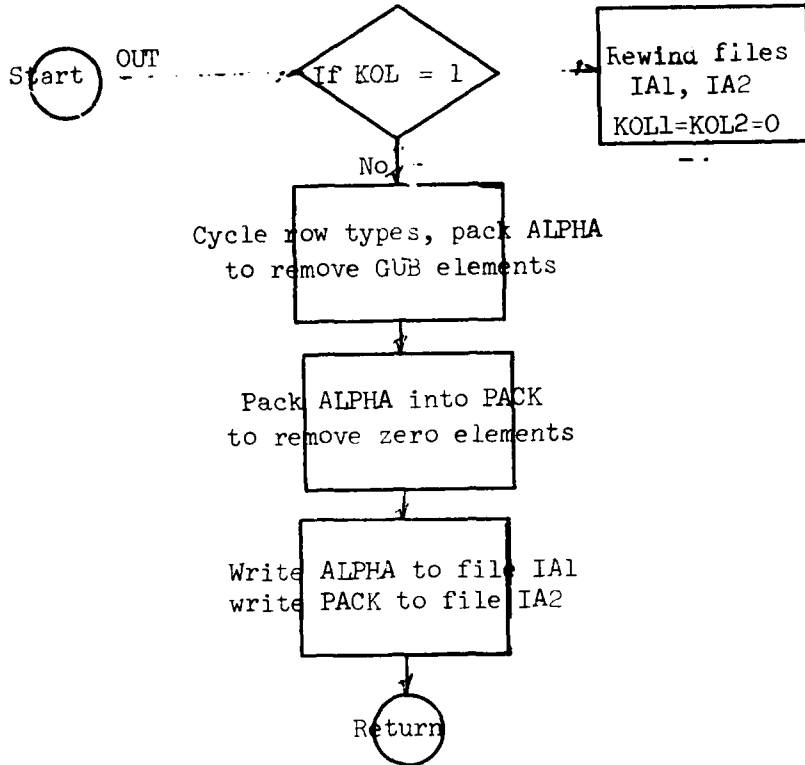


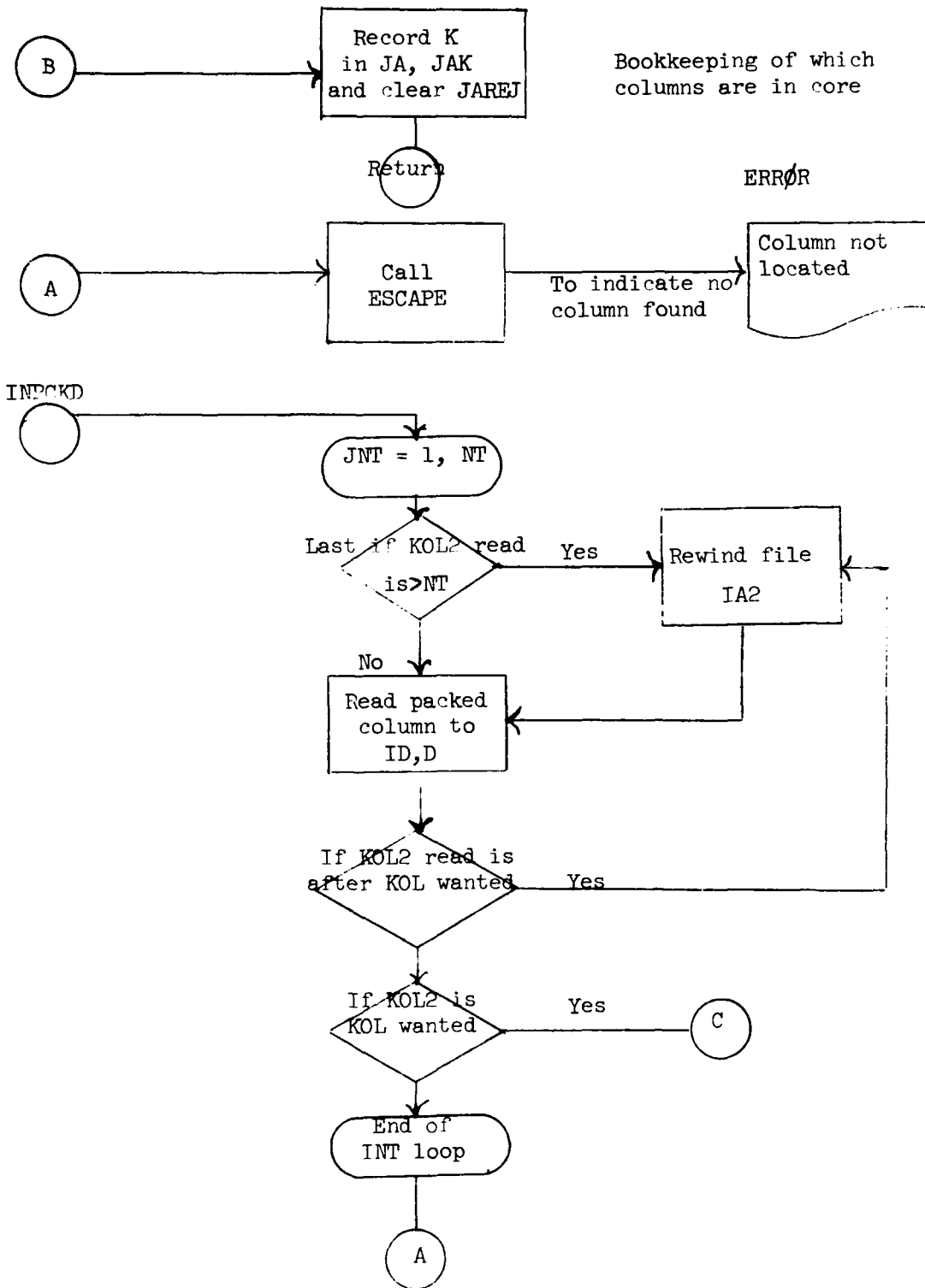
SETUP 3.

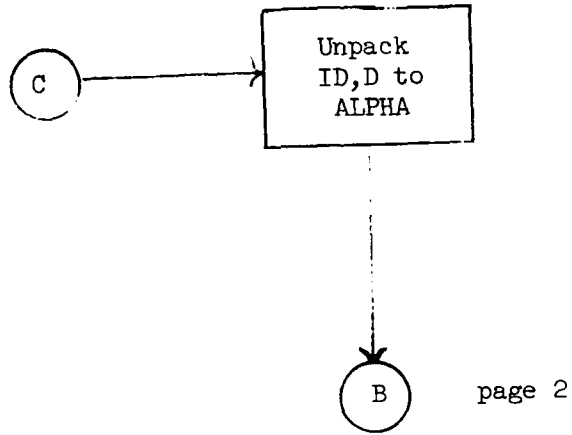


Subroutine IO

IO 1.



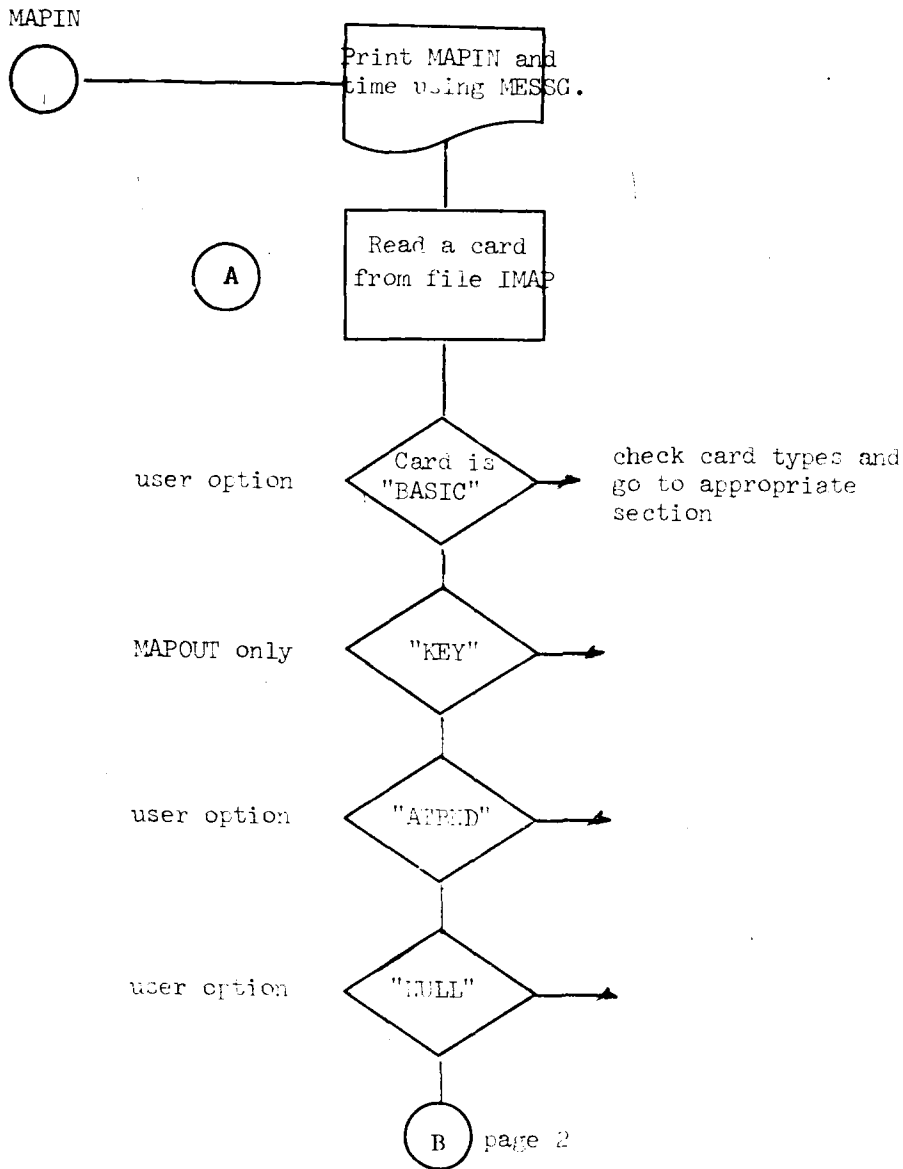


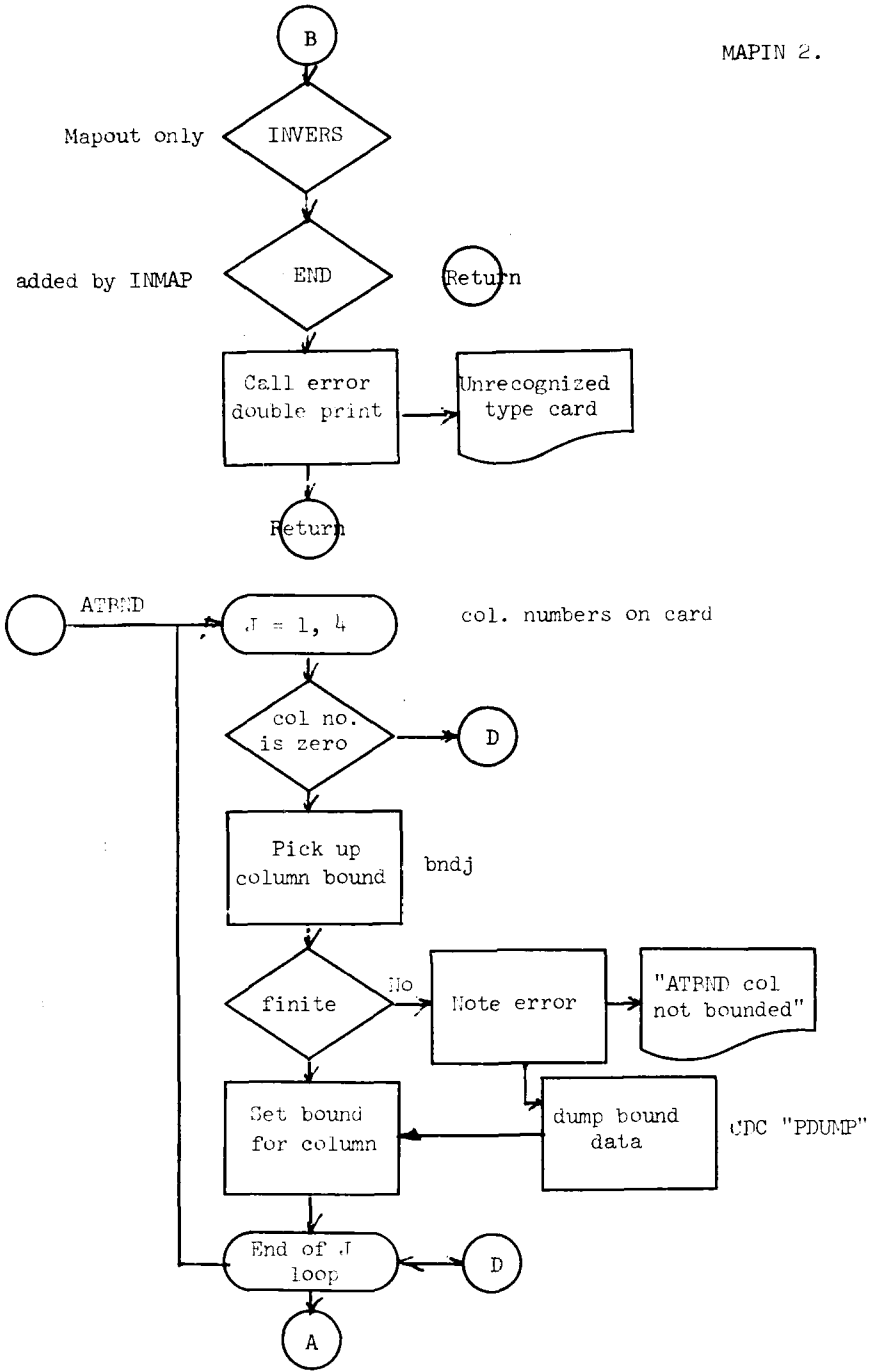


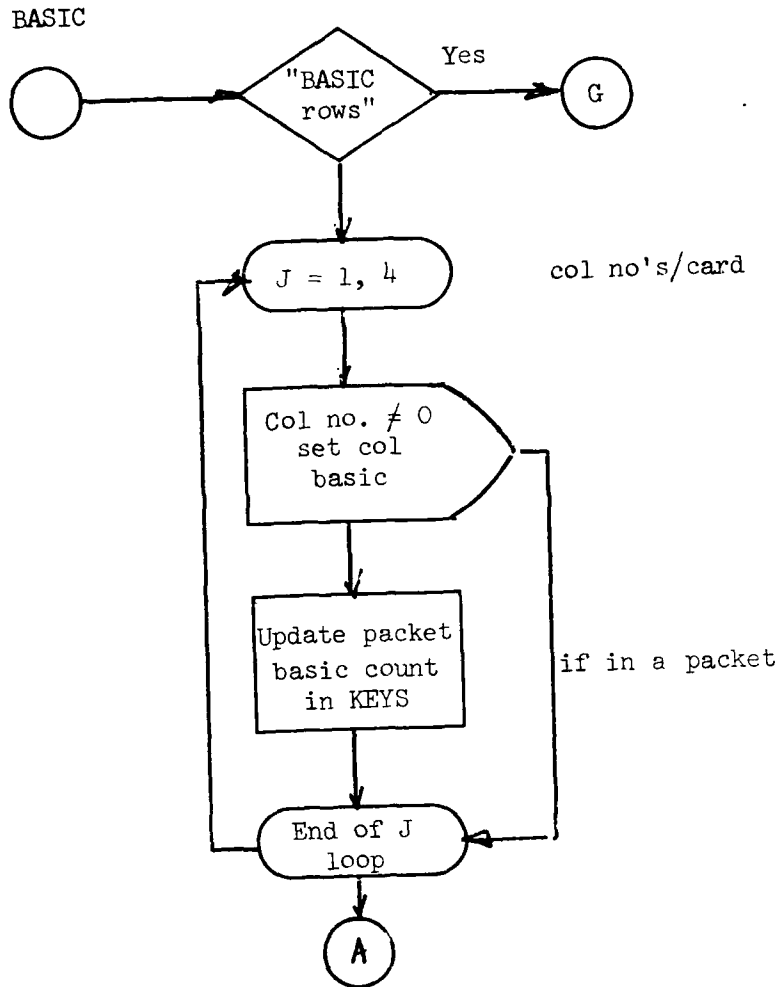
Subroutine MAPIN
Two entry points

MAPIN 1.

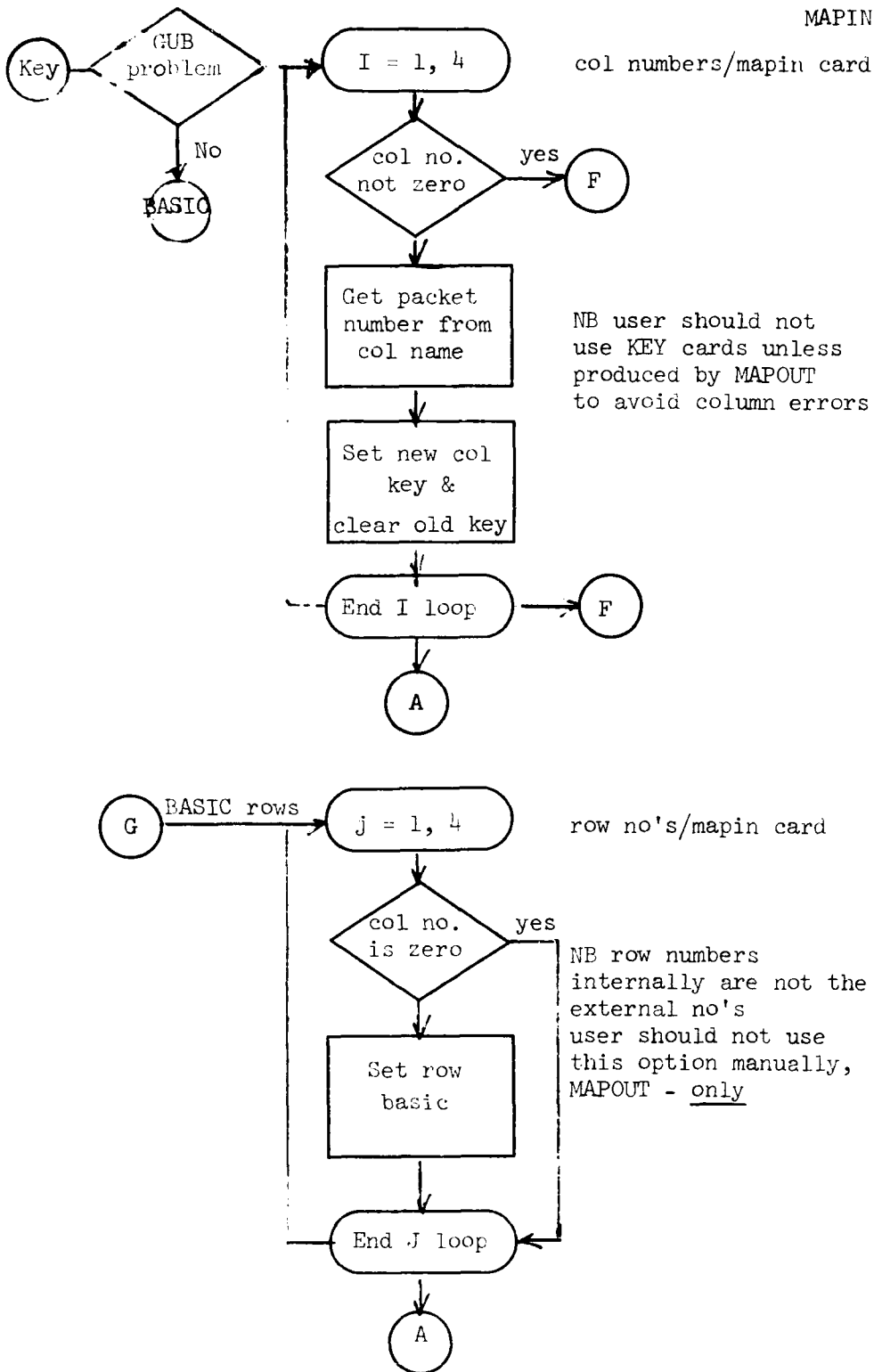
- (1) MAPIN - Reads MAPIN cards from file IMAI and sets NAME record
reads inverse from file INPUT to B.
- (2) INMAP - Loads file IMAI from input card stream.



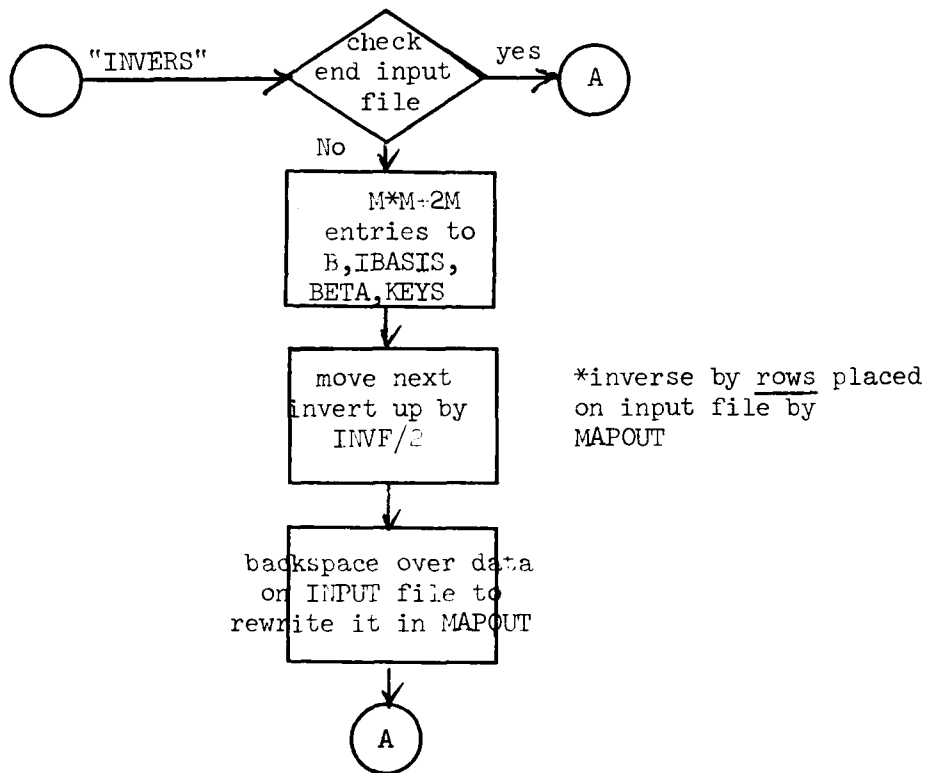
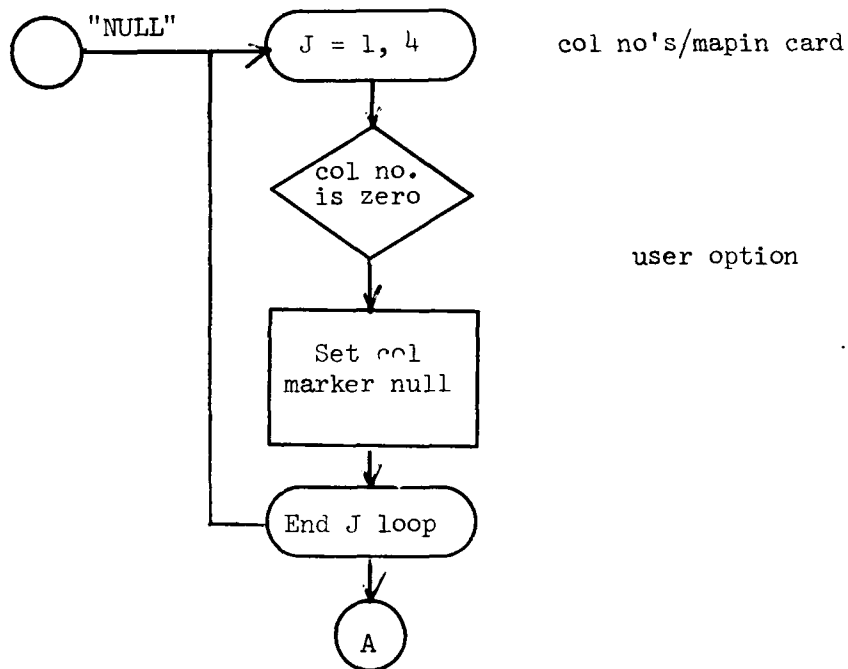




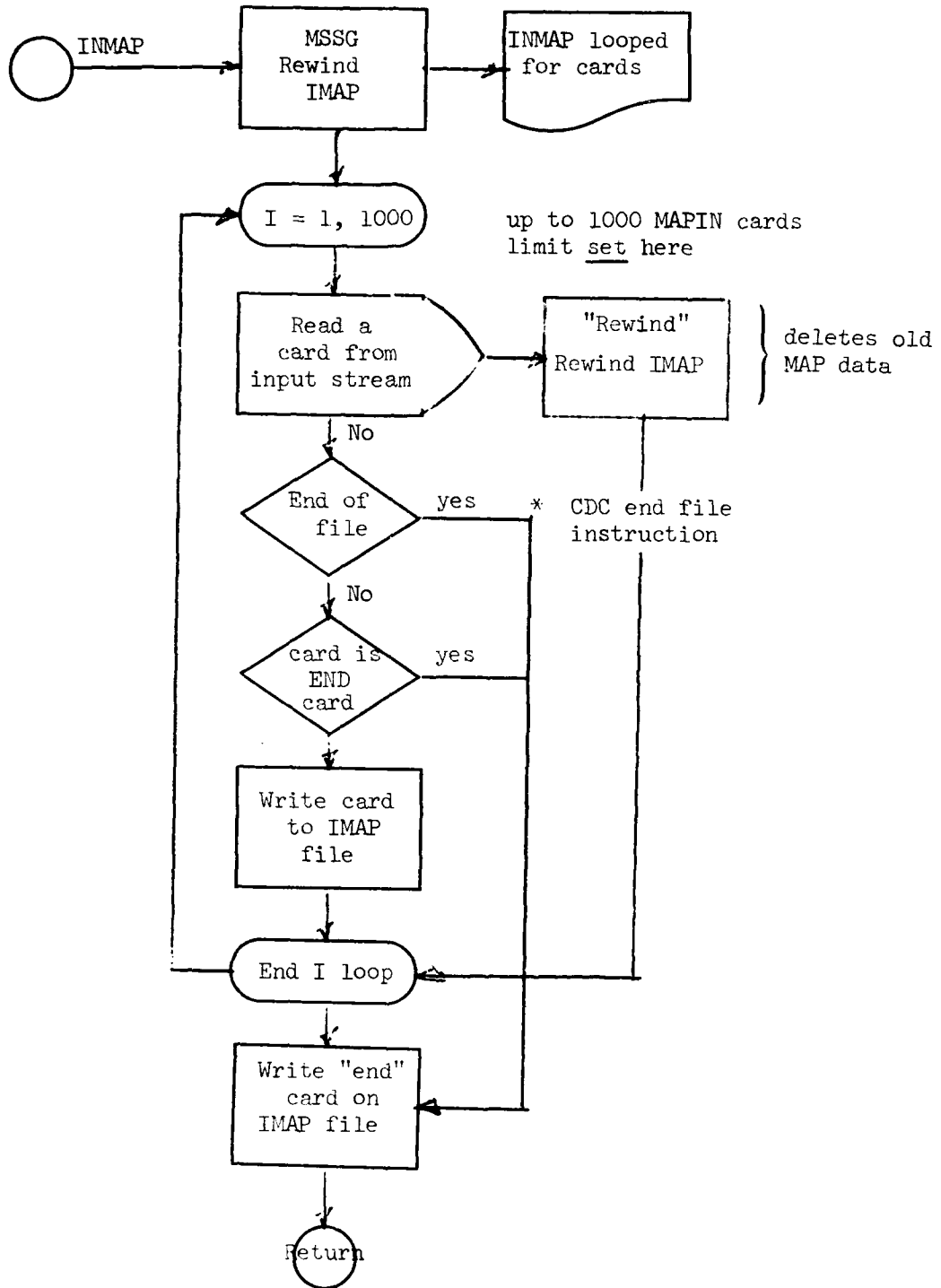
MAPIN 4.



MAPIN 5.



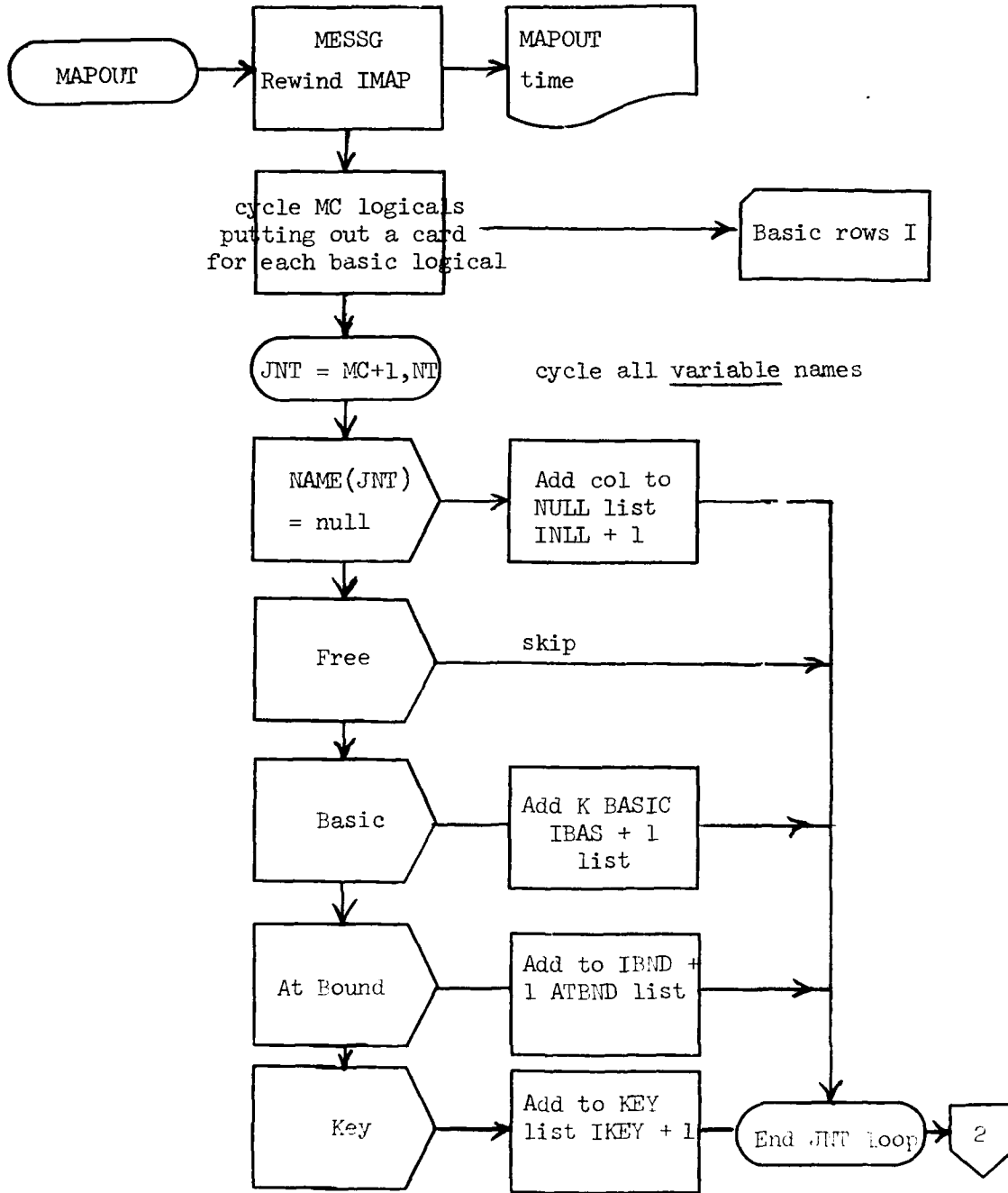
MAPIN 6.



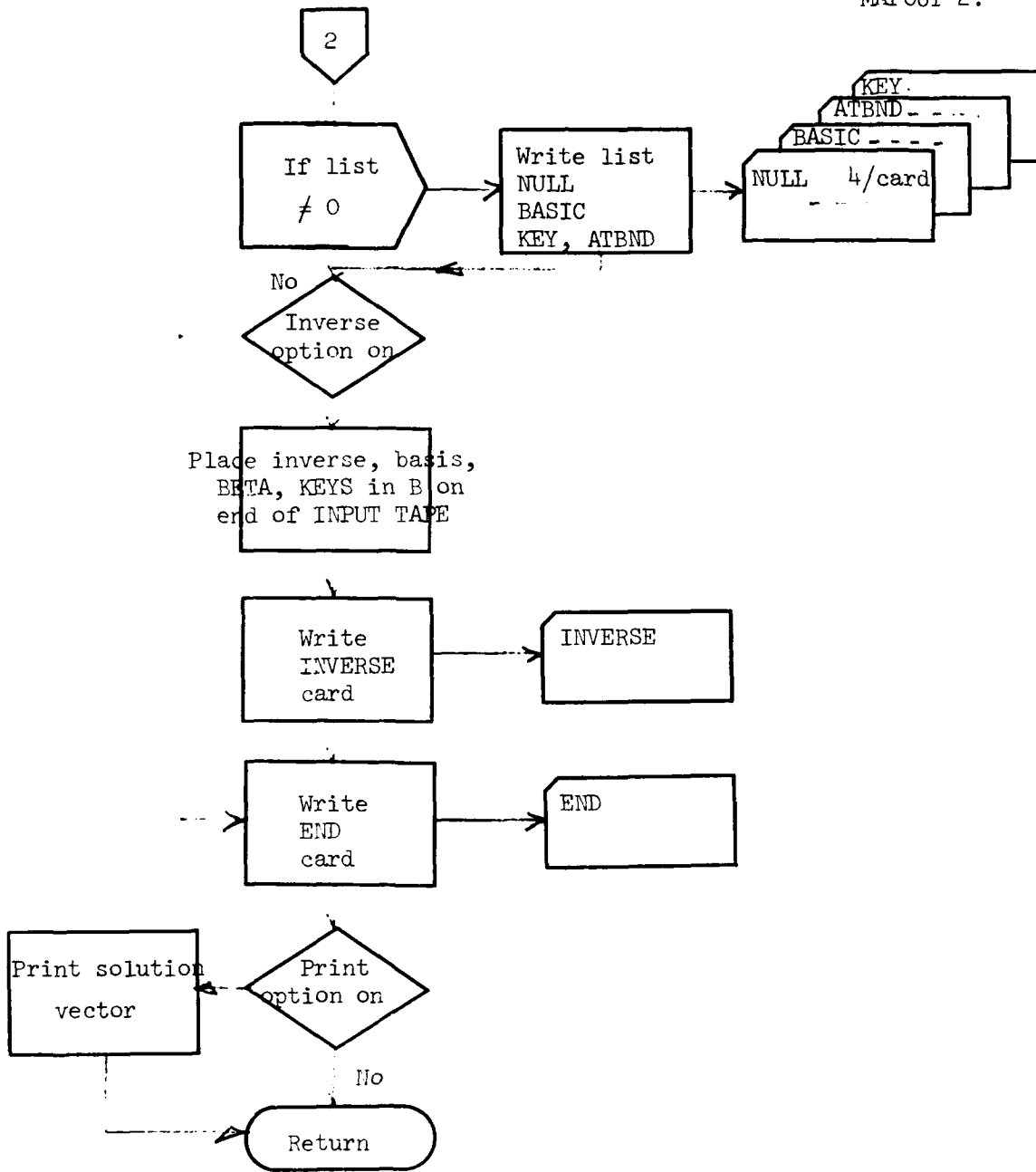
Subroutine MAPOUT

MAPOUT 1.

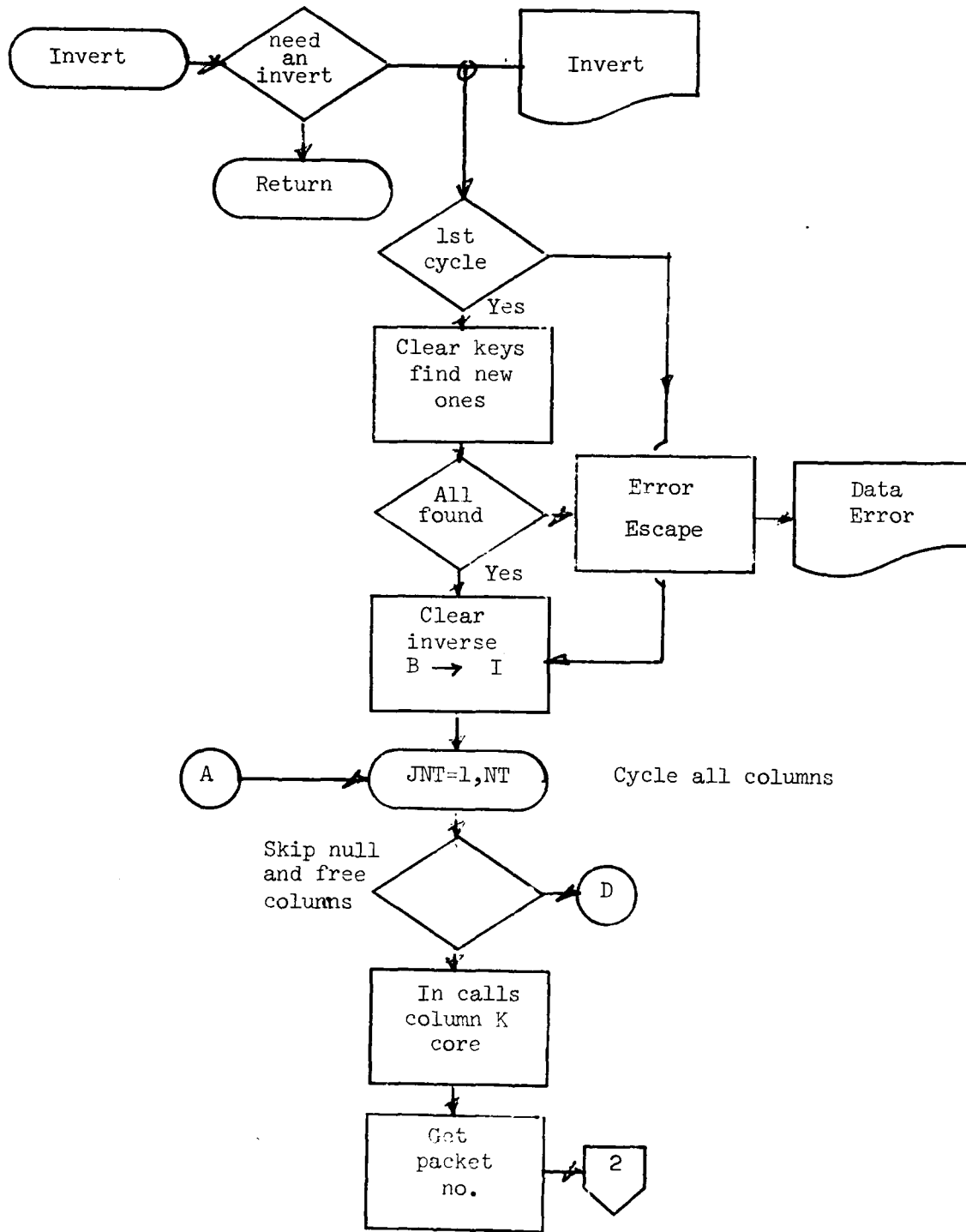
MAPOUT outputs on file IMAP a BCD card image definition of variables and inverse states compatible with MAPIN, whenever called and prints the current solution.

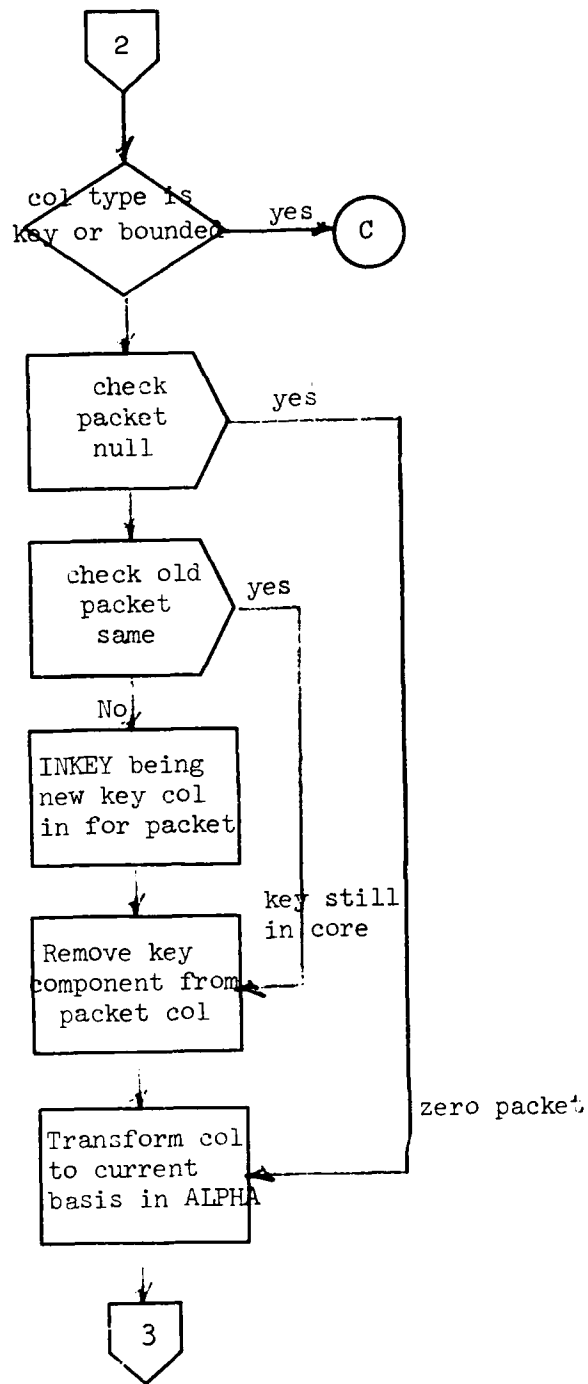


MAPOUT 2.

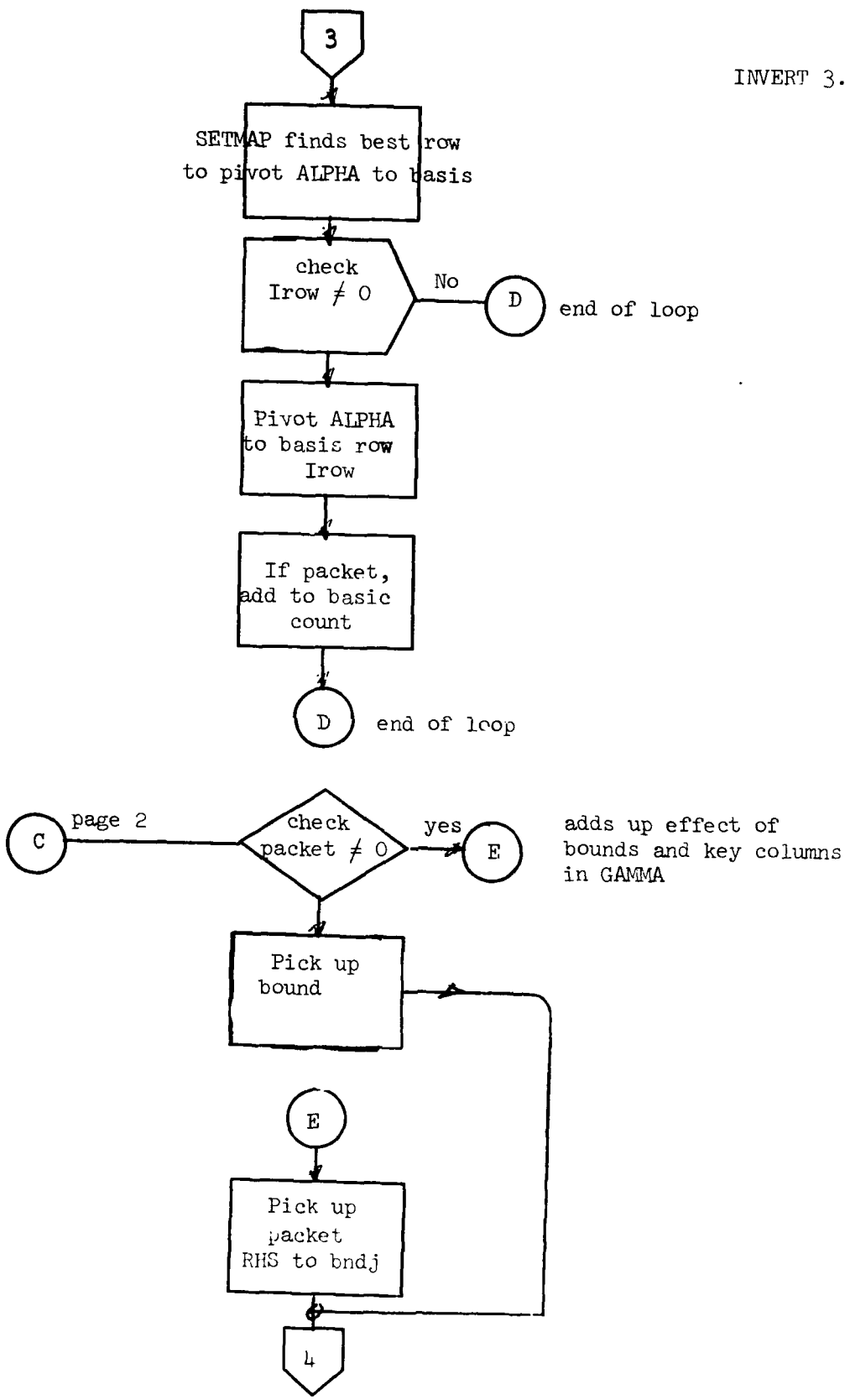


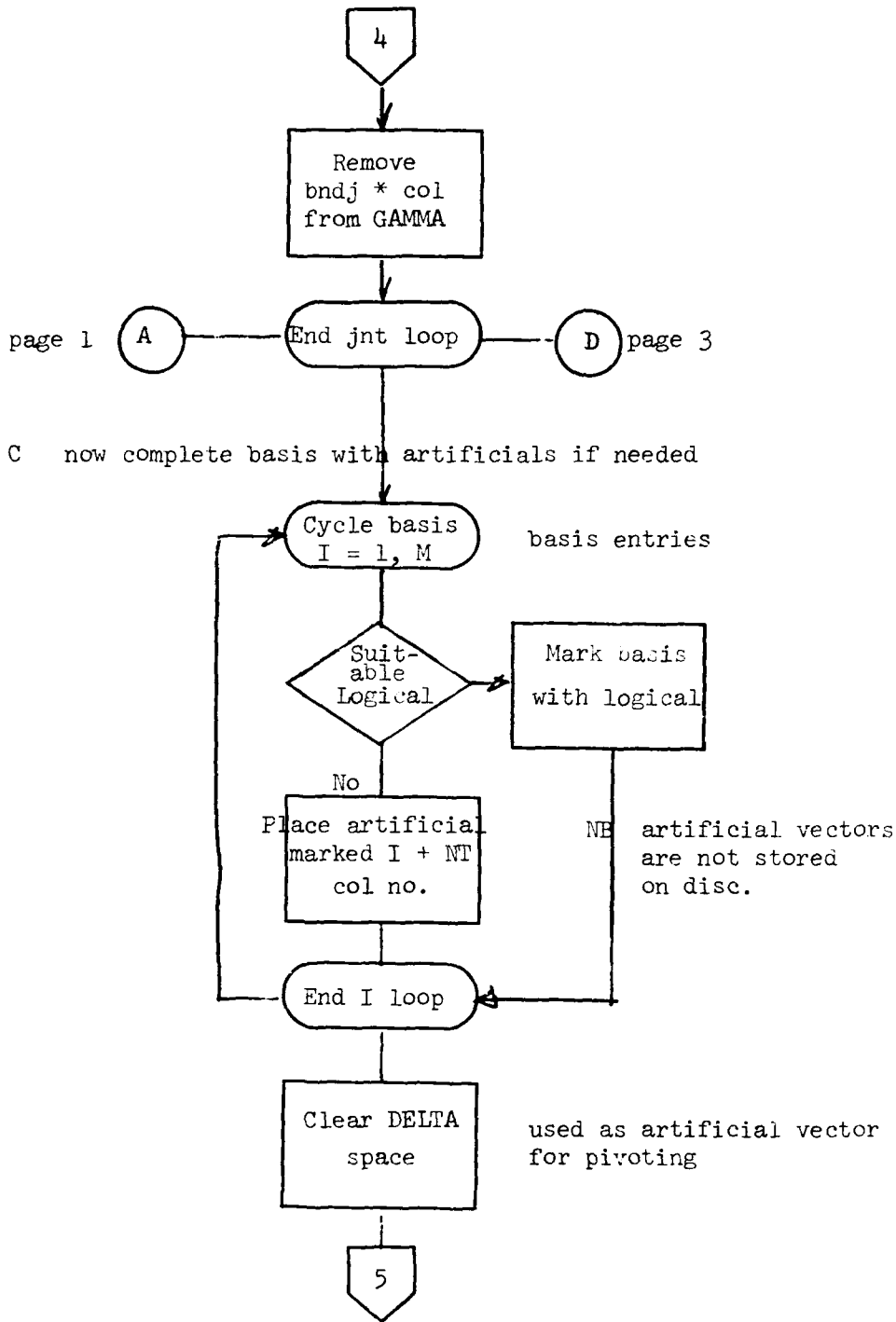
N.B. All MAPOUT cards can be generated manually. If they are inconsistent MAPIN uses the last setting of any column.



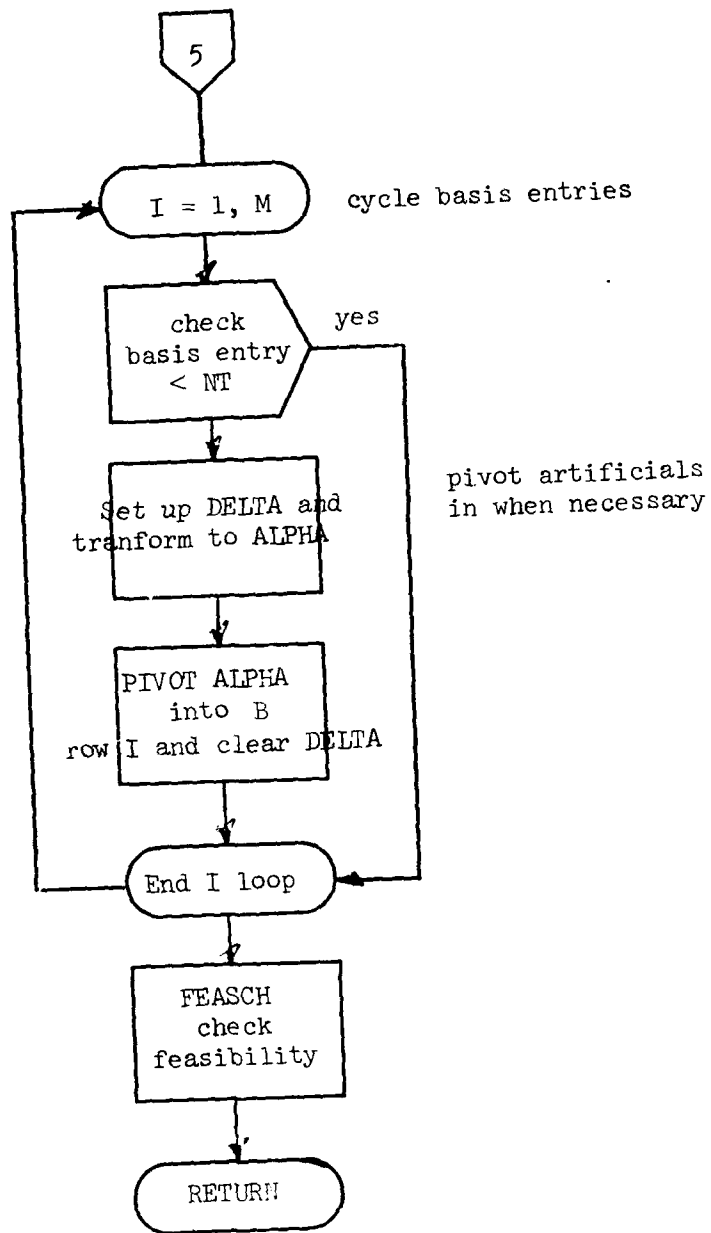


INVERT 3.





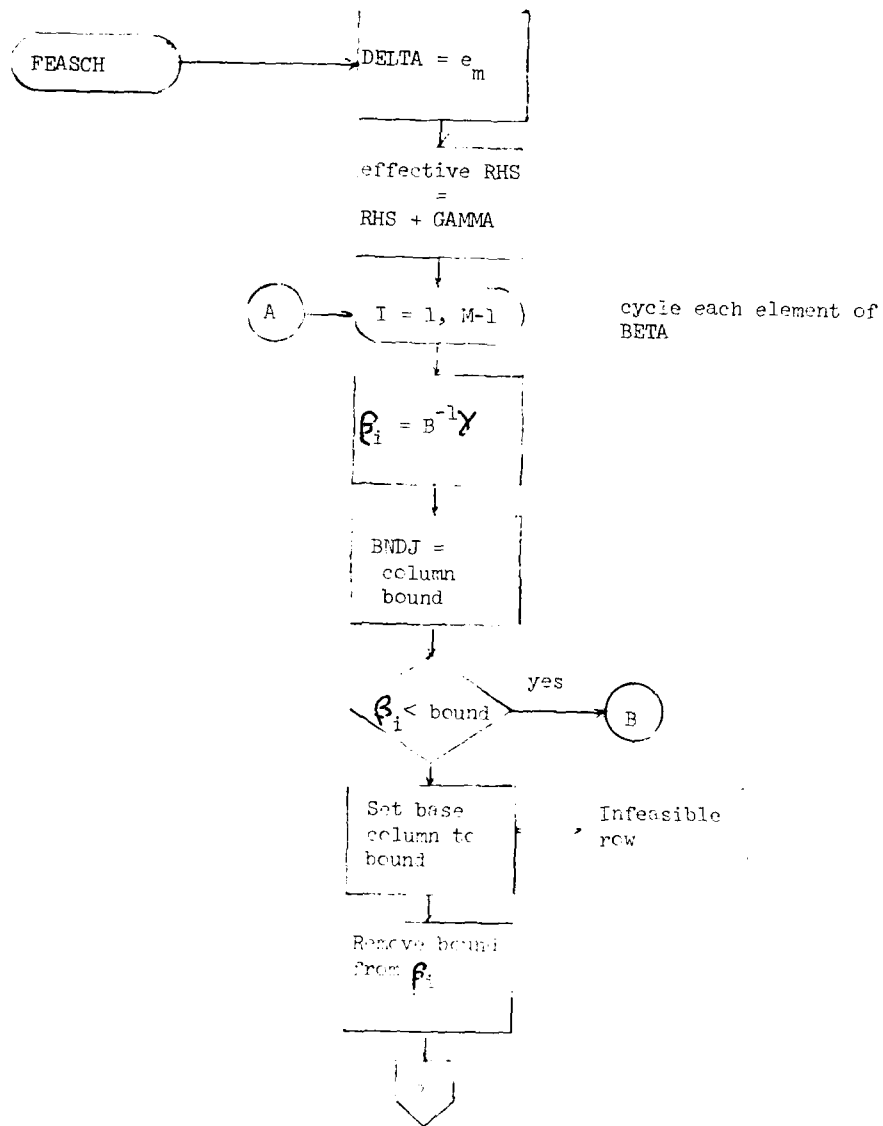
INVERT 5.

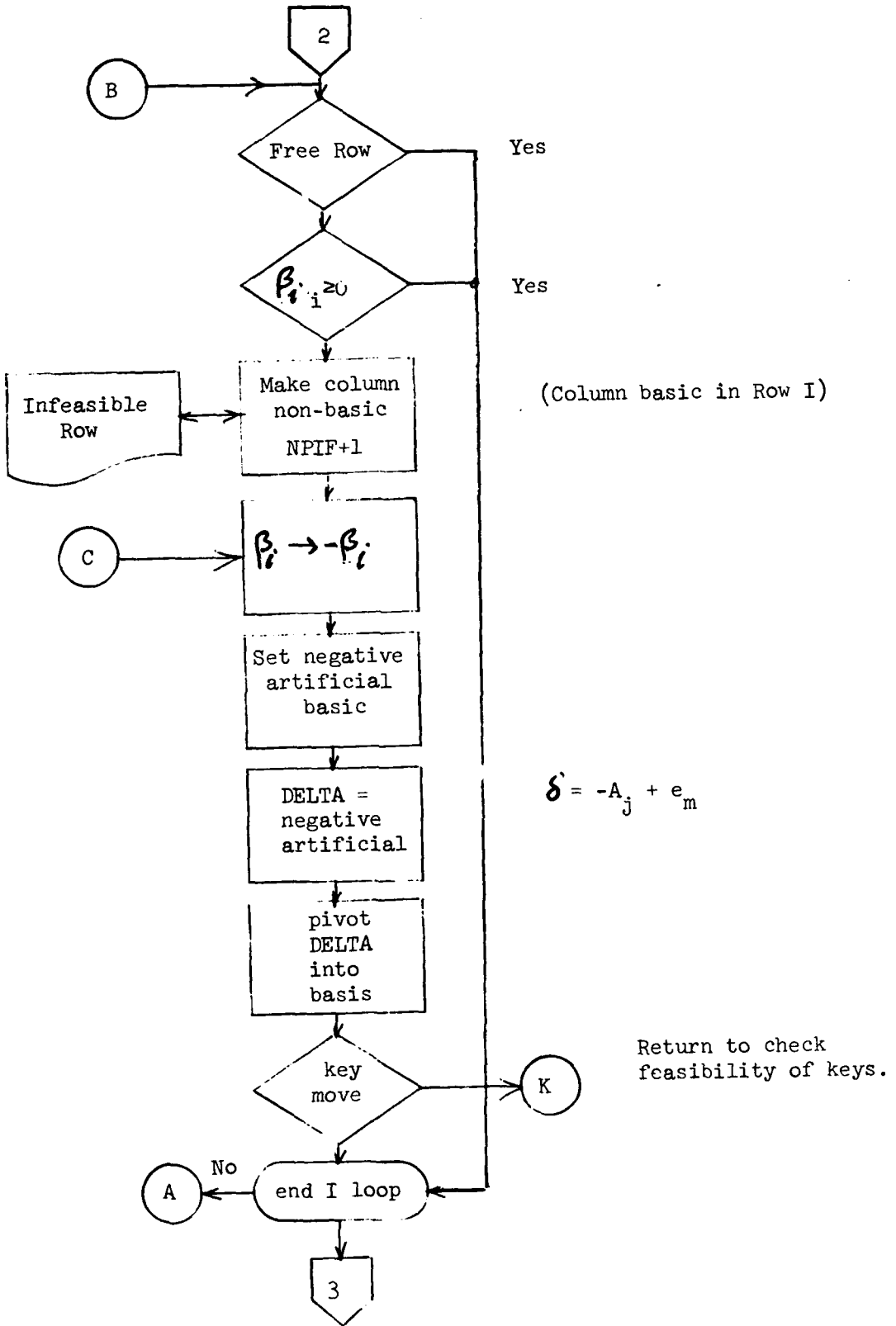


Subroutine FEASCH

FEASCH 1

This routine computes the solution and checks feasibility.

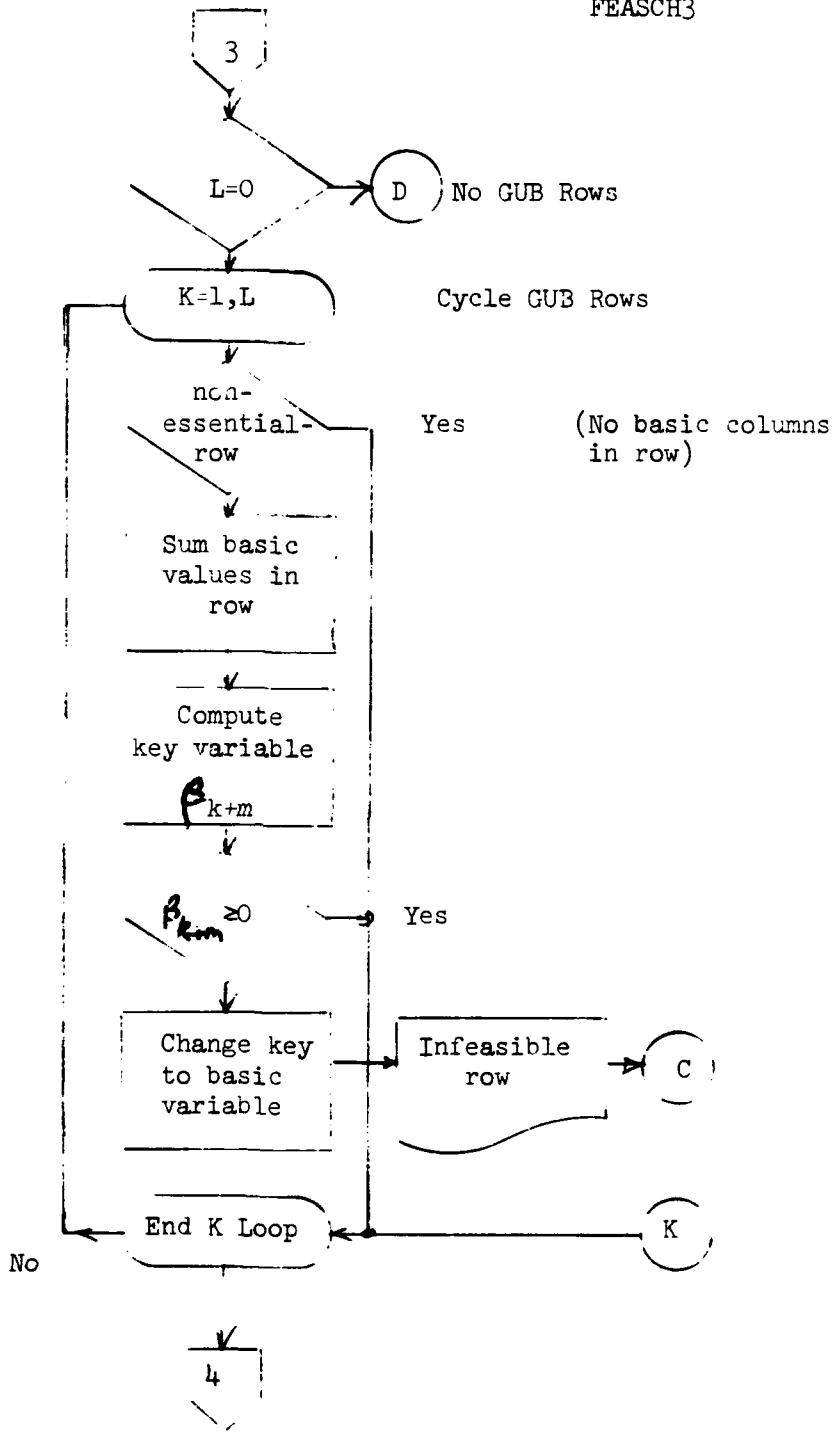


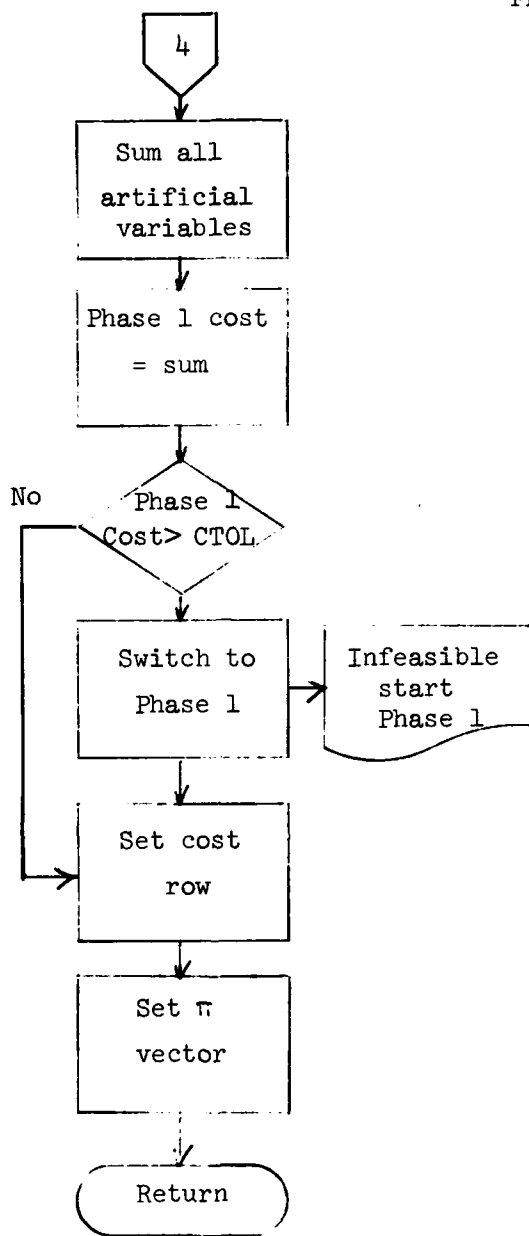


(Column basic in Row I)

$$\delta' = -A_j + e_m$$

Return to check feasibility of keys.

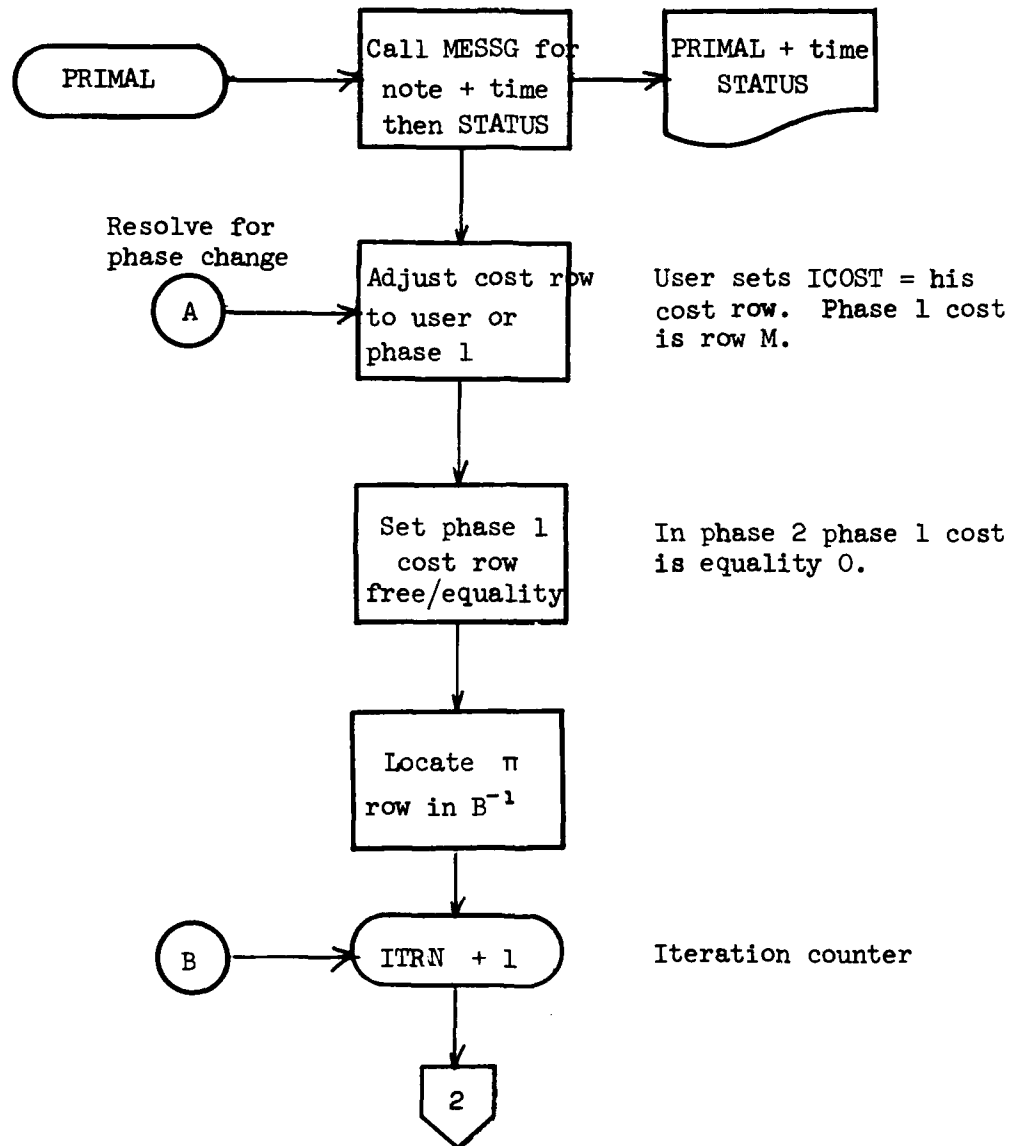




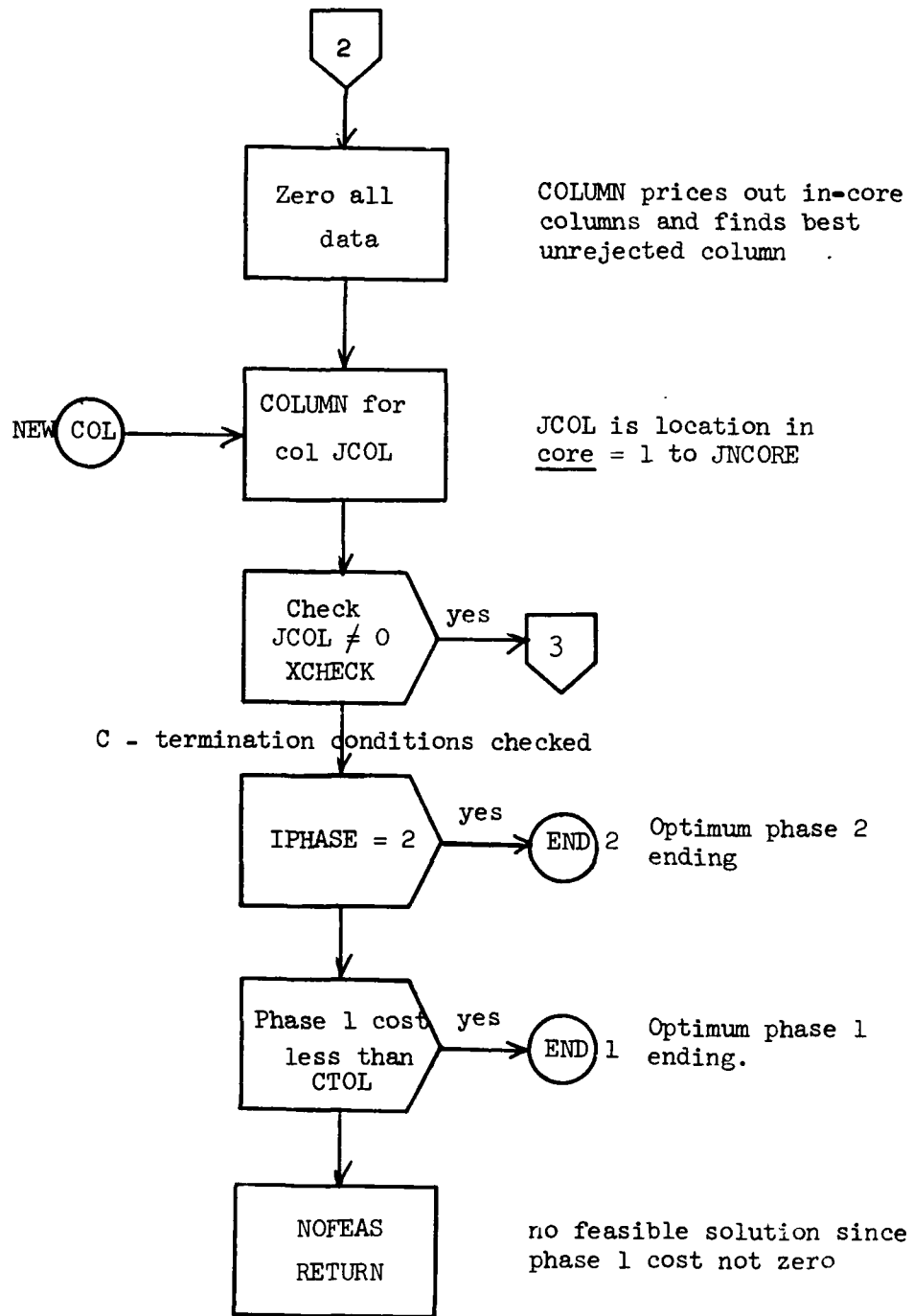
Subroutine PRIMAL

PRIMAL 1.

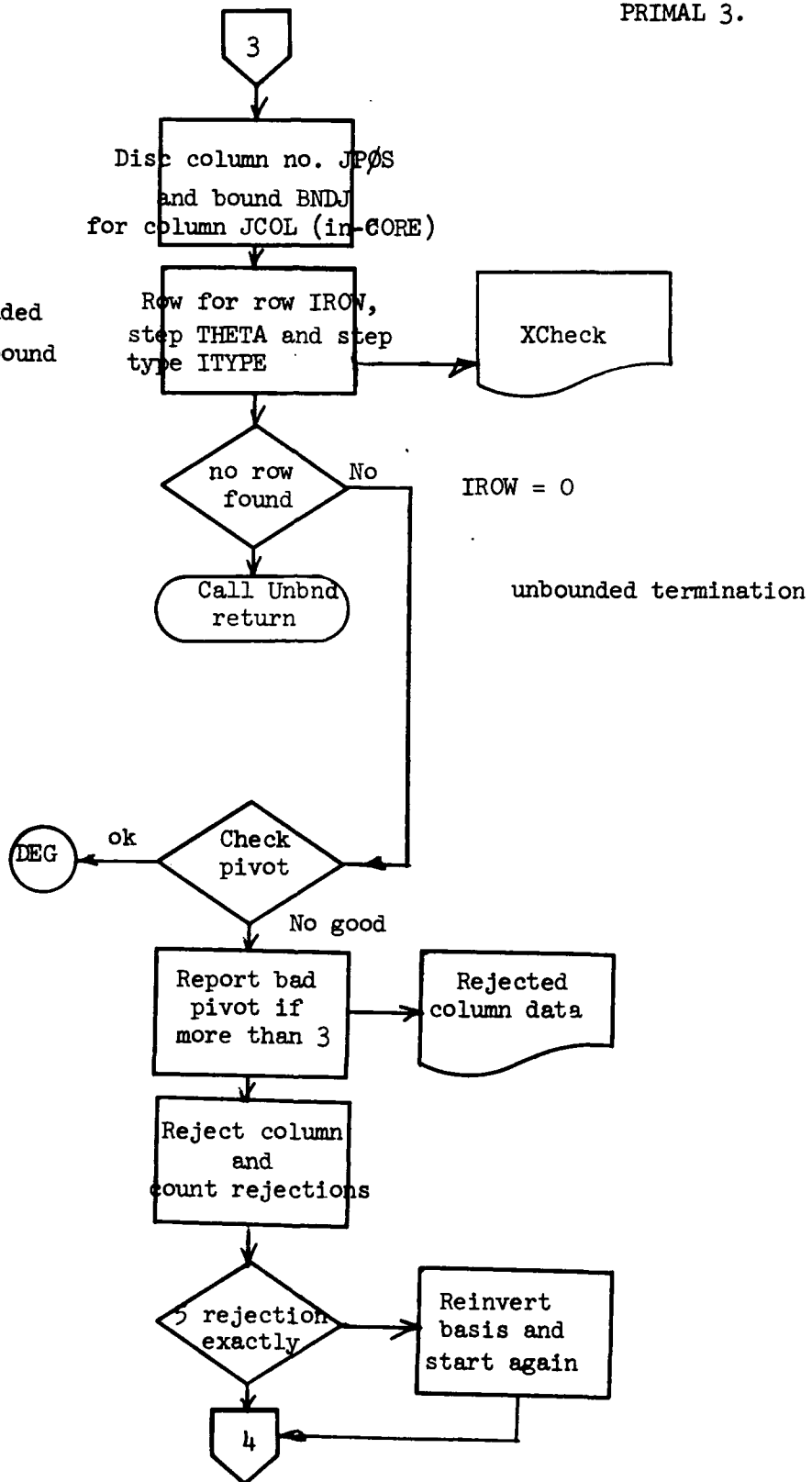
PRIMAL runs the 2 phase revised simplex algorithm from both phases and exits via EXIT.

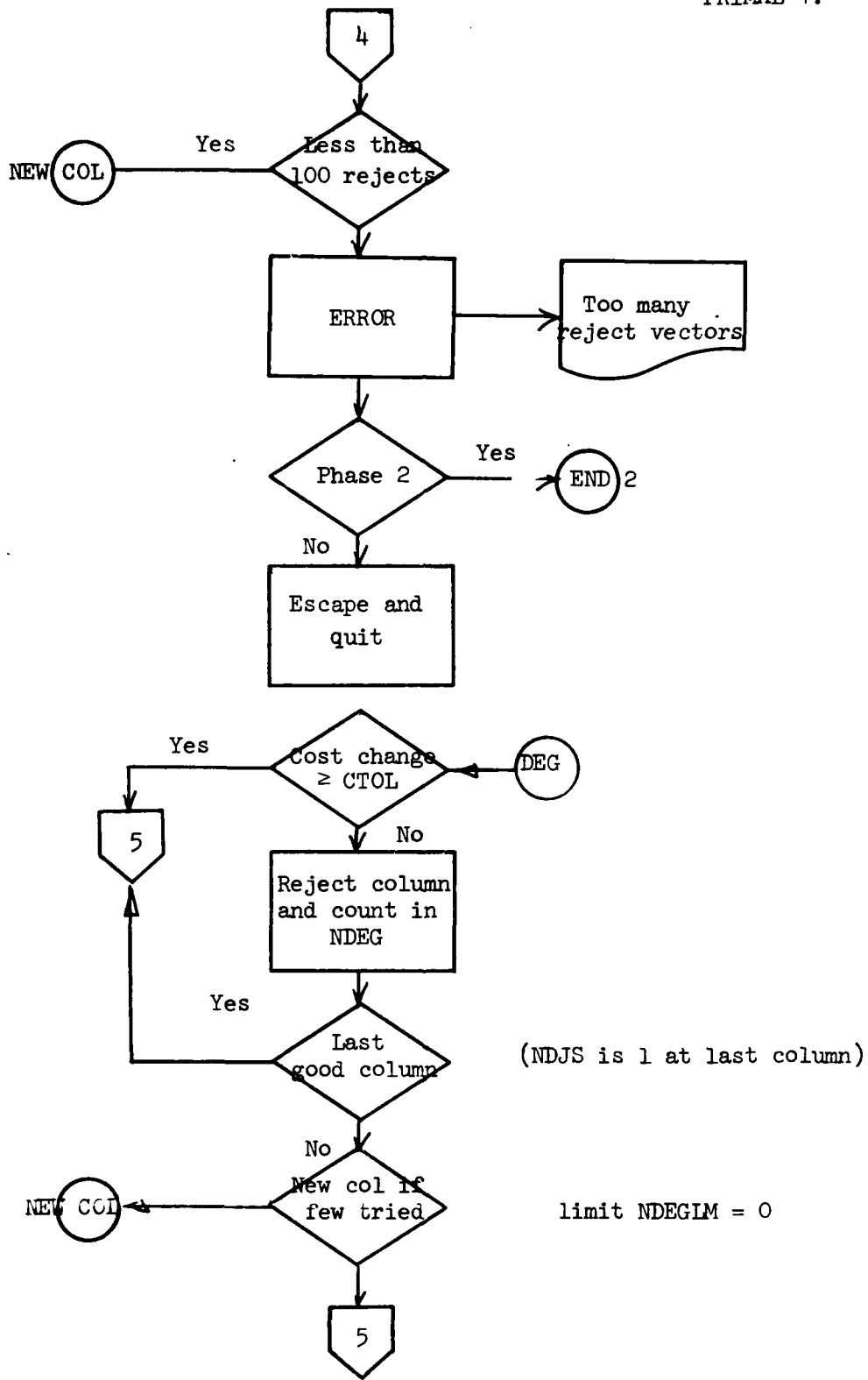


PRIMAL 2.

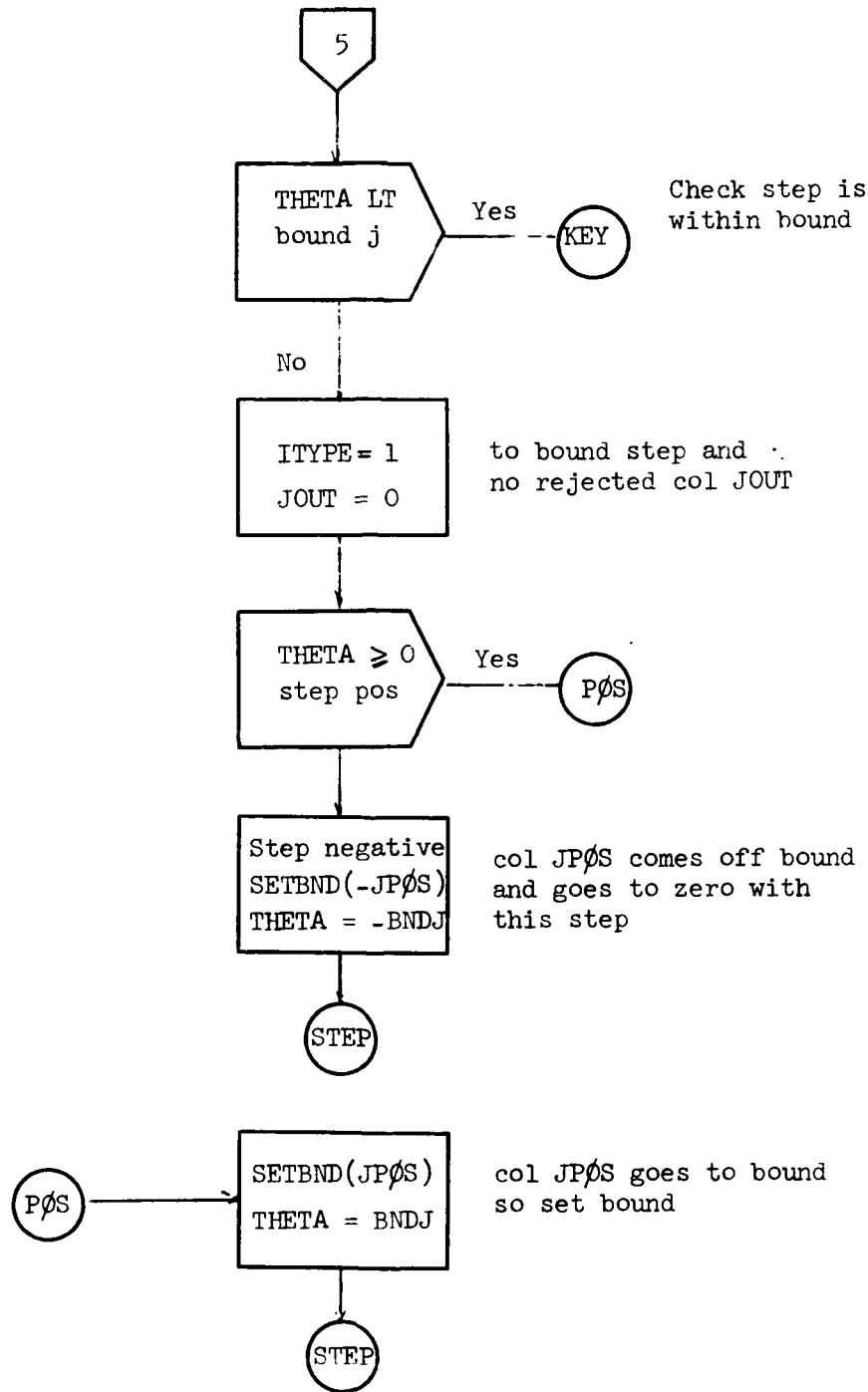


ITYPE = 1 unbounded
= 2 to a bound

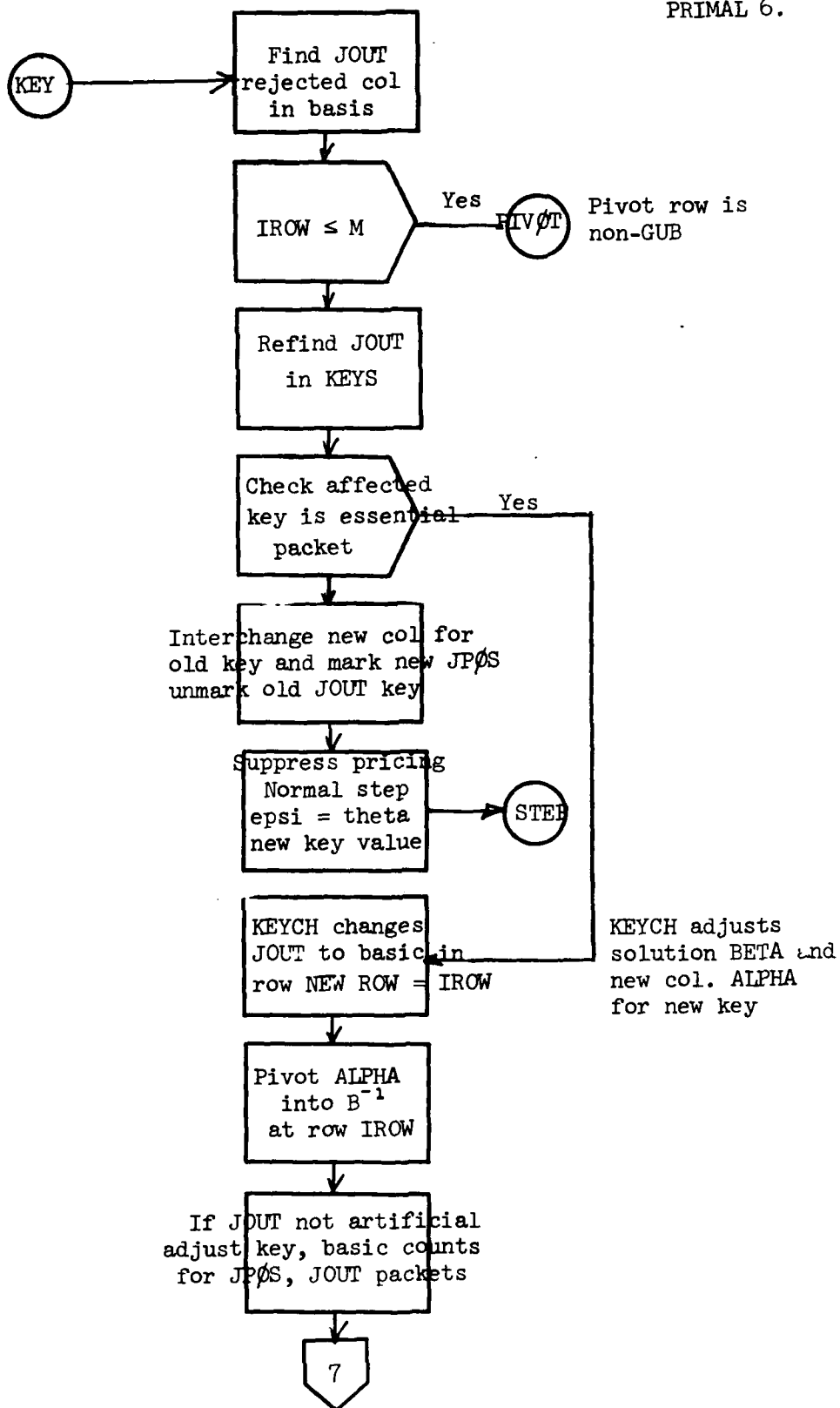


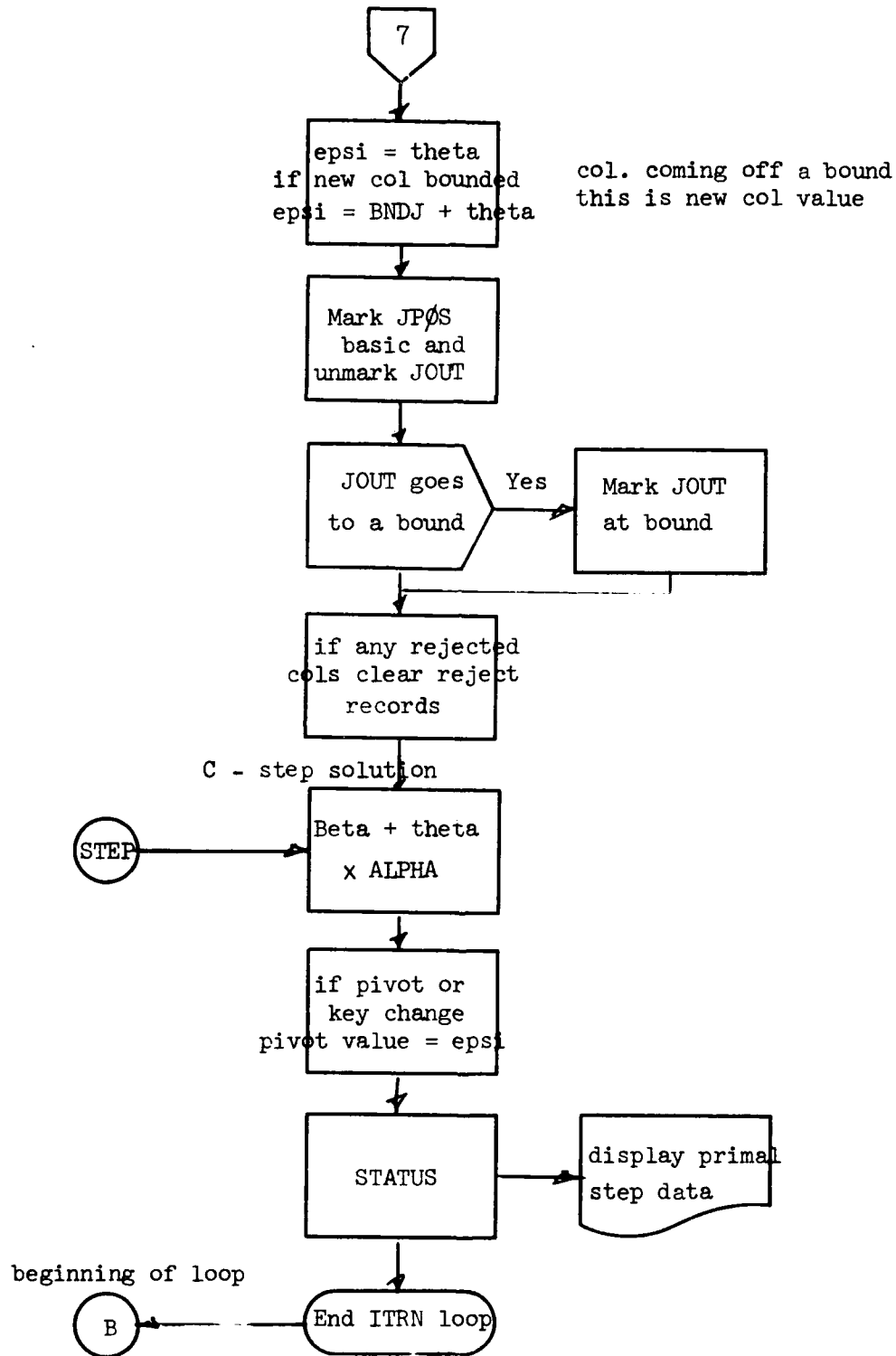


PRIMAL 5.

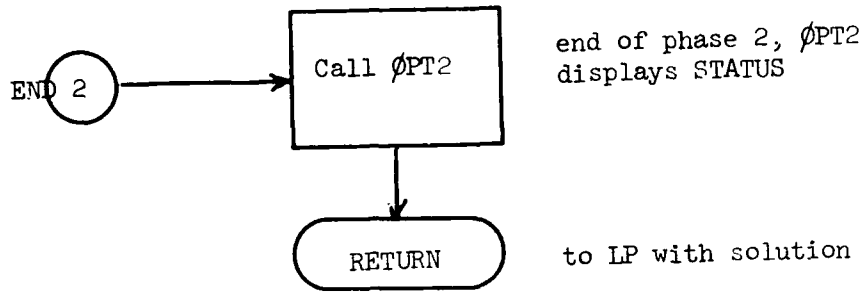
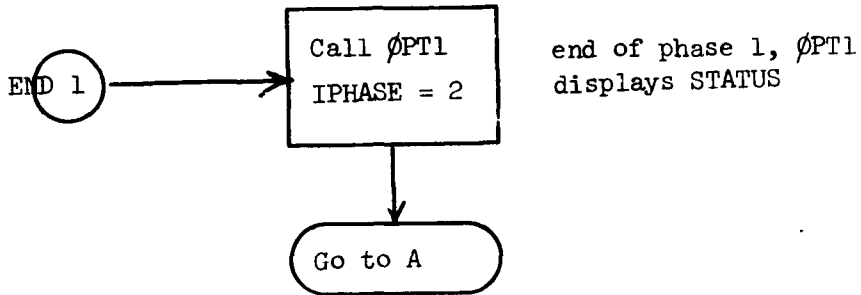


PRIMAL 6.





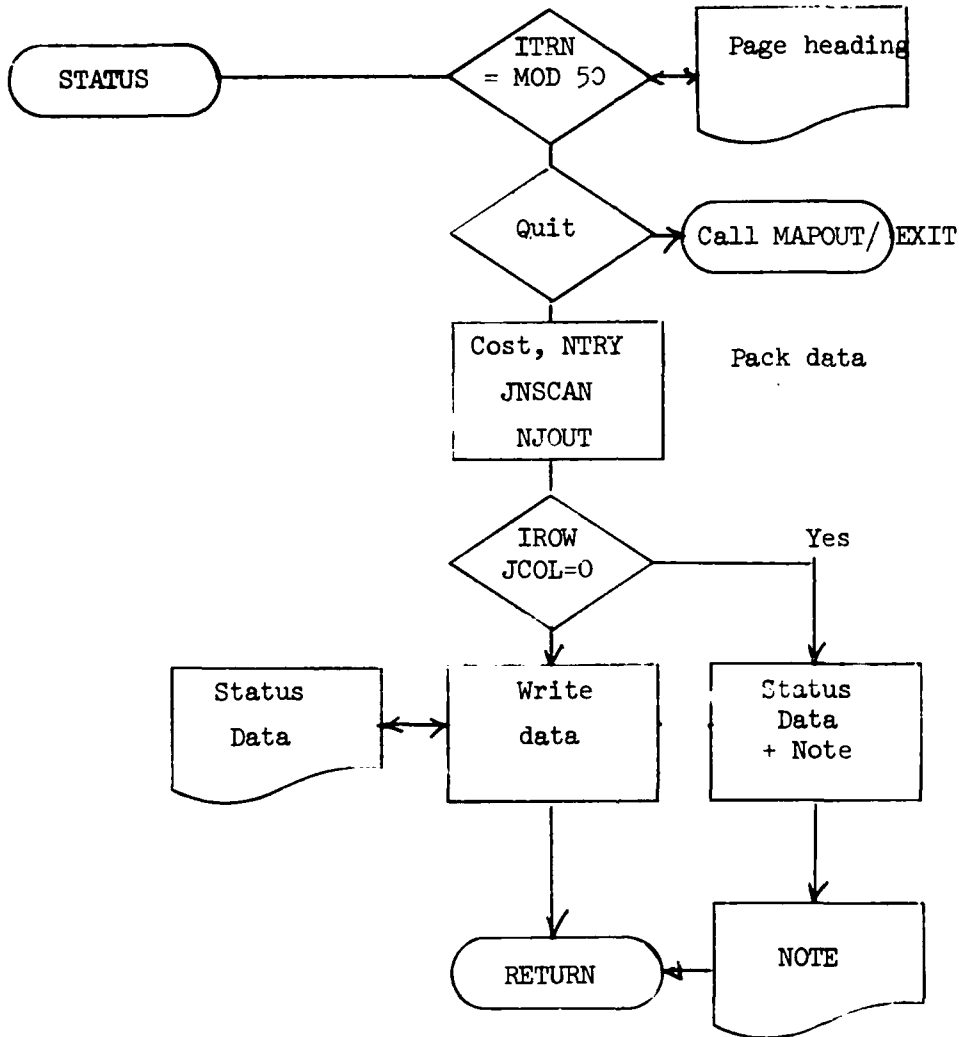
PRIMAL 8.



Subroutine STATUS

STATUS prints BRIMAL data

STATUS 1.



STATUS 2

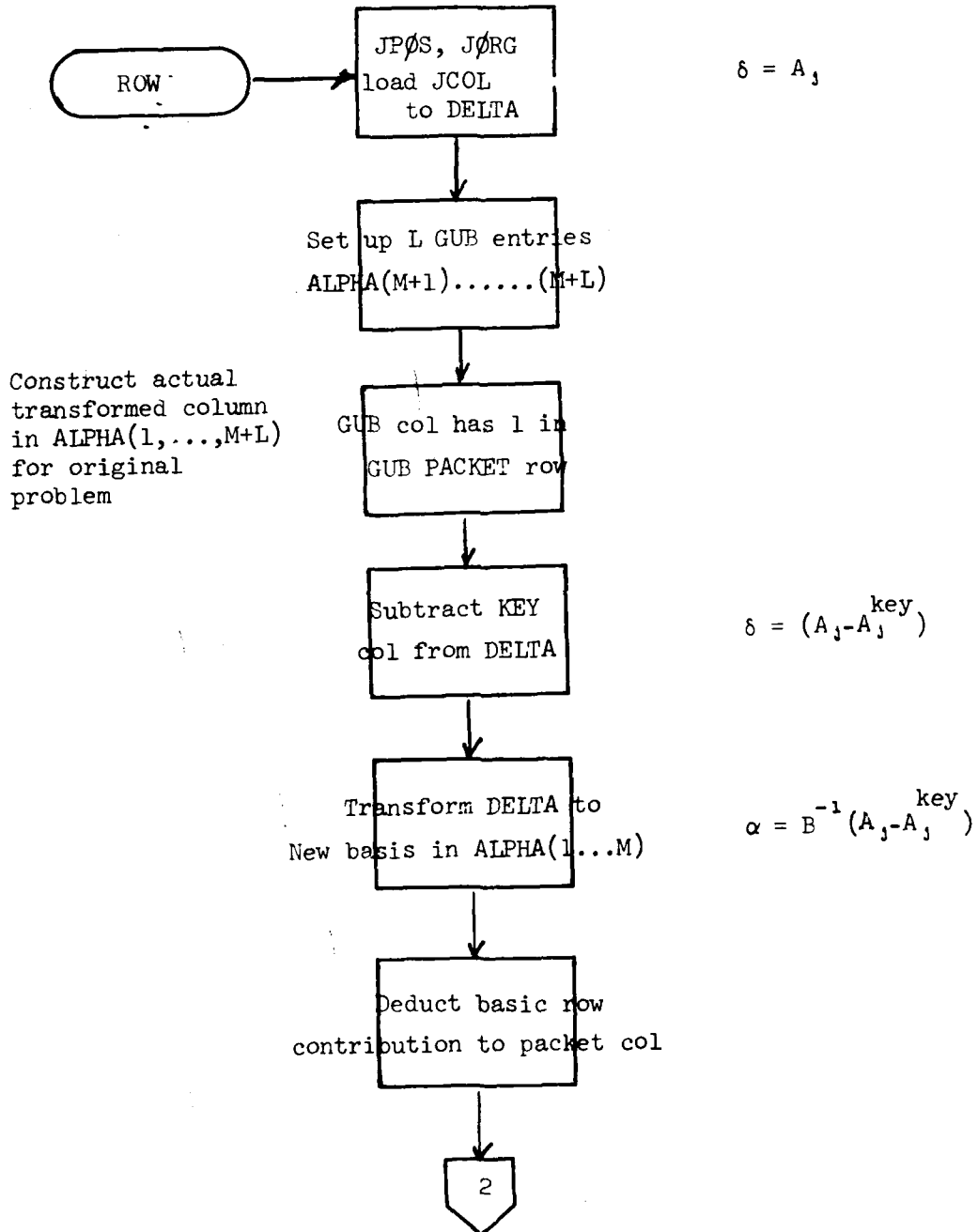
ERRØR - triple prints error messages

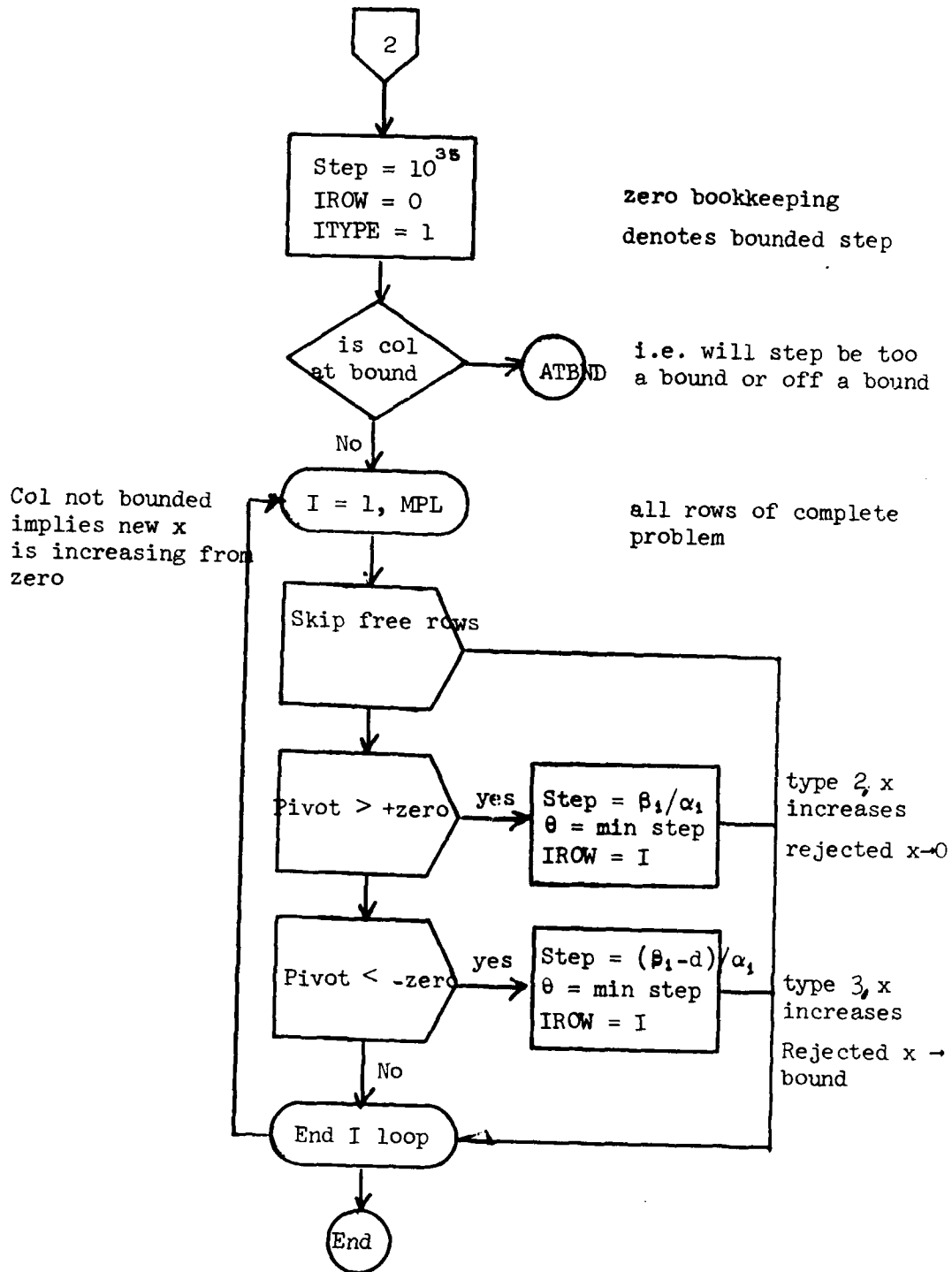
MESSG } single print error messages and print time in seconds since
MSSG } start of run.

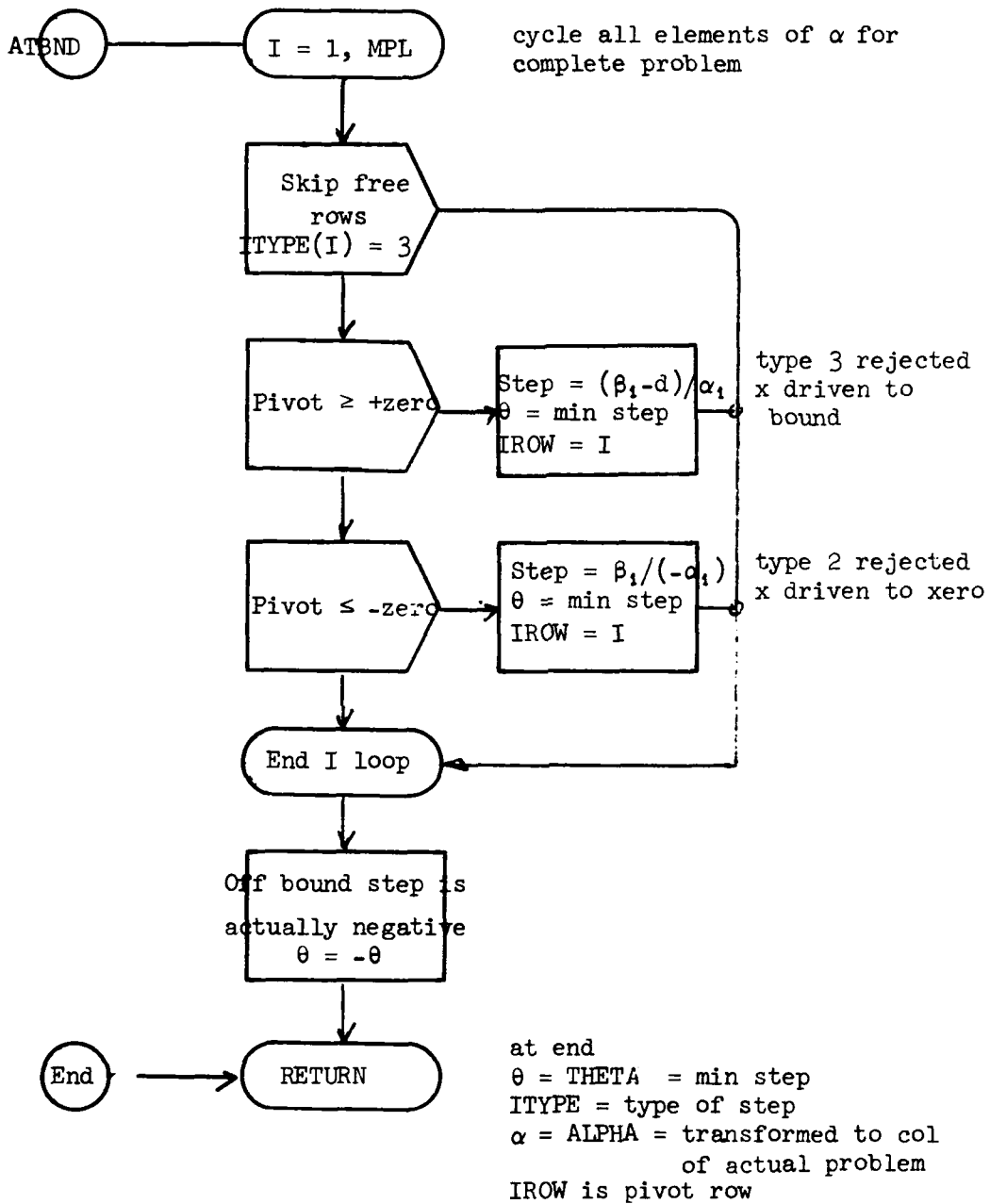
Subroutine ROW

ROW 1.

ROW computes current representation of selected column JCOL in core ALPHA then finds step MAX THETA which preserves feasibility.







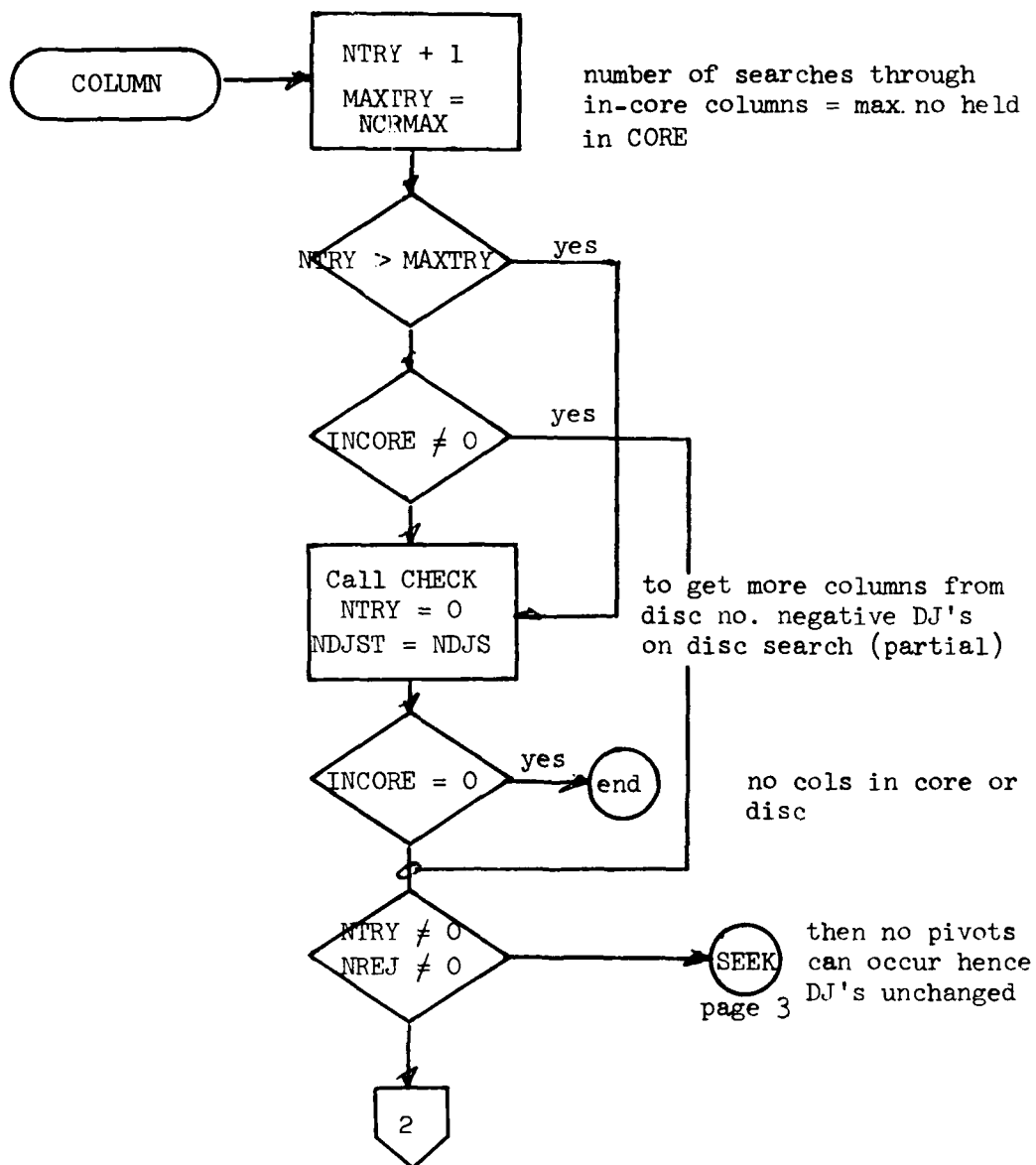
N.B. $\text{IROW} \leq M \Rightarrow$ pivot non-GUB row
 $> M \Rightarrow$ pivot on GUB row

Subroutine COLUMN

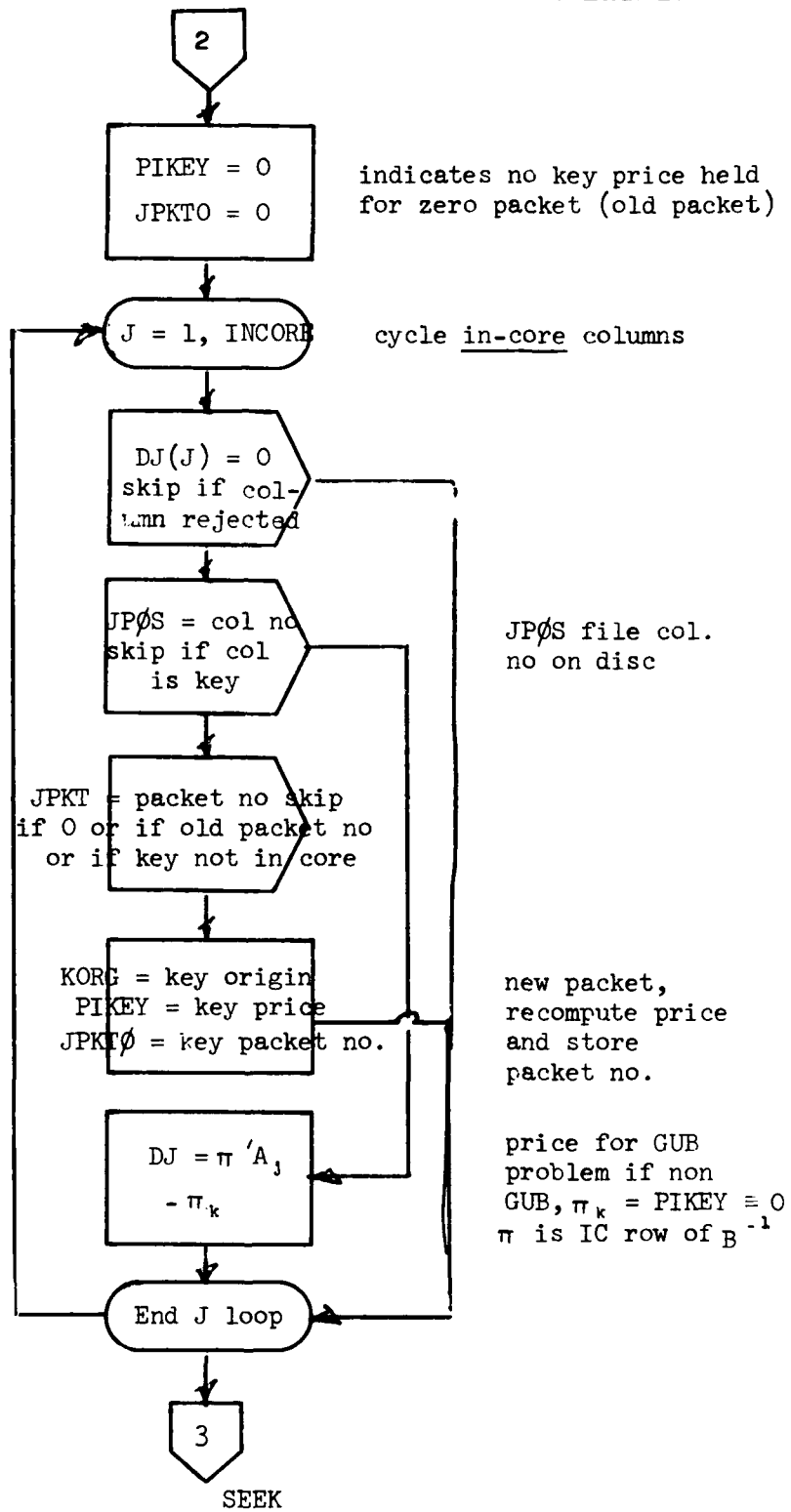
COLUMN 1.

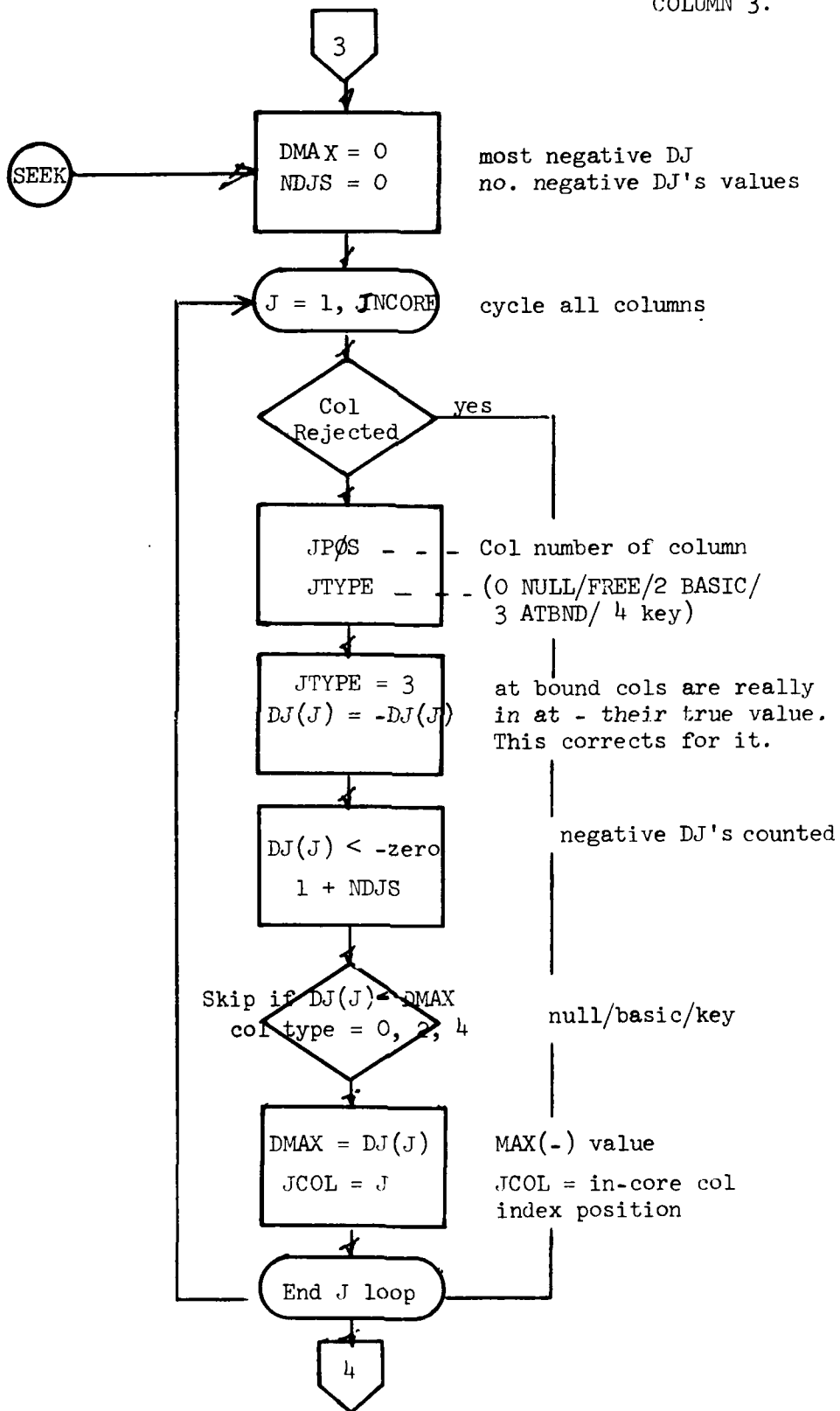
COLUMN selects a column JCOL from among vectors in core in AJ space.

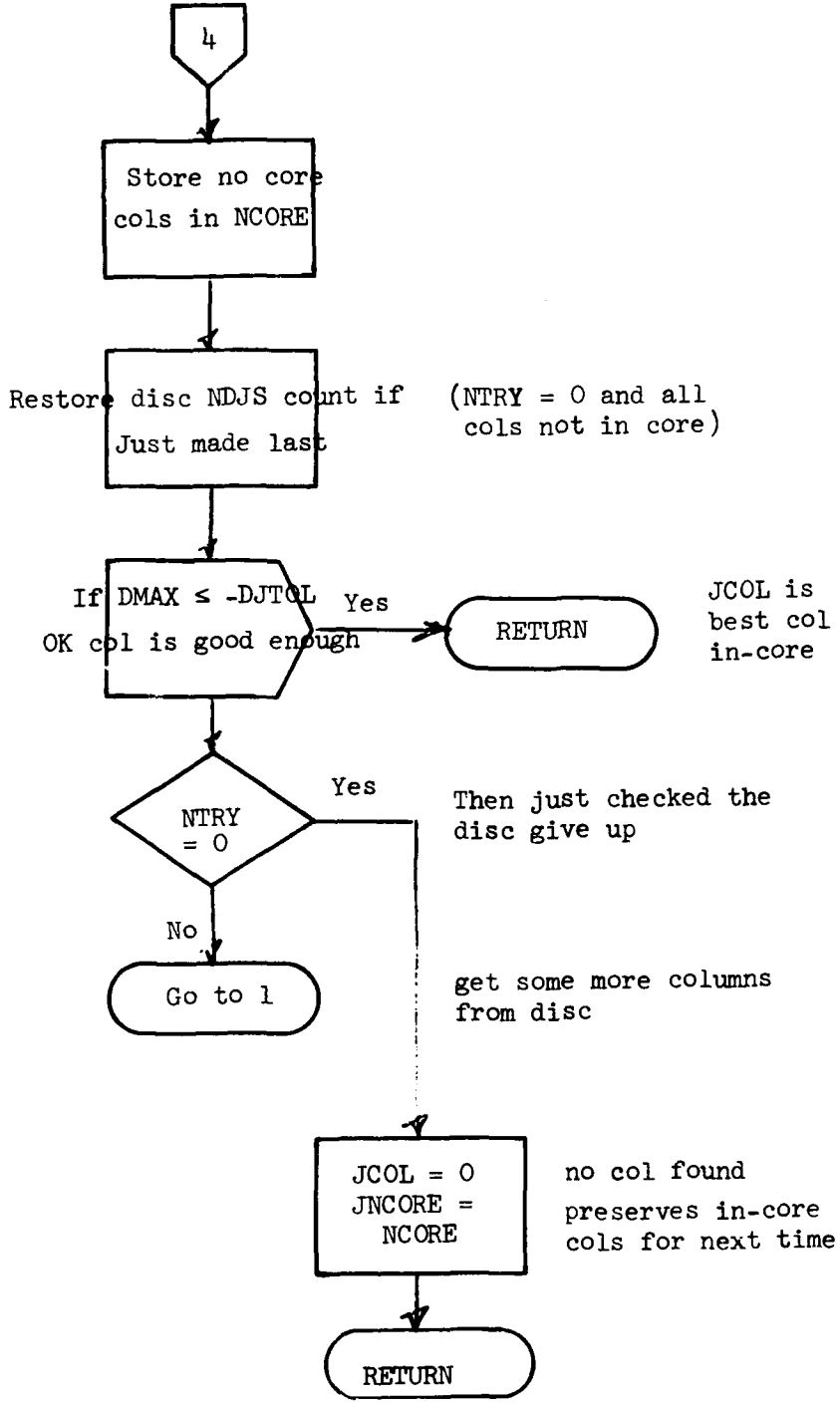
If no columns price out, it calls CHECK to search disc for replenishment.



Reprices old columns in core unless the prices obviously haven't changed (same $\beta-1$).



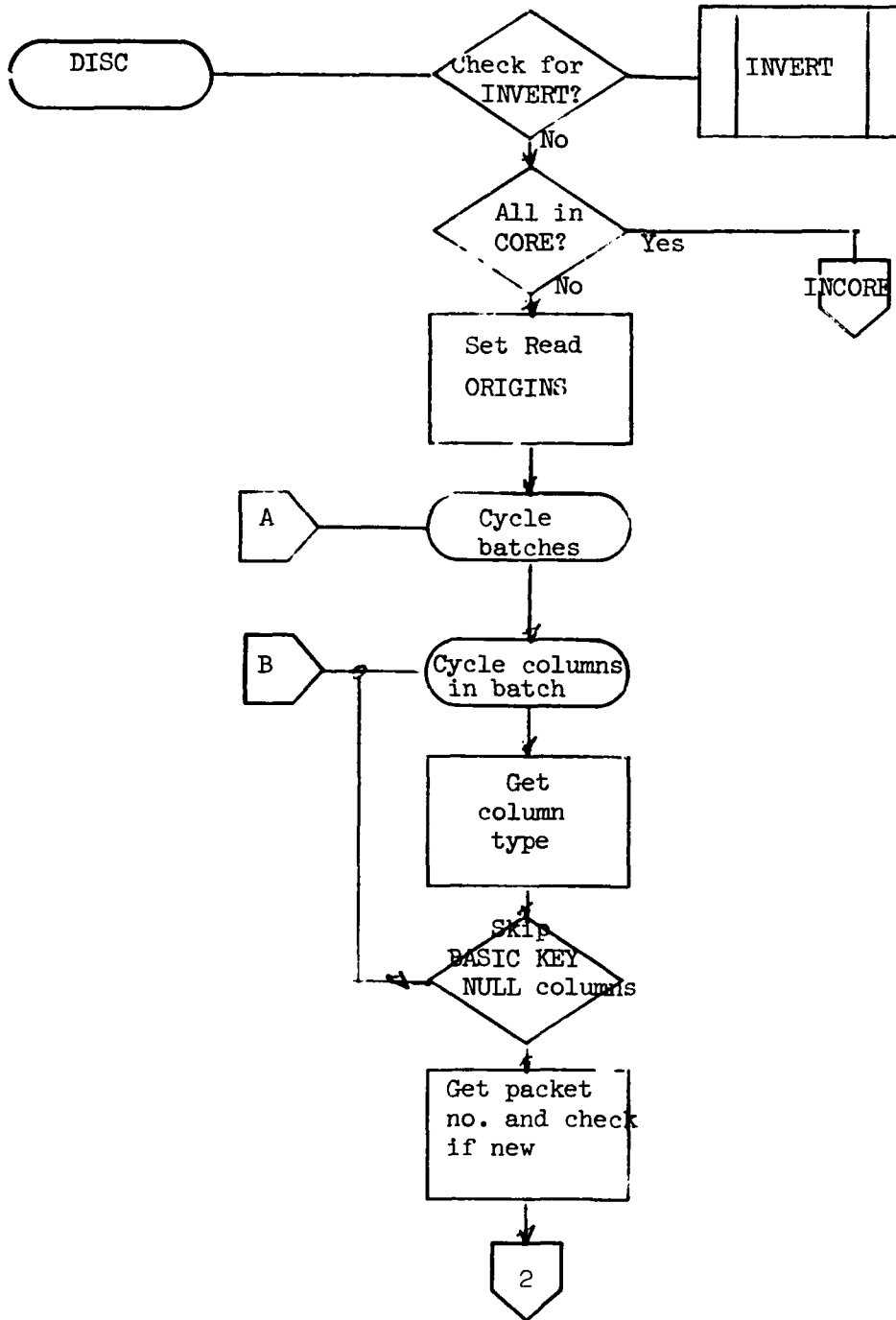


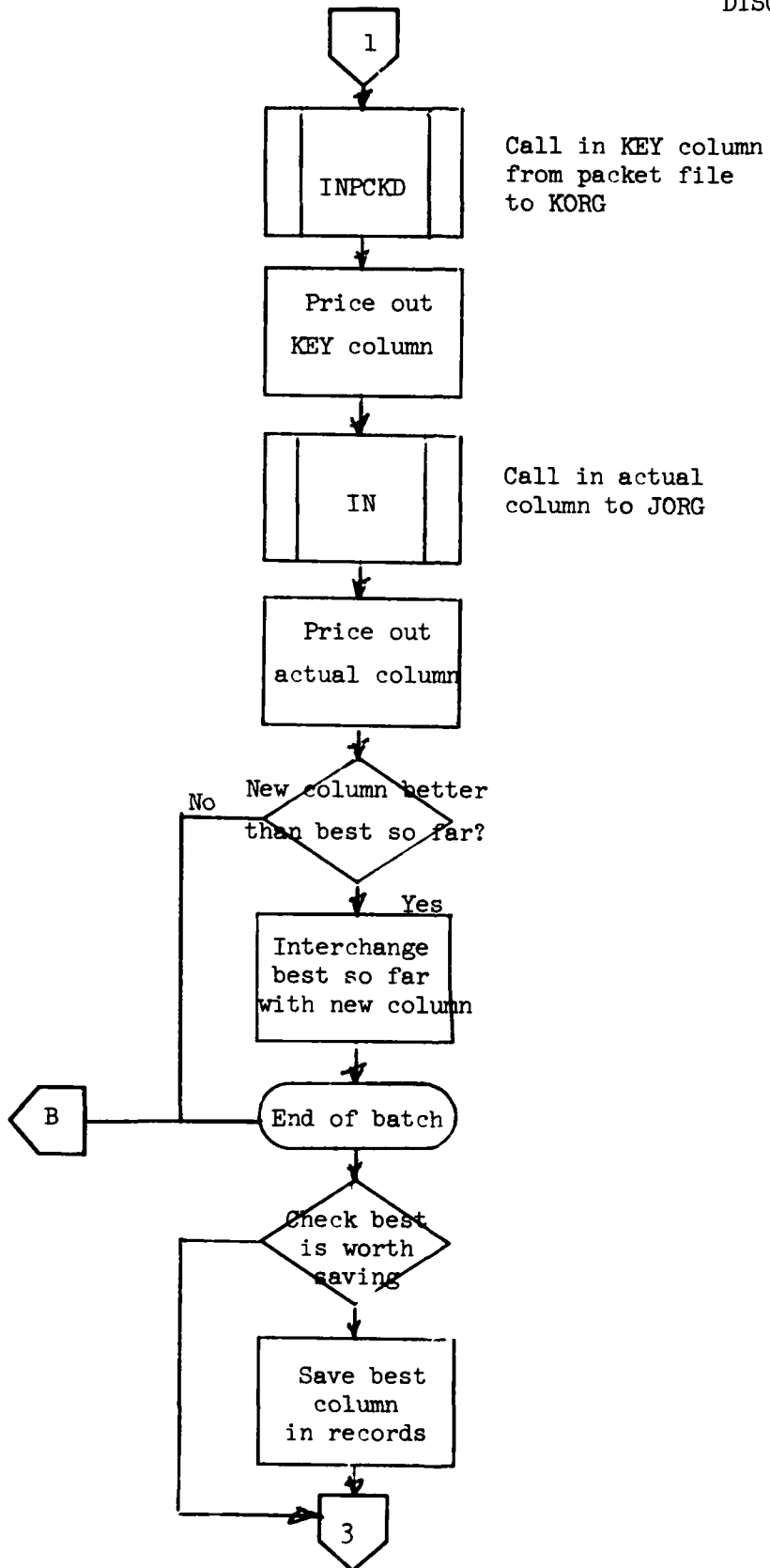


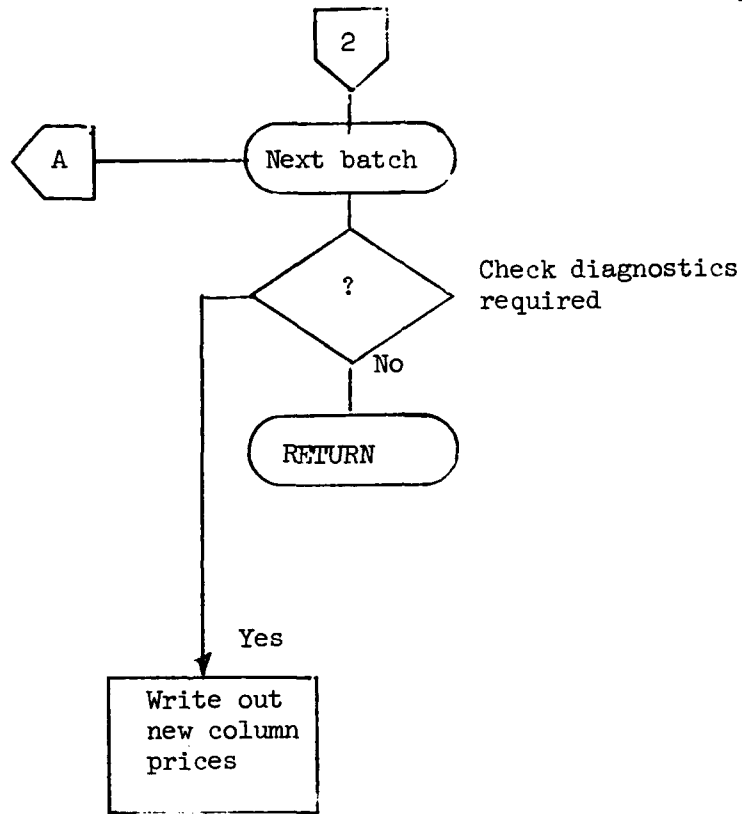
Subroutine DISC

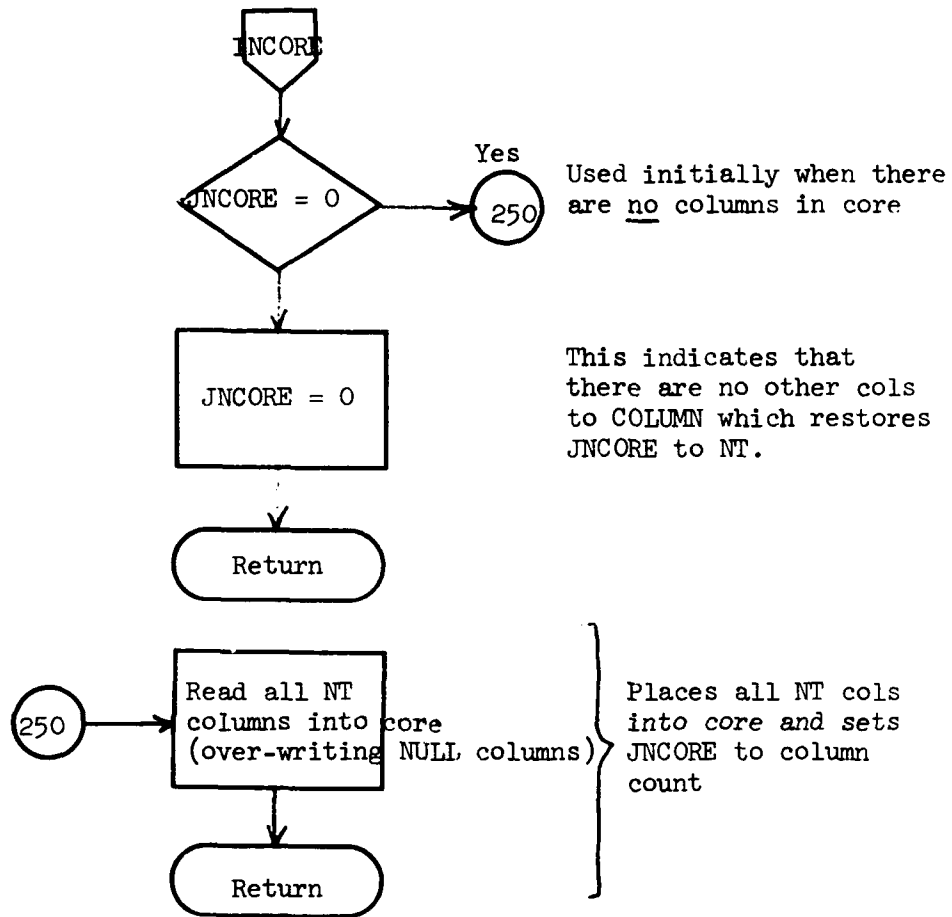
DISC 1

Checks if an inverse is necessary and then checks the DISC files IA1, IA2 for more useful columns using IN and INPCKD.







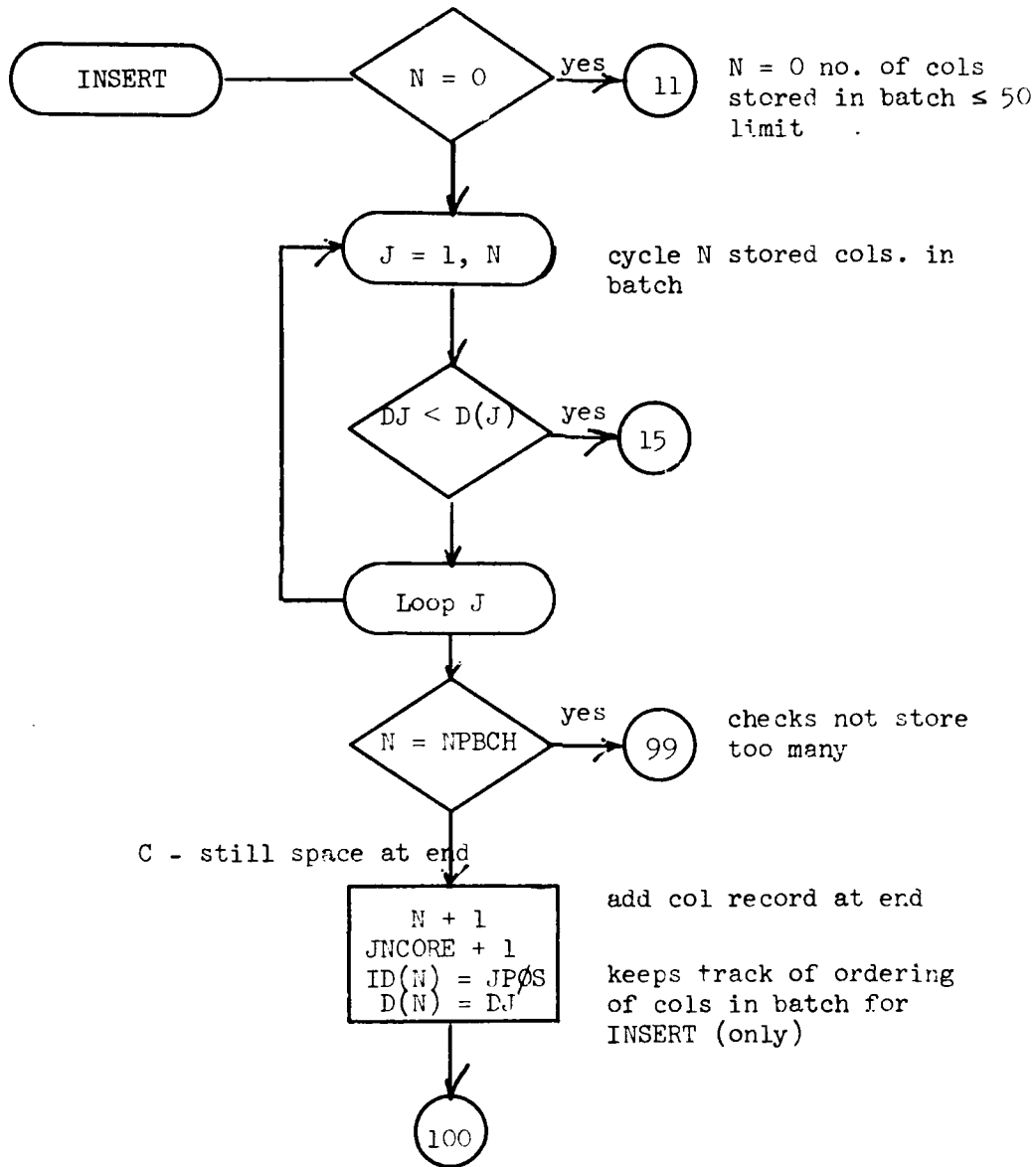


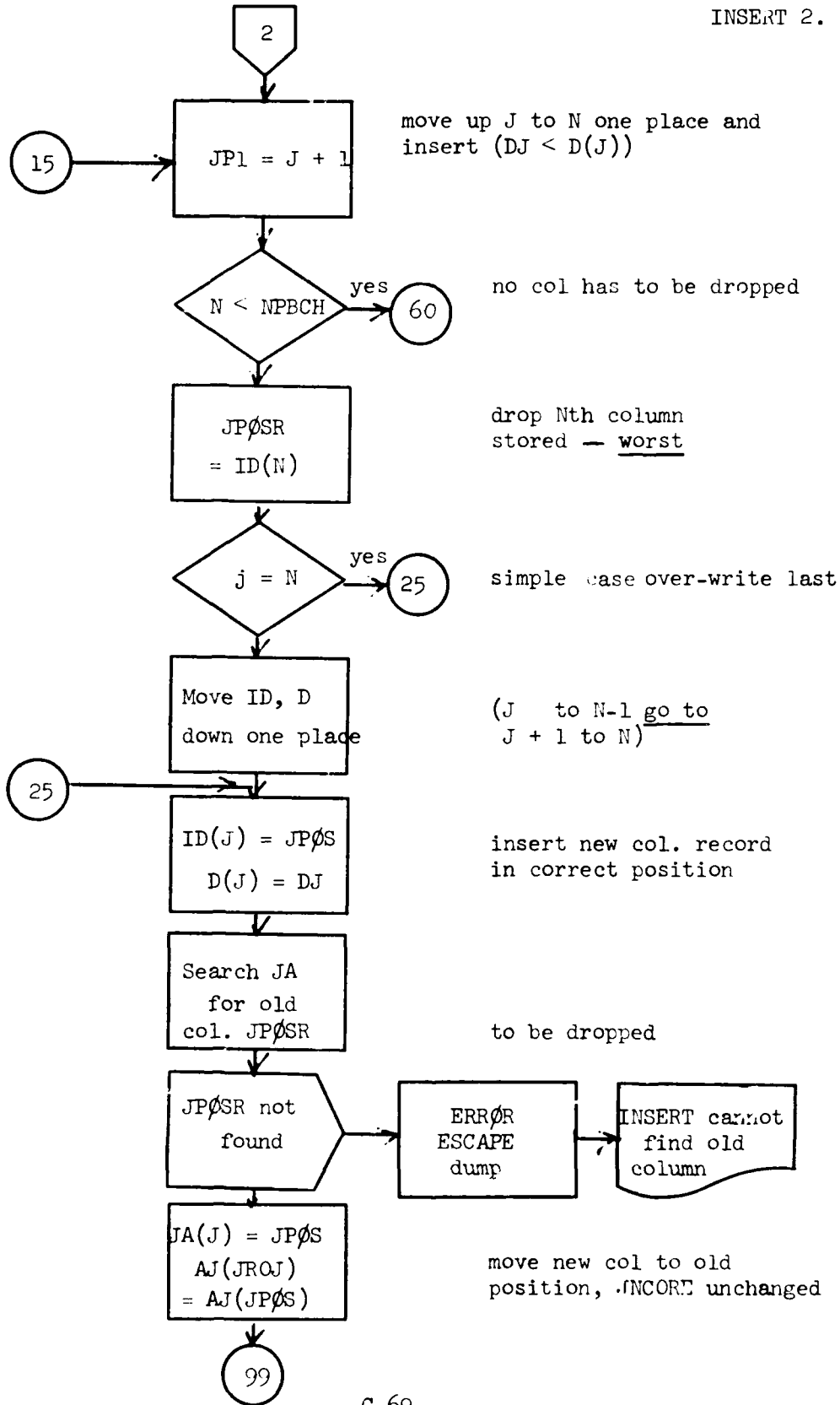
N.B. Once the columns are read into core, they stay there because COLUMN always restores JNCORE to NT and keeps track of them at the end of each phase.

Subroutine INSERT

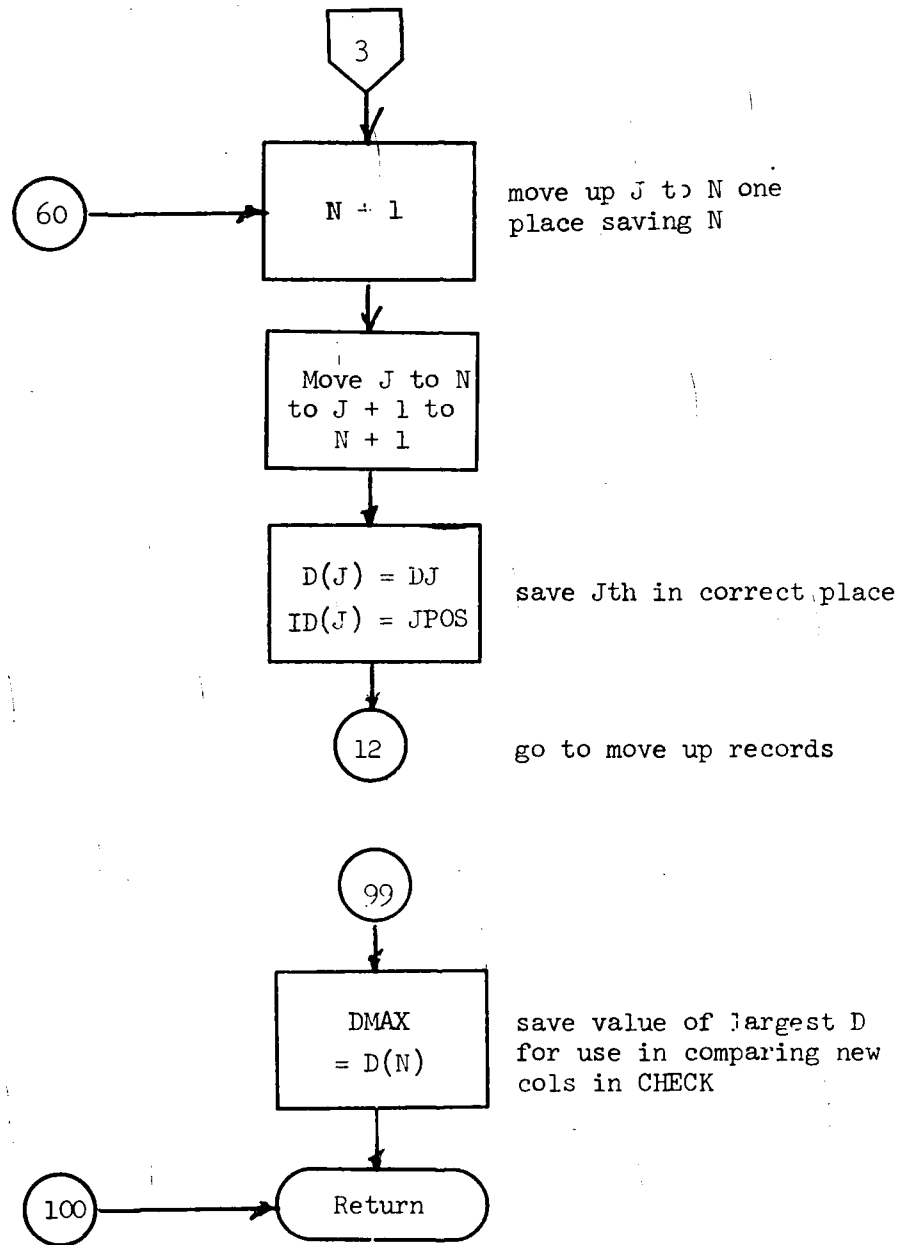
INSERT 1.

INSERT used by CHECK to order columns selected in the batch. It keeps track of worst column stored of the batch and over writes it if a better one is offered.

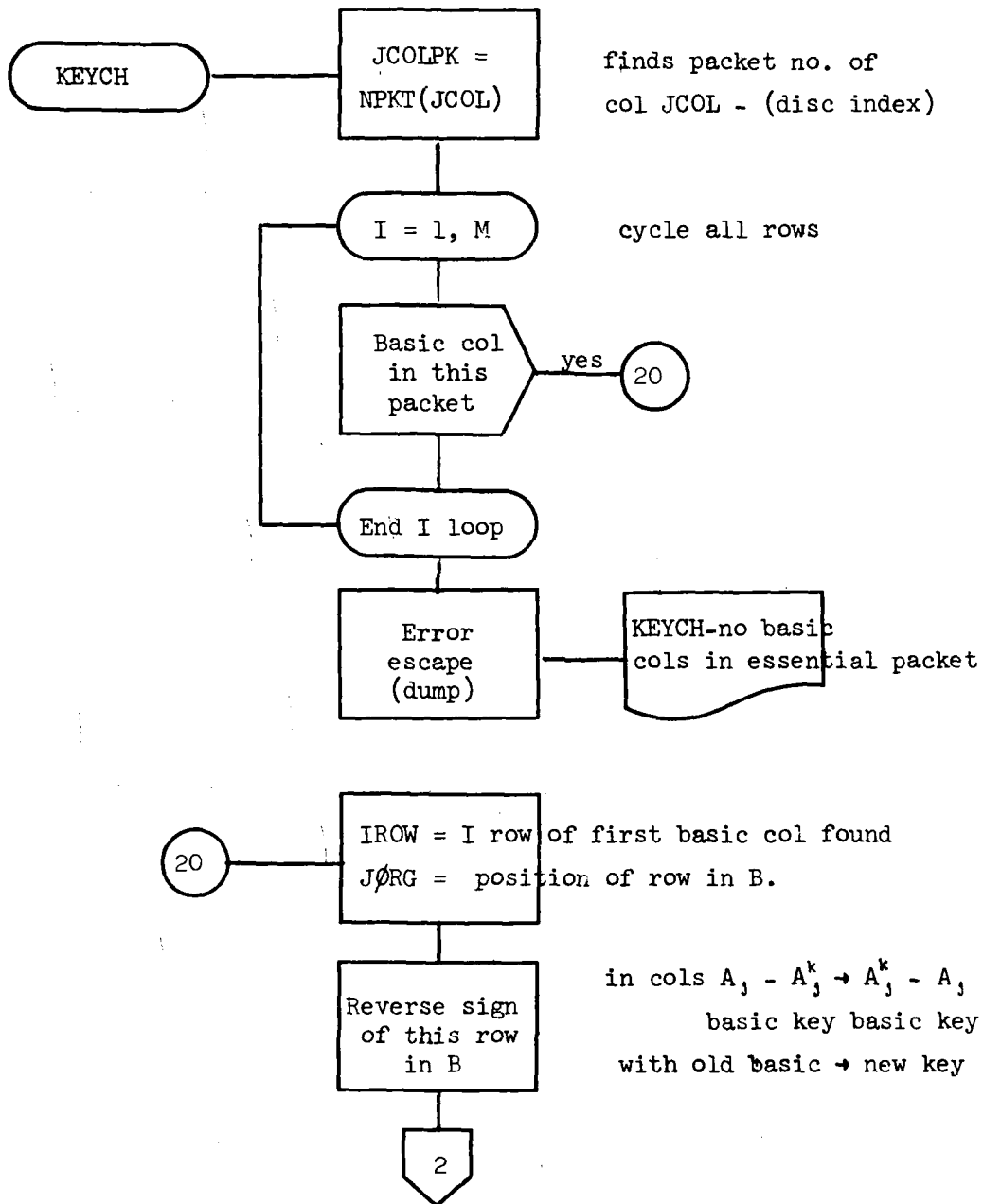


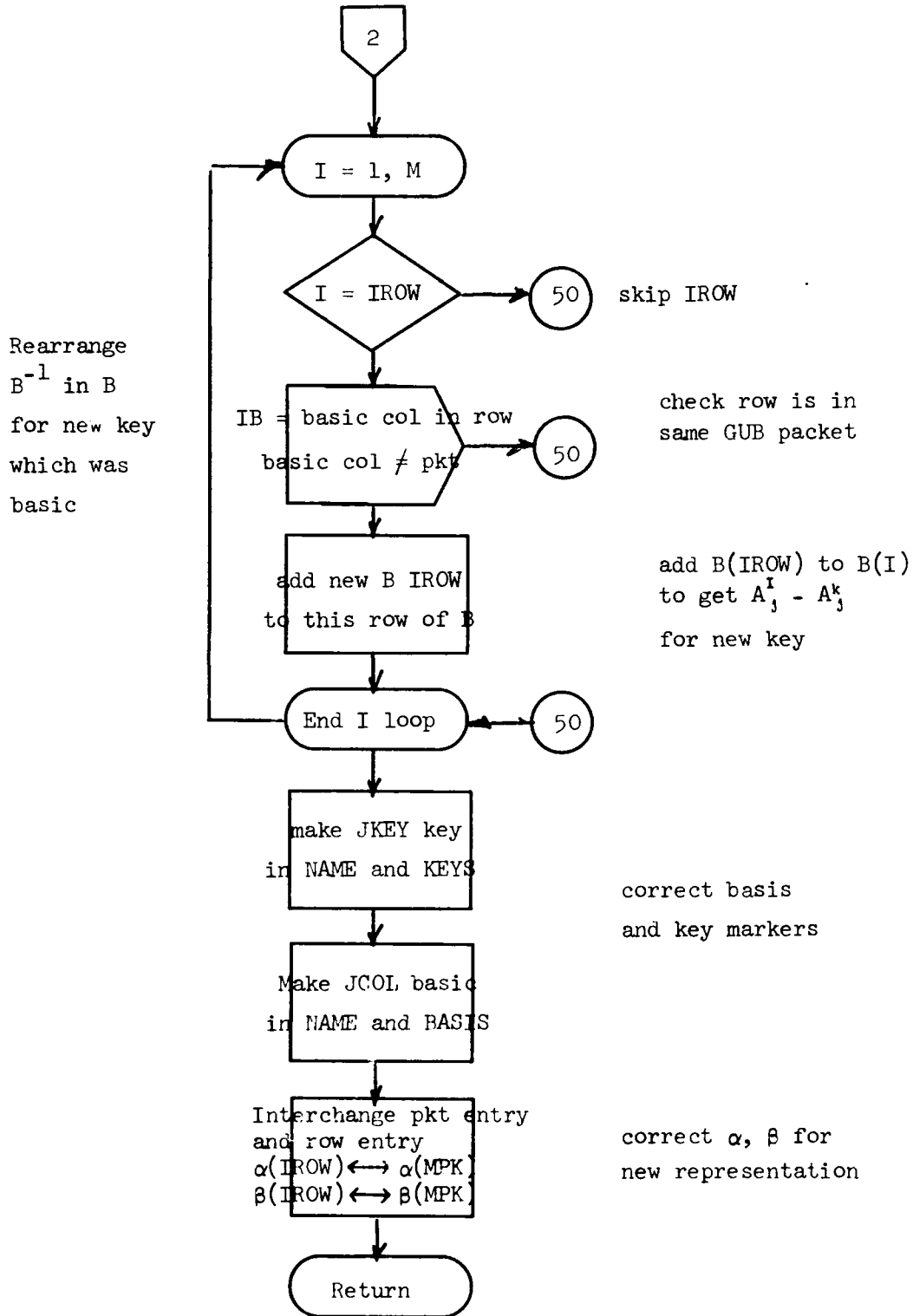


INSERT 3.



KEYCH changes key to make JCOL basic in some row IROW found, making old col key, and corrects α , β and B^{-1} .

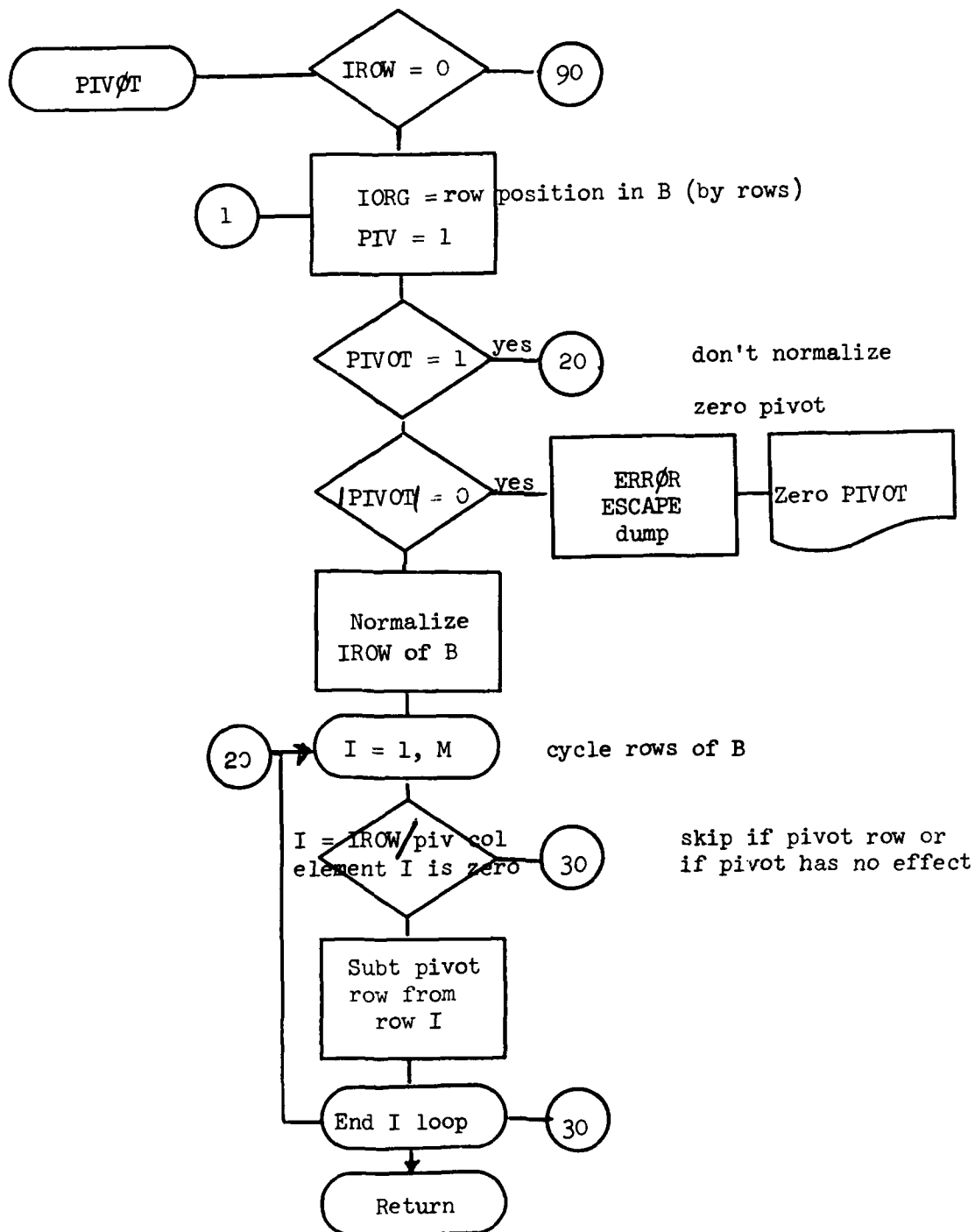




Subroutine PIVOT

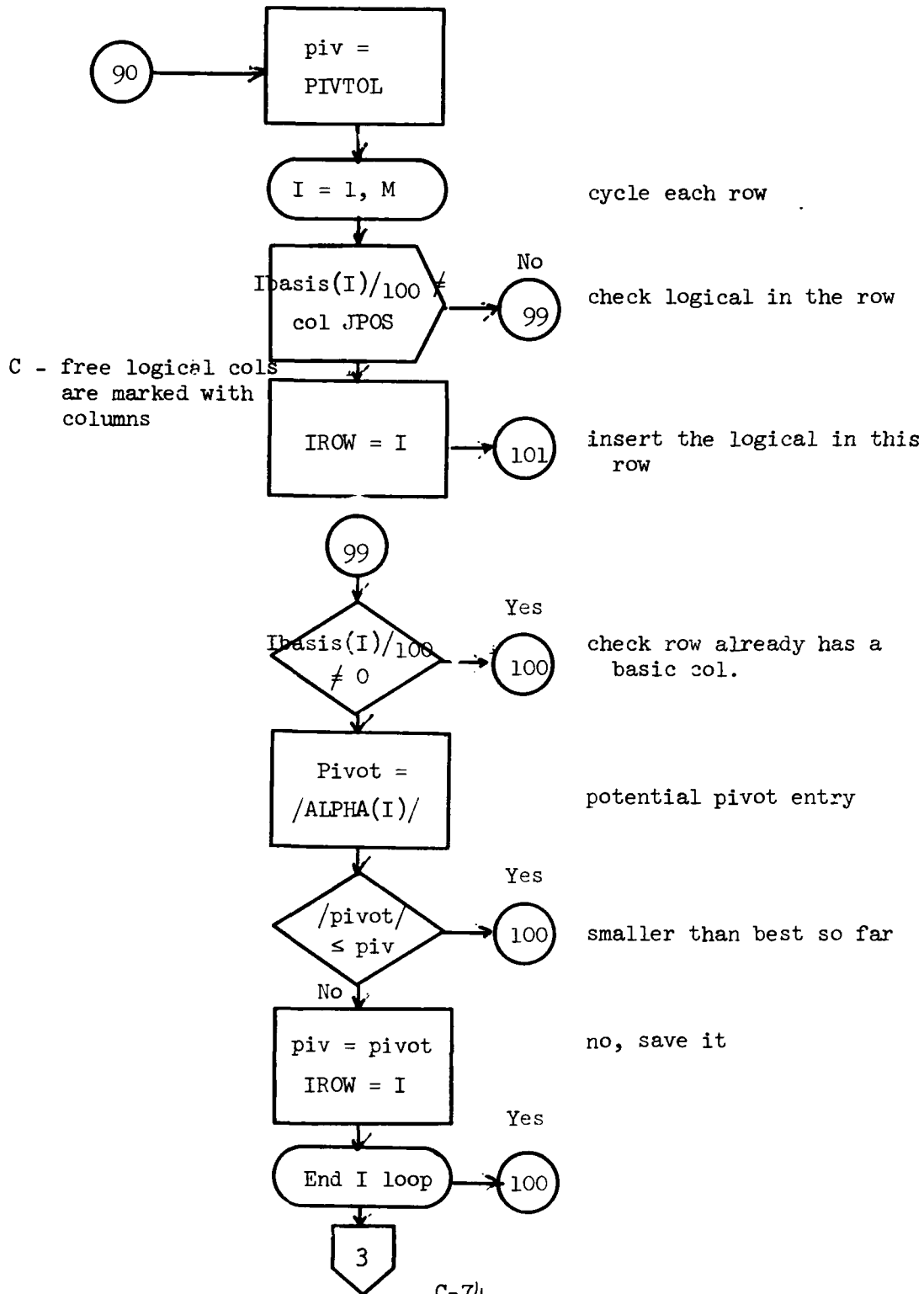
PIVOT 1.

PIVOT inserts current representation of column ALPHA into basic inverse B at IROW, if IROW = 0 it finds a slot for ALPHA or rejects it.

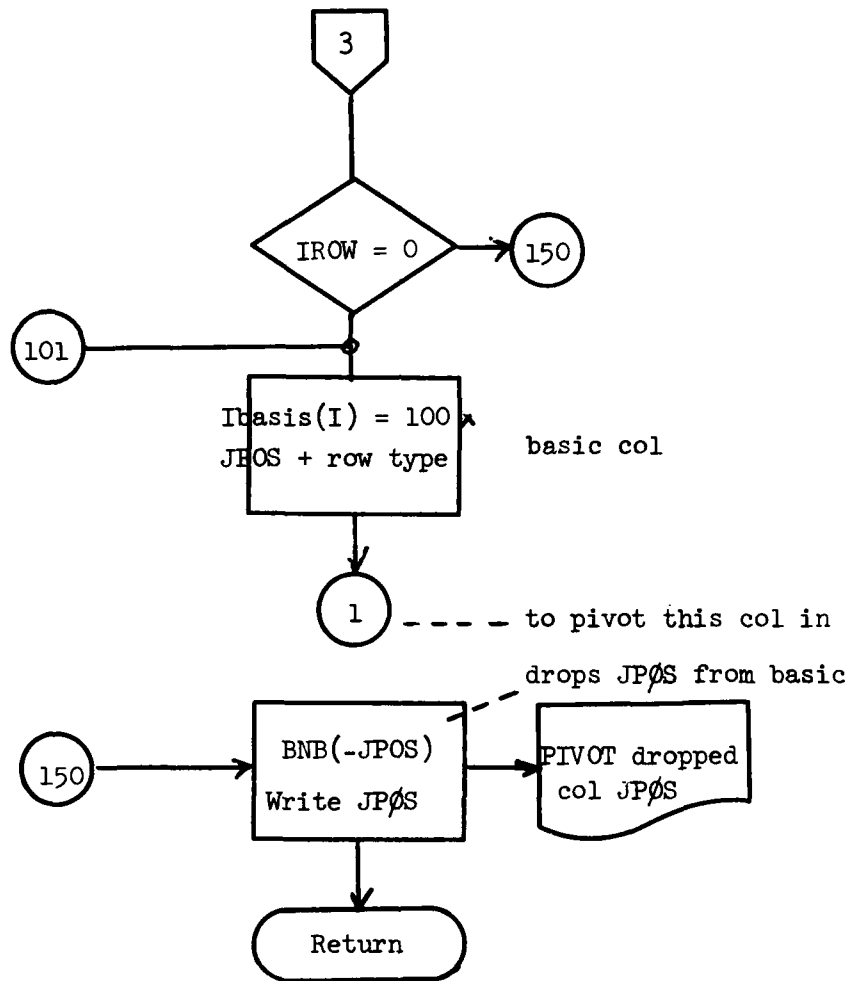


PIVOT 2.

Find best row for ALPHA called from INVERT when row not known.

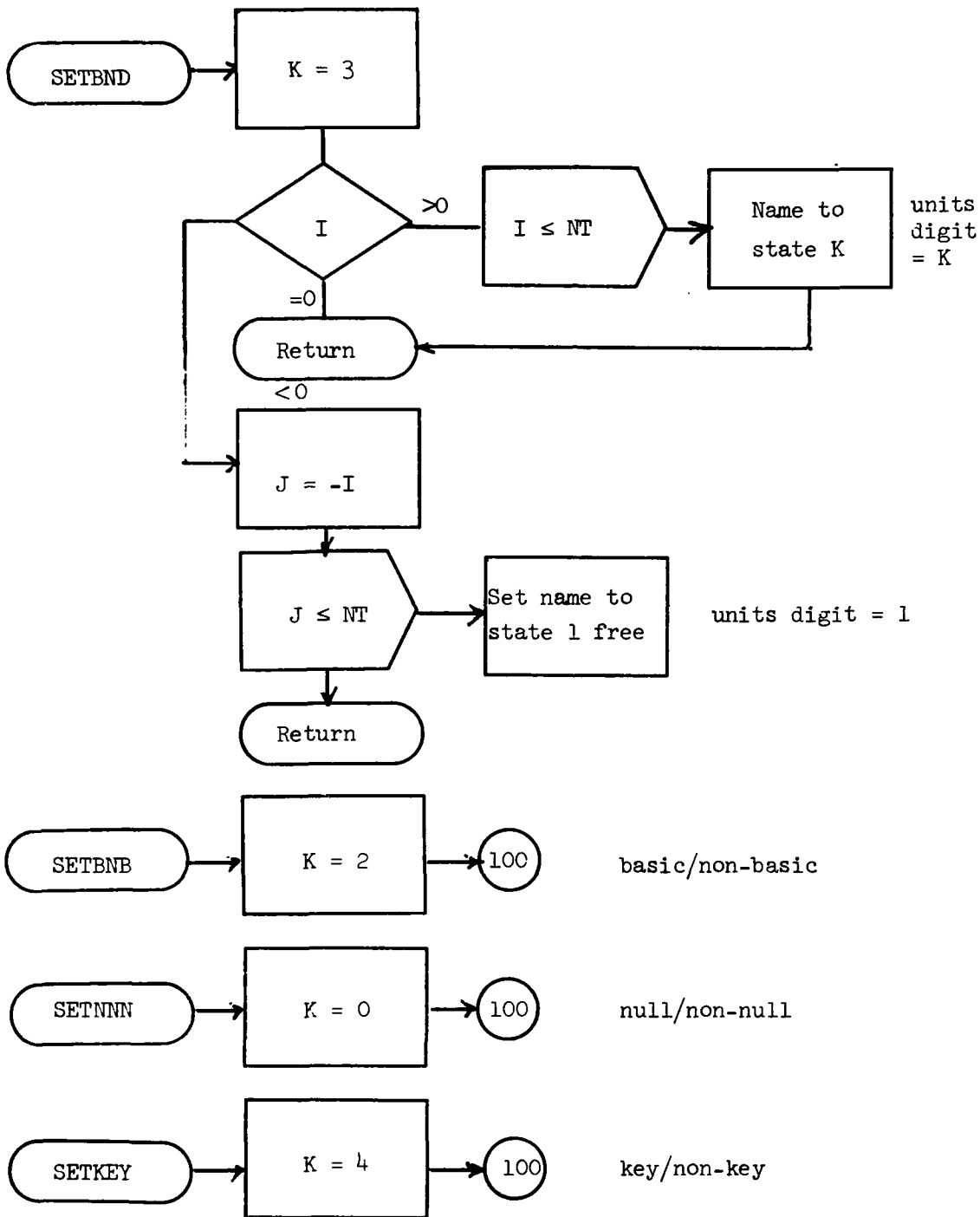


PIVOT 3.



SETBND 1.

SETBND, SETBNB, SETKEY, SETNNN all set or unset to state of a variable to bound/basic/key/null.

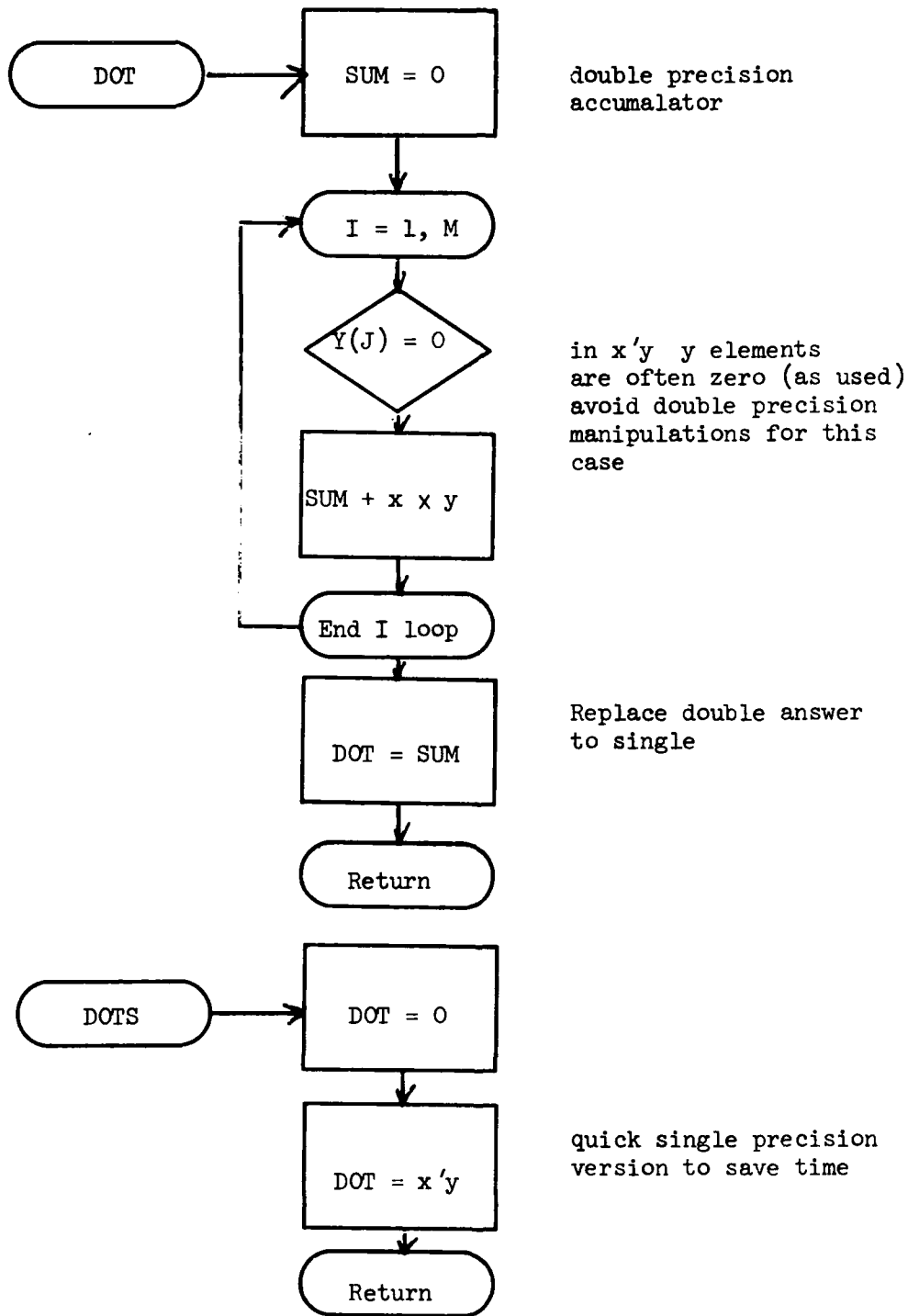


FUNCTION DOT, DOTS

DOT
DOTS

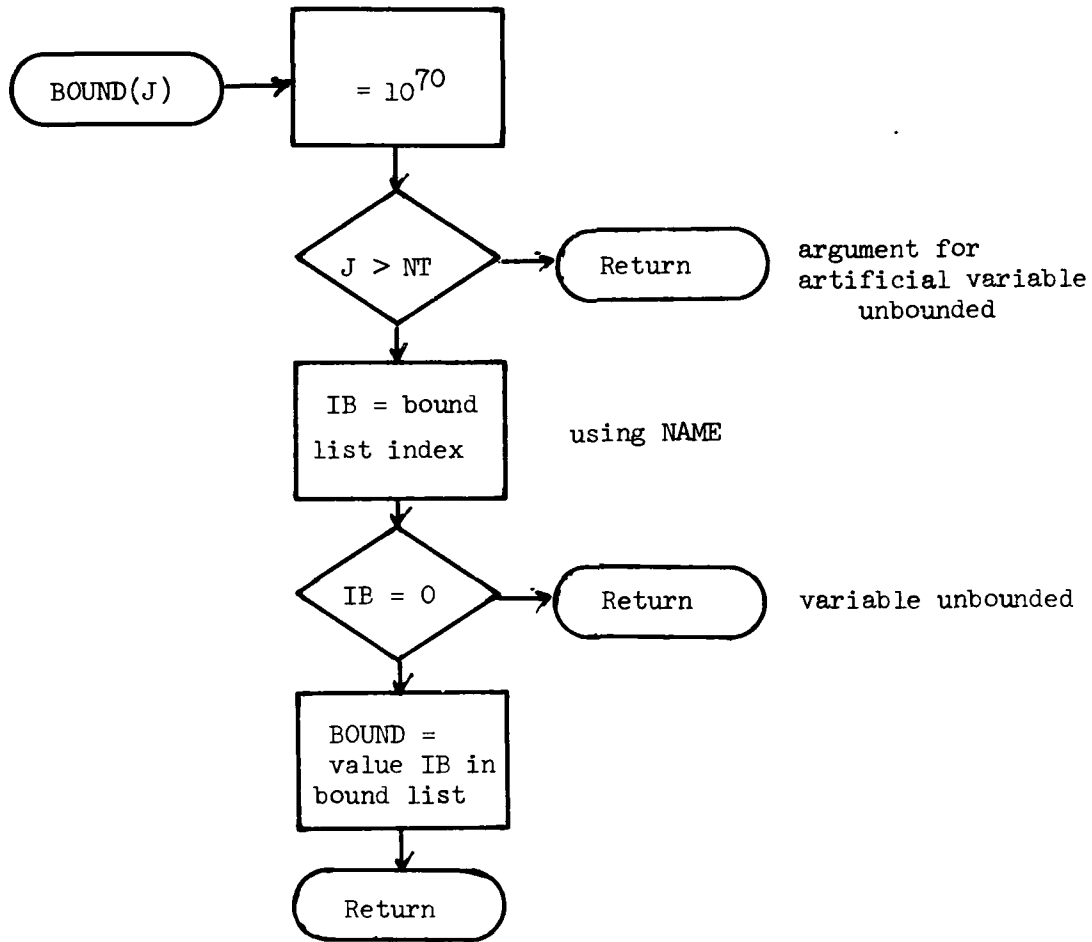
DOT and DOTS evaluate double and single precision inner products

$DOT = x'y$.



FUNCTION BOUND

BOUND - checks its argument and picks up the variable bound value.



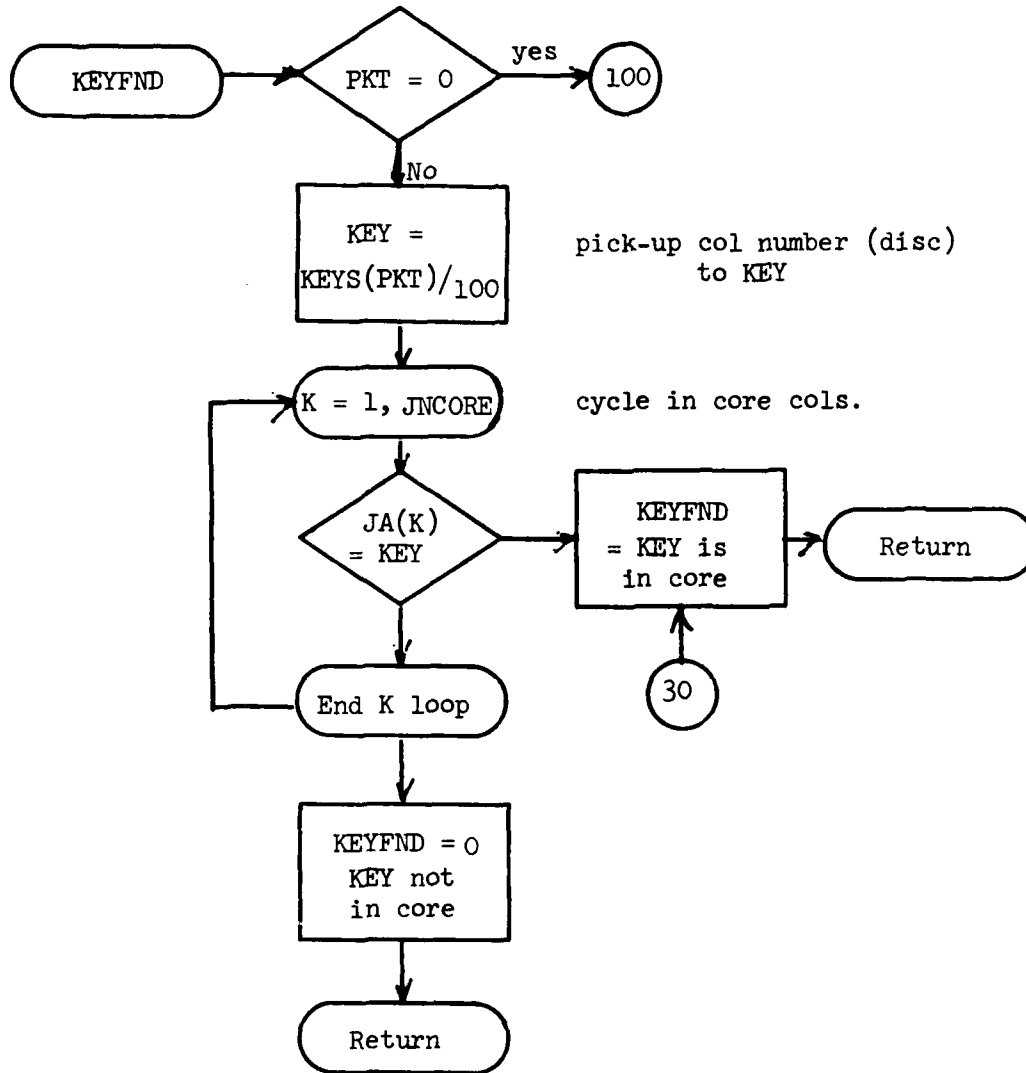
NB. NAME format has least significant decimal digits as shown.

.....	O	O	B	B		K	K	K	K		S
	bound	index				GUB					state
	or	0				packet					

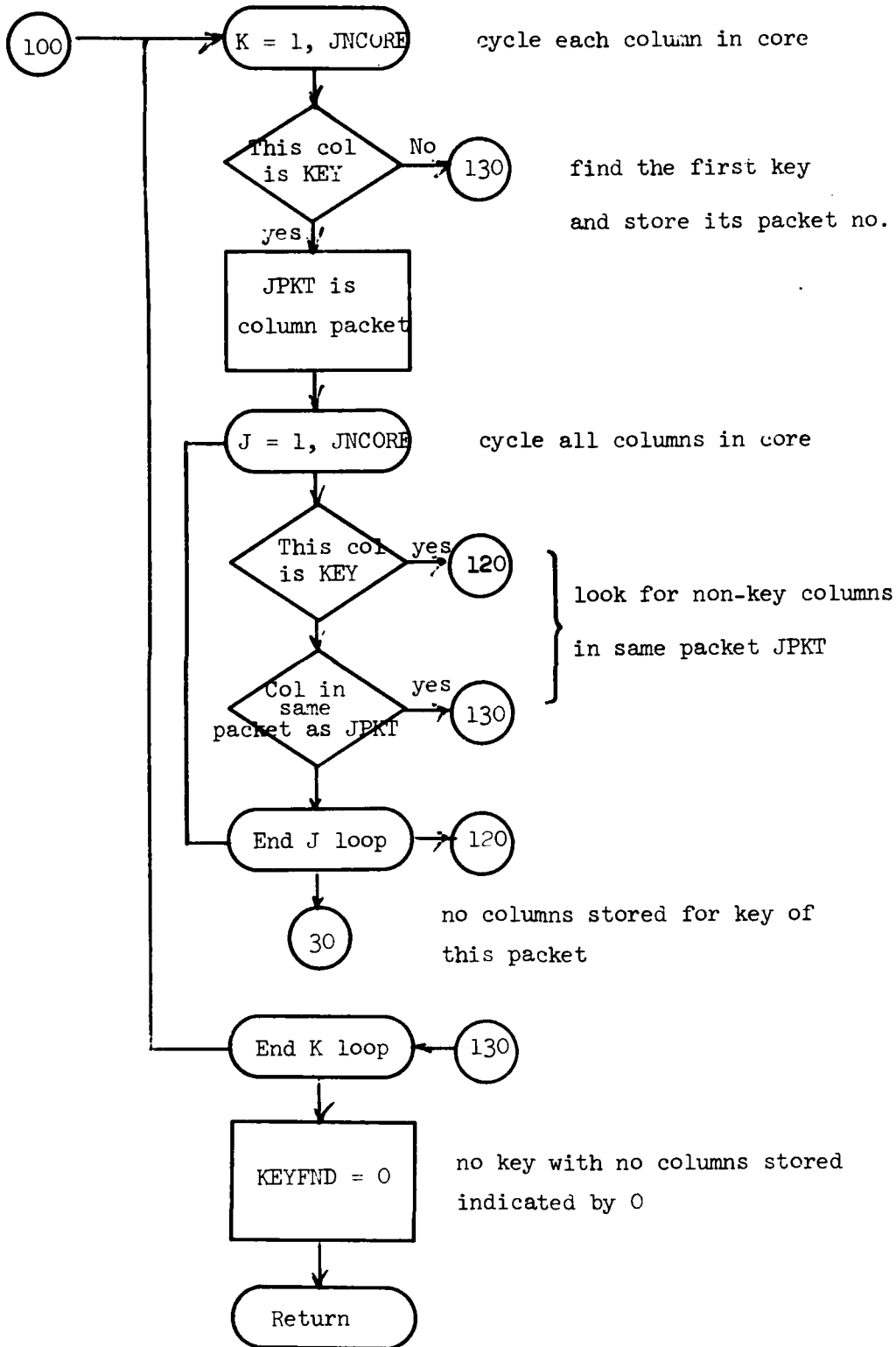
Function KEYFND

KEYFND 1.

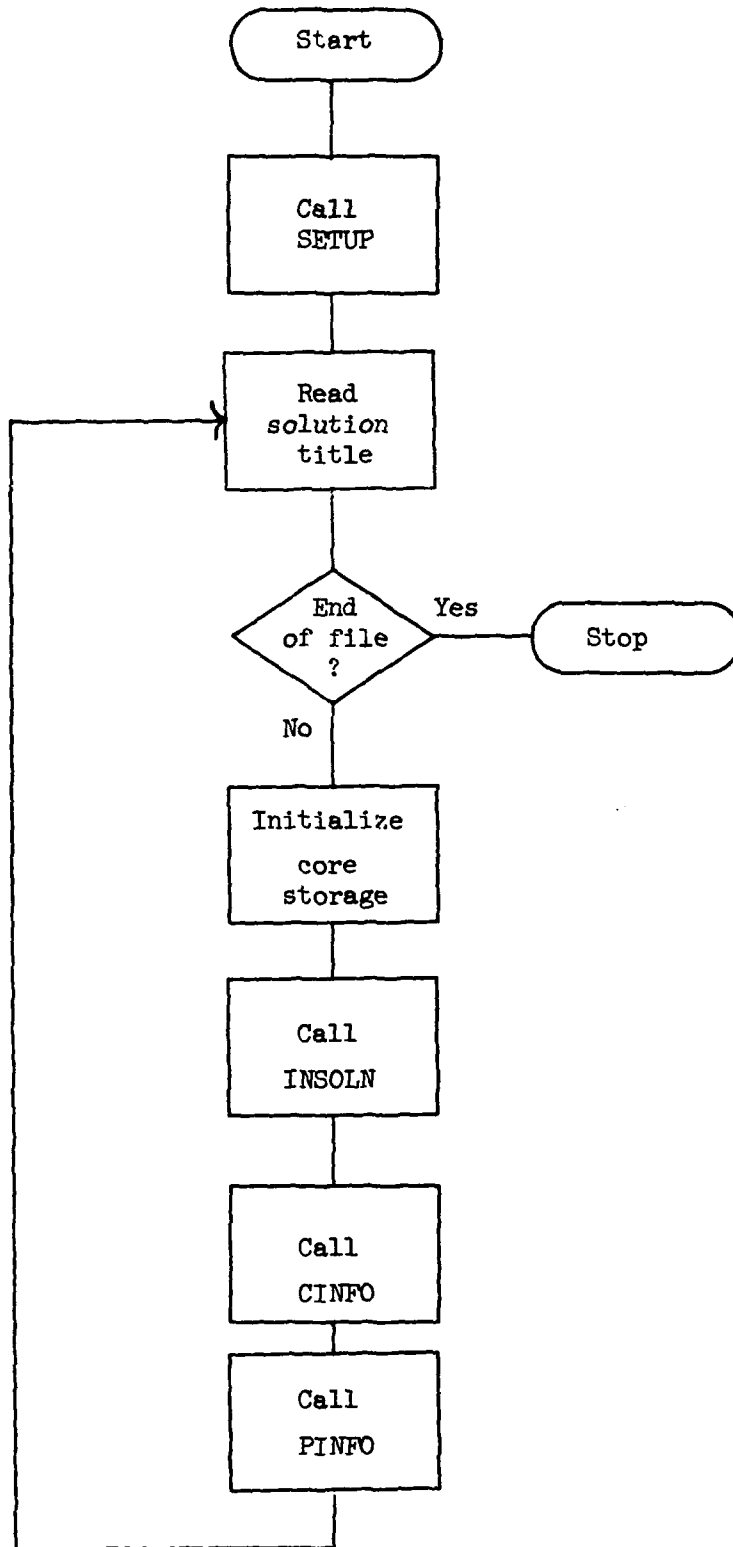
KEYFND finds key column of a packet in core or returns 0.



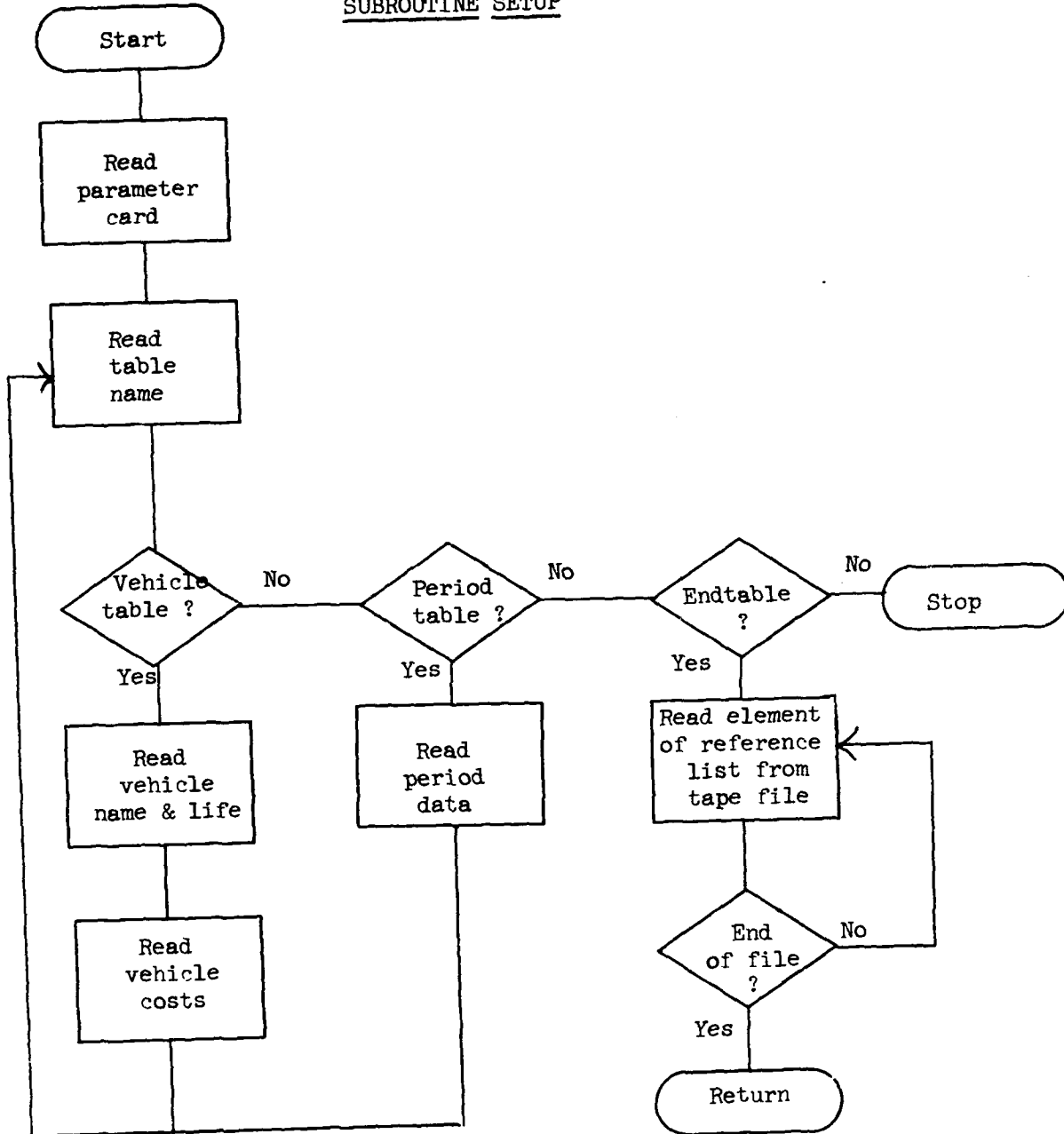
KEYFND 2.



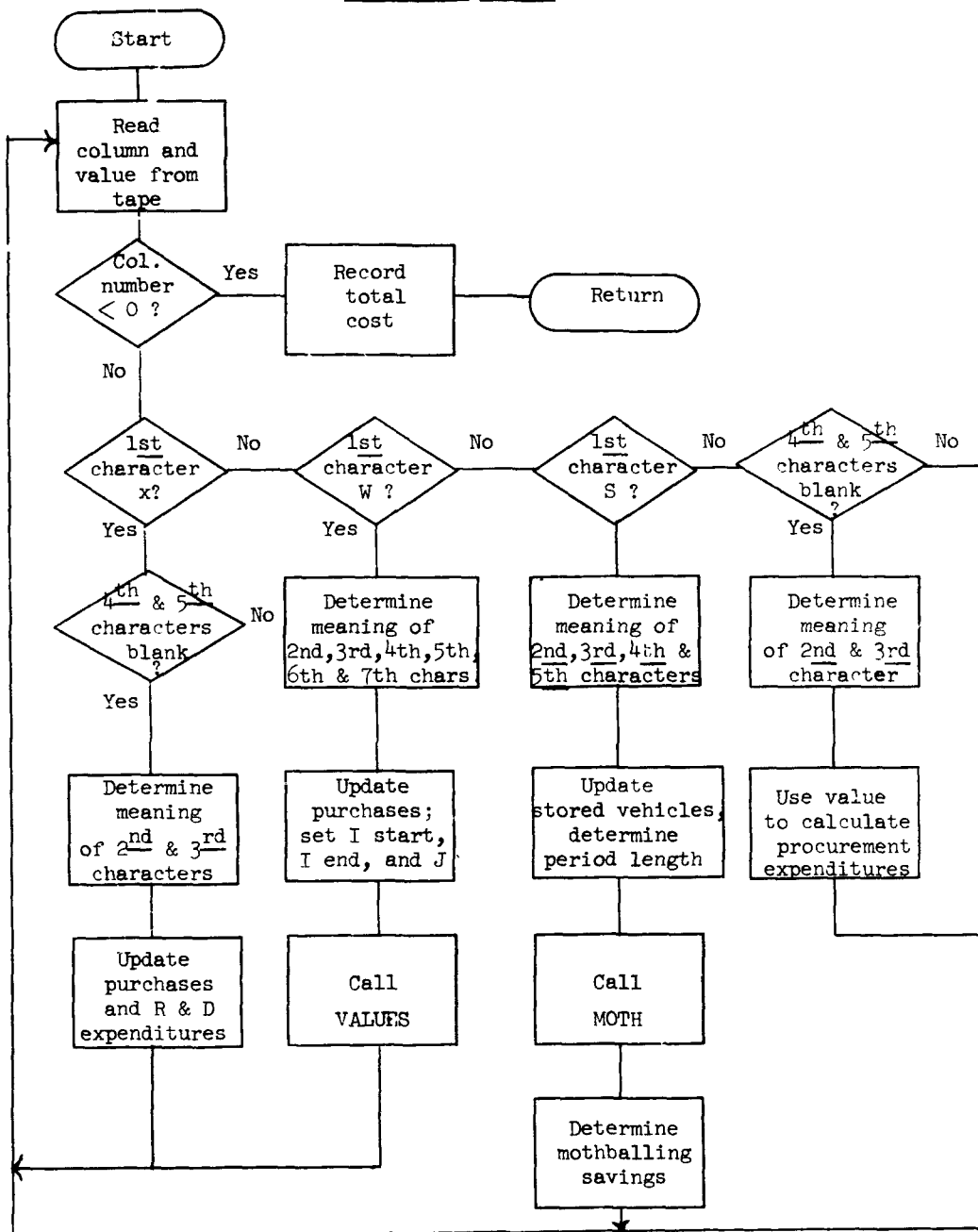
PROGRAM REPGEN



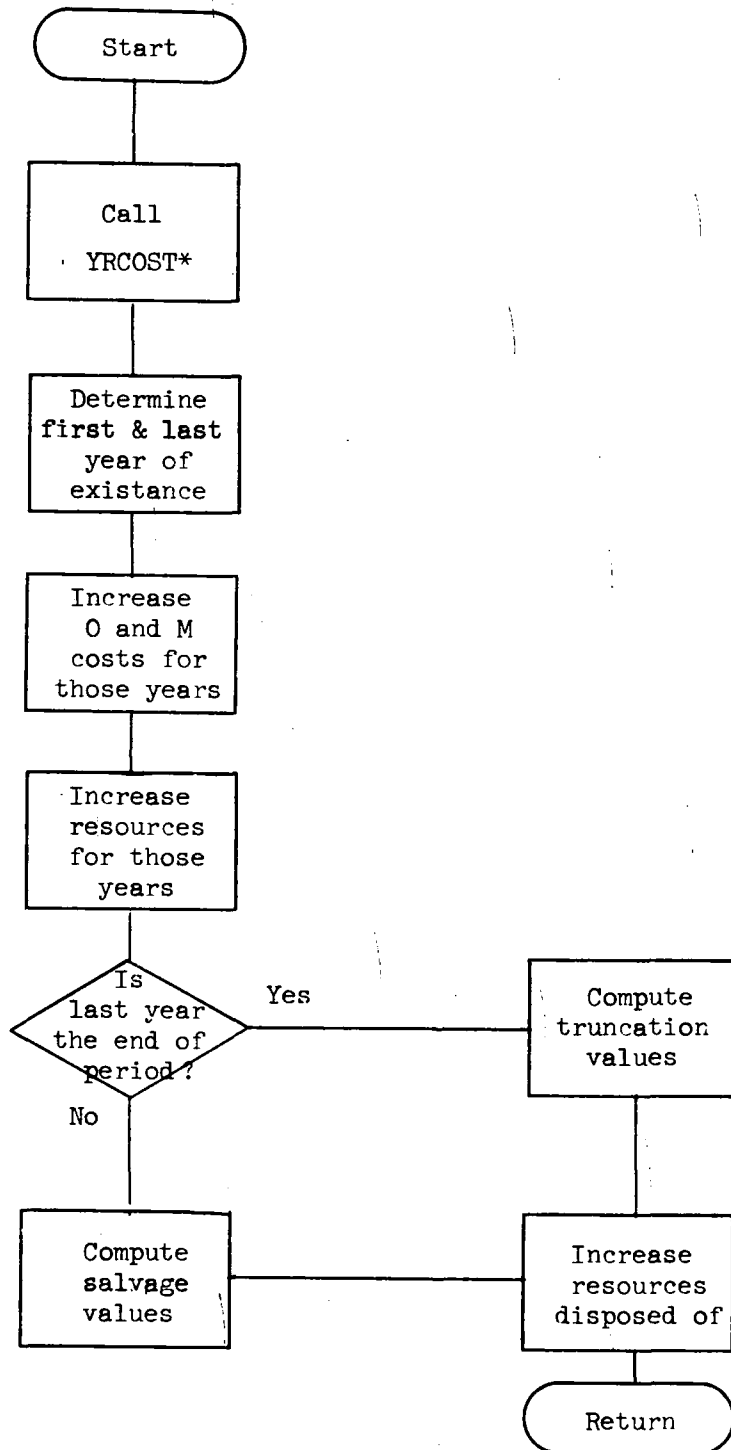
SUBROUTINE SETUP



SUBROUTINE INSOLN

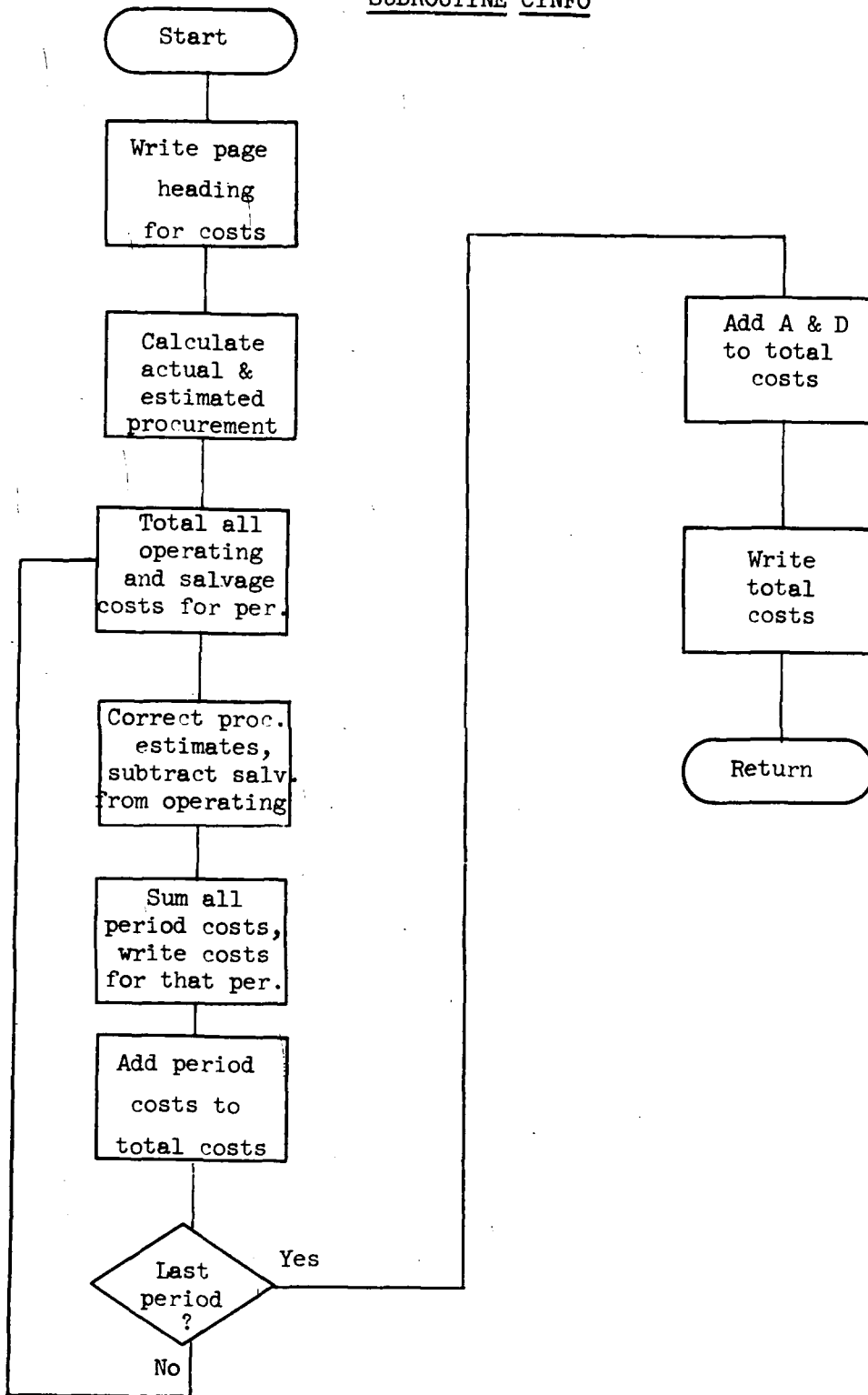


SUBROUTINE VALUES

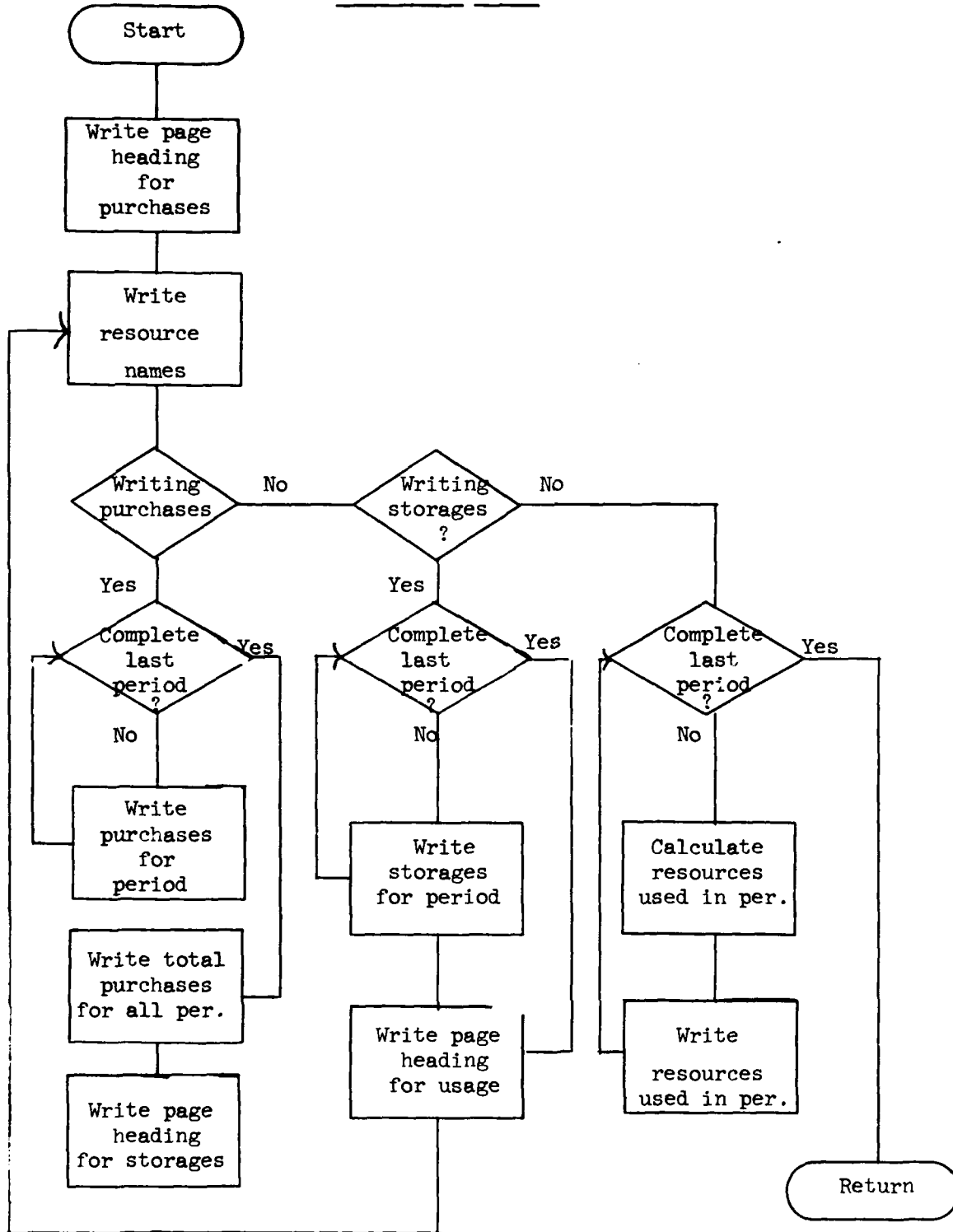


* This is the same routine as used in the matrix generator; its documentation is found there.

SUBROUTINE CINFO



SUBROUTINE PINFO



APPENDIX D

PROGRAM LISTING

GENLCP & SUBROUTINES D-2

BBCAV2 & SUBROUTINES D-22

REPGEN & SUBROUTINES D-86

```

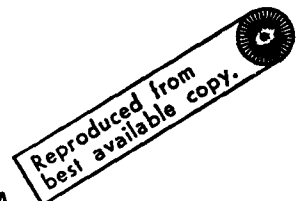
PROGRAM GENLCP(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE4,TAPE9, 10000010
*TAPE7) 10000020
C THIS PROGRAM GENERATES THE MATRIX FILE FOR THE LEAST COST PHASE-IN 10000030
C PROBLEM. 10000040
C 10000050
C THE DIMENSIONS HAVE BEEN SET TO HANDLE 10000060
C MAXIMUM NUMBER OF VEHICLES =7 10000070
C MAXIMUM VEHICLE LIFE (IN YEARS) =25 10000080
C MAXIMUM NUMBER OF YEARS PRIOR TO SY=16 10000090
C MAXIMUM NUMBER OF TASK TABLES =8 10000100
C MAXIMUM NUMBER OF ALTERNATIVES =288 10000110
COMMON /VECSTG/ VNAME(10), C,LEND, VLIFE(10), INH(10,16), 10000120
* VCOST(10,5), NAME(10), COSTS(3,3) 10000130
COMMON /ALTSTG / ALTER(288,9),YAVL(10) 10000140
INTEGER ALTER 10000150
INTEGER FNAME, SY,LY,VNAME,YAVL,VLIFE,YEAR(21) 10000160
DIMENSION BUDG(10) 10000170
DIMENSION NVFHU(20),NL(10),NM(10) 10000180
DIMENSION NAMES(10), AU(16), UB(10),YRINT(20) 10000190
COMMON /TSKSTG/ U(7,288,9),NTSK(9) 10000200
COMMON /PRDSTG/ NPERYR(10,3), NPTASK(10,9), PTASK(10,9) 10000210
DIMENSION IHVN(10) 10000220
DIMENSION NP(288),NM(9),NPM(10) 10000230
C 10000240
DATA(NP(I),I=1,240)/2H01,2H02,2H03,2H04,2H05,2H06,2H07,2H08,2H09, 10000250
*2H10,2H11,2H12,2H13,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22, 10000260
*2H23,2H24,2H25,2H26,2H27,2H28,2H29,2H30,2H31,2H32,2H33,2H34,2H35, 10000270
* 2H36,2H37,2H38,2H39,2H40,2H41,2H42,2H43,2H44,2H45,2H46,2H47,2H48, 10000280
* 2H49,2H50,2H51,2H52,2H53,2H54,2H55,2H56,2H57,2H58,2H59,2H60, 10000290
* 2H61,2H62,2H63,2H64,2H65,2H66,2H67,2H68,2H69,2H70,2H71,2H72, 10000300
* 2H73,2H74,2H75,2H76,2H77,2H78,2H79,2H80,2H81,2H82,2H83,2H84, 10000310
* 2H85,2H86,2H87,2H88,2H89,2H90,2H91,2H92,2H93,2H94,2H95,2H96, 10000320
* 2H97,2H98,2H99,2HA0,2HA1,2HA2,2HA3,2HA4,2HA5,2HA6,2HA7,2HA8, 10000330
* 2HA9,2H90,2H91,2HR2,2HR3,2HR4,2HR5,2HR6,2HR7,2HR8,2HR9,2HC0, 10000340
* 2HC1,2HC2,2HC3,2HC4,2HC5,2HC6,2HC7,2HC8,2HC9,2HD0,2HD1,2HD2, 10000350
* 2HD3,2HD4,2HD5,2HD6,2HD7,2HD8,2HD9,2HE0,2HE1,2HE2,2HE3,2HE4, 10000360
* 2HE5,2HE6,2HE7,2HE8,2HE9,2HF0,2HF1,2HF2,2HF3,2HF4,2HE5,2HF6, 10000370
* 2HF7,2HF8,2HF9,2HG0,2HG1,2HG2,2HG3,2HG4,2HG5,2HG6,2HG7,2HG8, 10000380
* 2HG9,2HM0,2HM1,2HM2,2HM3,2HM4,2HM5,2HM6,2HM7,2HM8,2HM9,2HJ0, 10000390
* 2HJ1,2HJ2,2HJ3,2HJ4,2HJ5,2HJ6,2HJ7,2HJ8,2HJ9,2HK0,2HK1,2HK2, 10000400
* 2HK3,2HK4,2HK5,2HK6,2HK7,2HK8,2HK9,2HL0,2HL1,2HL2,2HL3,2HL4, 10000410
* 2HL5,2HL6,2HL7,2HL8,2HL9,2HM0,2HM1,2HM2,2HM3,2HM4,2HM5,2HM6, 10000420
* 2HM7,2HM8,2HM9,2HN0,2HN1,2HN2,2HN3,2HN4,2HN5,2HN6,2HN7,2HN8, 10000430
* 2HN9,2HP0,2HP1,2HP2,2HP3,2HP4,2HP5,2HP6,2HP7,2HP8,2HP9,2HQ0/ 10000440
DATA(NP(I),I=241,288)/2HQ1,2HQ2,2HQ3,2HQ4,2HQ5,2HQ6,2HQ7,2HQ8, 10000450
*2HQ9,2HR0,2HR1,2HR2,2HR3,2HR4,2HR5,2HR6,2HR7,2HR8,2HR9,2HT0,2HT1, 10000460
*2HT2,2HT3,2HT4,2HT5,2HT6,2HT7,2HT8,2HT9,2HU0,2HU1,2HU2,2HU3,2HU4, 10000470
*2HU5,2HU6,2HU7,2HU8,2HU9,2HW0,2HW1,2HW2,2HW3,2HW4,2HW5,2HW6, 10000480
*2HW7,2HW8/ 10000490
DATA NM/2HM1,2HM2,2HM3,2HM4,2HM5,2HM6,2HM7,2HM8,2HM9/ 10000500
DATA NZ/2HM0/ 10000510
DATA SX,SW,SP,SS,SR,SG/1HX,1HW,1HP,1HS,1HB,1HG/ 10000520
DATA IVT,ITT,IPT,IED /8HVEHICLE ,8HTASK ,8HPERIOD , 10000530
* 8HNOTABLE / 10000540
ONE=1.0 10000550
ONEM=-1.0 10000560
C 10000570

```

```

00 3 I=1,7
00 2 J=1,299
00 1 K=1,9
U(T,J,K)=0.0
1 CONTINUE
2 CONTINUE
3 CONTINUE
C THE FIRST DATA CARD CONTAINS 1A. THE FILENAME TO BE USED =FNAME
C 13. THE STARTING YEAR (OR DECISION YEAR)
C 1C. THE LAST YEAR = LY =SY
C 2A. THE NUMBER OF VEHICLES = NV
C 2B. THE NUMBER OF TASKS = NT
C 2C. THE NUMBER OF PERIODS = NPP
C 3A. -P(I-1) PARAMETER
READ(5,100) FNAME, SY,LY, NV,NT,NPP,I7P,IM1
1010 FORMAT(A9,2X,6I5)
WRITE(6,1010) FNAME, SY,LY, NV,NT,NPP
1015 FORMAT(50H GENERATING THE MATRIX FOR THE LEAST COST PHASE-IN PROB
*LEM //11H FILENAME= ,A9,16H STARTING YEAR =,I5,12H LAST YEAR =,I5,1000760
*//11H WILL INPUT, I7,20H VEHICLE TABLES, AND, I3,16H TASK TABLE, AND 10000770
*, I3,15H PERIOD TABLES. )
1000770
NVP=1
NTP=0
NPT=0
NIV=0
NINHP=0
C READ TITLE OF NEXT TABLE
10 READ(5,1000) ITABLE
C DECIDE ON THE TYPE OF TABLE AND GO READ ITS DATA
11 IF(ITABLE .EQ. IVT) GO TO 20
IF(ITABLE .EQ. ITT) GO TO 40
IF(ITABLE .EQ. IPT) GO TO 60
IF(ITABLE .EQ. IFD) GO TO 100
C THE TABLE NAME IS NOT RECOGNIZED, THERE IS AN INPUT ERROR
WRITE(6,1020) ITABLE
1020 FORMAT(1X,A9,60H IS NOT A TABLE NAME, INPUT ERROR. EXECUTION IS TE
*MINATED. )
STOP 1
20 WRITE(6,1030)
1030 FORMAT(30H0 READING IN A VEHICLE TABLE )
NVP=NVP+1
READ(5,1040) VNAME(NVP), YAVL(NVP), VLIFE(NVP)
WRITE(6,1050) VNAME(NVP), YAVL(NVP), VLIFE(NVP)
1040 FORMAT(A9,1X,I4,6X,I2)
1050 FORMAT(1X,A9,2X,2I10)
C VNAME=NAME OF VEHICLE, YAVL= 1ST YEAR VEHICLE AVAILBLE,
C VLIFE= MAXIMUM LIFE OF VEHICLE IN YEARS.
C
C IF THIS VEHICLE WAS AVAILBLE BEFORE THE STARTING YEAR, THEN READ IN
C SIZE OF INHERITED FLEET. BY YEAR BUILT.
NU=SY - YAVL(NVP)
IF(NU .LE. 0) GO TO 25
NTV=NIV+1
IA=1
IR=9
23 READ(5,1060) (INH(NVP,I), I=IA,IR)
1060 FORMAT(AI10)
IF(IR .GE. NU) GO TO 25
10000590
10000590
10000600
10000610
10000620
10000630
10000640
10000650
10000660
10000670
10000680
10000690
10000700
10000710
10000720
10000730
10000740
10000750
10000760
10000770
10000780
10000790
10000800
10000810
10000820
10000830
10000840
10000850
10000860
10000870
10000880
10000890
10000900
10000910
10000920
10000930
10000940
10000950
10000960
10000970
10000980
10000990
10001000
10001010
10001020
10001030
10001040
10001050
10001060
10001070
10001080
10001090
10001100
10001110
10001120
10001130
10001140
10001150

```



IA=IB+1	10001160
IB=IB+8	10001170
GO TO 23	10001180
25 READ (5,1070) (VCOST(NVR,I),I=1,5)	10001190
1070 FORMAT (5F10.2)	10001200
C VCONST(NVR,1) .GT. 1.0E30 INDICATES THIS VEHICLE IS NOT AVAILABLE FOR	10001210
C PURCHASE.	10001220
GO TO 10	10001230
C	10001240
C IT IS ASSUMED THAT ALL VEHICLE TABLES ARE INPUTED FIRST.	10001250
43 WRITE(6,1090)	10001260
1090 FORMAT(25H0 PEADING IN A TASK TABLE)	10001270
NTP=NTR+1	10001280
READ(5,1090) IDT,NU,NA	10001290
WRITE(6,1090) IDT,NU,NA	10001300
1090 FORMAT(3I10)	10001310
C IDT=TASK IDENTIFICATION NUMBER, NU=NUMBER OF VEHICLES,	10001320
C NA=NUMBER OF ALTERNATIVES	10001330
NTSK(IDT)=NA	10001340
IA=1	10001350
IB=8	10001360
43 READ(5,1100) (NAMES(I),I=IA,IB)	10001370
1100 FORMAT(9(A8,2X))	10001380
IF(IB .GE. NU) GO TO 45	10001390
IA=IB+1	10001400
IB=IB+8	10001410
GO TO 43	10001420
45 DO 47 I=1,NU	10001430
DO 46 J=1,NVP	10001440
IF(VNAME(J) .NE. NAMES(I)) GO TO 46	10001450
NAMES(I)=J	10001460
GO TO 47	10001470
46 CONTINUE	10001480
WRITE(6,1110) NAMES(I)	10001490
1110 FORMAT(15H0 VEHICLE NAME ,A8,60H NOT DEFINED IN A VEHICLE TABLE, E	10001500
*EXECUTION TERMINATED.)	10001510
STOP 2	10001520
47 CONTINUE	10001530
C	10001540
C NOW READ IN U(J,K,L), NUMBER OF VEHICLES OF TYPE J REQUIRED TO	10001550
C PERFORM TASK L WITH ALTERNATIVE K.	10001560
DO 55 K=1,NA	10001570
IA=1	10001580
IB=8	10001590
48 READ(5,1120) (AU(I),I=IA,IB)	10001600
1120 FORMAT(8F10.0)	10001610
IF(IB .GE. NU) GO TO 49	10001620
IA=IB+1	10001630
IB=IB+8	10001640
GO TO 48	10001650
49 DO 50 I=1,NU	10001660
J=NAMES(I)	10001670
U(J,K,IDT)=AU(I)	10001680
50 CONTINUE	10001690
55 CONTINUE	10001700
56 READ(5,1000) ITABLE	10001710
GO TO 11	10001720
C	10001730

60	WRITE(6,1130)	10001740
1130	FORMAT(30H0 READING IN A PERIOD TABLE)	10001750
C	THE PERIOD TABLES ARE EXPECTED IN CHRONOLOGICAL ORDER.	10001760
	NPT=NPT+1	10001770
	READ(5,1140) (NPERY0(NPT,I),I=1,2),RUDG(NPT)	10001780
1140	FORMAT(I4,I5,3X,F8.2)	10001790
	WRITE(6,1150) (NPERYR(NPT,I),I=1,2)	10001800
1150	FORMAT(2I5)	10001810
	IF(NPT.EQ.1) GO TO 61	10001820
	IF(NPERY0(NPT-1,2)+1.EQ.NPERYR(NPT,1)) GO TO 61	10001830
	WRITE(6,1155)	10001840
1155	FORMAT(35H THE PERIOD TABLES ARE OUT OF ORDER)	10001850
	STOP 3	10001860
61	IF(SY.GT.NPERY0(NPT,1)) GO TO 70	10001870
	READ(5,1158) NU,YPRINT(NPT)	10001880
1158	FORMAT(I10,F10.7)	10001890
C	ALL THE TASKS ARE SCALED BY THE FACTOR YPRINT(NPT) IN THE PERIOD NPT	10001900
	NPERYR(NPT,3)=NU	10001910
	IA=1	10001920
	IR=A	10001930
	NA=NPT-NINHP	10001940
63	READ(5,1160) ((NPTASK(NA,I),PTASK(NA,I)),I=IA,IP)	10001950
1160	FORMAT(9(I5,F5.0))	10001960
	IF(IP.GE.NU) GO TO 56	10001970
	IA=IA+1	10001980
	IR=IR+A	10001990
	GO TO 63	10002000
70	NINHP=NINHP+1	10002010
	GO TO 56	10002020
C		10002030
C	ALL TABLE HAVE BEEN READ IN. NOW PROCESS THEM TO BE ABLE TO GENERATE	10002040
C	THE MATRIX.	10002050
C	FIRST CHECK TO DETERMINE IF THE EXPECTED NUMBER OF TABLE WERE INPUTED	10002060
105	IF((NV.EQ.NVP).AND.(NT.EQ.NTP).AND.(NPP.EQ.NPT))GO TO	10002070
	WRITE(6,1170)	10002080
1170	FORMAT(71H WARNING-THE NUMBER OF TABLE ACTUALLY INPUT WAS NOT THE	10002090
	*EXPECT NUMBER.)	10002100
C		10002110
C	ORDER THE VECHILES SO THE ARE IN DEFENDING ORDER OF R+D COST.	10002120
105	NPD=0	10002130
	DO 107 I=1,NVP	10002140
	NAMFN(I)=I	10002150
	IF(VCOST(I,3).LE.0.0) GO TO 107	10002160
	NPD=NPD+1	10002170
107	CONTINUE	10002180
	IF(NPD.EQ.0) GO TO 151	10002190
	NV=NVP-1	10002200
	DO 120 II=1,NV	10002210
	I=NAMFN(II)	10002220
	IMAX=I	10002230
	IP1=II+1	10002240
	CMAX=VCOST(I,3)	10002250
	DO 110 JJ=IP1,NVP	10002260
	J=NAMFN(JJ)	10002270
	IF(CMAX.GE.VCOST(J,3))GO TO 110	10002280
	IMAX=JJ	10002290
	CMAX=VCOST(J,3)	10002300
110	CONTINUE	10002310

```

J=NAMEN(TMAX) 10002320
NAMEN(IMAX)=NAMEN(II) 10002330
NAMEN(II)=J 10002340
IF(VCOST(J,3) .GT. 0.0) GO TO 125 10002350
GO TO 125 10002360
120 CONTINUE 10002370
C 10002380
C NOW DETERMINE IF FOR ANY R+D VEHICLE ITS DEVELOPMENT IS NOT OPTIONAL. 10002390
C IT IS ASSUMED ALL TASKS ARE PREFORMED DURING SOME PERIOD AND 10002400
C THE TASKS HAVE BEEN NUMBERED SEQUENTIALLY. 10002410
125 DO 140 T=1,NRD 10002420
    NAMES(I)=1 10002430
    J=NAMEN(T) 10002440
    DO 133 L=1,NTR 10002450
        NA=NTSK(L) 10002460
        DO 130 K=1,NA 10002470
            IF(U(J,K,L) .EQ. 0.0) GO TO 133 10002480
130 CONTINUE 10002490
C FOUND A TASK REQUIRING THAT VEHICLE J BE DEVELOPED 10002500
    NAMES(I)=2 10002510
    GO TO 140 10002520
133 CONTINUE 10002530
140 CONTINUE 10002540
C NAMES(I)=2 IF THE I TH MOST EXPENSIVE R+D COSTING VEHICLE MUST BE 10002550
C DEVELOPED, =1 OTHERWISE 10002560
C IF A VEHICLE MUST BE DEVELOPED TREAT IT AS IF ITS R+D COST =0. 10002570
    NA=C 10002580
    DO 145 I=1,NRD 10002590
        K=NAMES(I) 10002600
        GO TO (145,143),K 10002610
143 L=NAMEN(I) 10002620
    IP1=I+1 10002630
    K=NRD-NA 10002640
    DO 144 IT=IP1,K 10002650
144 NAMEN(II-1)=NAMFN(II) 10002660
    NAMEN(K)=L 10002670
    NA=NA+1 10002680
145 CONTINUE 10002690
    NRD=NRD-NA 10002700
C LIST VEHICLE NAMES AND CORRESPONDING VARIABLE LABELS. 10002710
    WRITE(6,1180) 10002720
1180 FORMAT(33H0 VEHICLE NAME VARIABLE NAME / 8X, 10002730
    * 21HOPTIONAL R+D VEHICLES ) 10002740
    DO 150 II=1,NRD 10002750
        I=NAMEN(II) 10002760
        WRITE(6,1190) VNAME(I), NP(II) 10002770
1190 FORMAT(6X, A8, 5X, 1HX, A2) 10002780
150 CONTINUE 10002790
    IF(NVR .LE. NRD) GO TO 200 10002800
151 WRITE(6,1200) 10002810
1200 FORMAT(13X,14HOETHER VEHICLES ) 10002820
    J=NRD+1 10002830
    DO 155 II=J,NVR 10002840
        I=NAMEN(II) 10002850
        WRITE(6,1190) VNAME(I), NP(II) 10002860
155 CONTINUE 10002870
C 10002880
    NROW=0 10002890

```

	MOQL=C	10002900
C	LIST ROW NAMES	10002910
	20) WRITE(6,1210) FNAME	10002920
121	FORMAT(2H *, 4HNAME,11Y,A9/ 2H *,4HPDWS)	10002930
	WRITE(4,1211) FNAME	10002940
1211	FORMAT (4HNAME,1(Y,A9/ 4HPDWS)	10002950
C		10002960
C		10002970
C	NOW THE ROW LABELS FOR THE MASTER VARIABLES	10002980
	DO 220 I=1,NVP	10002990
	WRITE(6,1240) NP(I)	10003000
124	FORMAT(2H *, 9H F SUMX,A2)	10003010
	WRITE(4,1241) NP(I)	10003020
	NPROW=NPROW+1	10003030
1241	FORMAT (9H F SUMX,A2)	10003040
220	CONTINUE	10003050
C		10003060
C		10003070
C	ROWS FOR PROCUREMENT CONSTRAINTS	10003080
	IA=NPT-NINHP	10003090
	DO 225 I=1,IA	10003100
	WRITE (4,1225) NP(I)	10003110
1225	FORMAT (6H F PC,A2)	10003120
	WRITE (6,1224) NP(I)	10003130
1224	FORMAT (2H *,6H F PC,A2)	10003140
	NPROW=NPROW+1	10003150
225	CONTINUE	10003160
C		10003170
C		10003180
C	NOW THE ROWS ACCOUNTING FOR THE INHERITED FLEET.	10003190
	IF(NINHP .EQ. 0) GO TO 300	10003200
	IR=NINHP - 1	10003210
	IF(IR .EQ. 0) GO TO 240	10003220
	DO 230 I=1,IR	10003230
	J=NINHP - I	10003240
	NPM(I)=NM(J)	10003250
230	CONTINUE	10003260
240	NPM(NINHP) =N7	10003270
	IF(MTV .EQ. 0) GO TO 300	10003280
	NA=NPD + 1	10003290
	JC=1	10003300
	DO 250 JJ=NA,NVP	10003310
	J=NAMEN(JJ)	10003320
	IF(YAVL(J) .GE. SY) GO TO 260	10003330
	THVN(JC)=JJ	10003340
	JC=JC+1	10003350
	DO 250 I=1,NINHP	10003360
	IF(YAVL(J) .GT. NPERYP(I,2)) GO TO 250	10003370
	IA=MAX0(YAVL(J),NPERYP(I,1)) - YAVL(J) + 1	10003380
	IR=NPERYP(I,2) - YAVL(J) + 1	10003390
	DO 245 K=IA,IR	10003400
	IF(INH(J,K) .GT. 0) GO TO 240	10003410
245	CONTINUE	10003420
	GO TO 250	10003430
249	WRITE(6,1250) NP(JJ),NPM(I)	10003440
1250	FORMAT(2H *, 6H F IW,A2,1HP,A2)	10003450
	WRITE(4,1251) NP(JJ),NPM(I)	10003460
	NPROW=NPROW+1	10003470
1251	FORMAT (6H F IW,A2,1HP,A2)	

250	CONTINUE	10003480
260	CONTINUE	10003490
C		10003500
C	NOW PUT OUT THE LABELS FOR THE ROWS FOR EACH PERIOD, THE VEHICLE	10003510
C	BALANCE ROWS FIRST, THEN THE TASK ROWS.	10003520
300	IA=NINHP + 1	10003530
	DO 350 I=IA,NPT	10003540
	I9=I - NINHP	10003550
	NU=NPERYR(I,3)	10003560
	DO 340 JJ=1,NVR	10003570
C	IF THE VEHICLE IS NOT YET AVAILABLE IT CAN NOT BE USED.	10003590
	J=NAMEN(JJ)	10003580
	IF(YAVL(J) .GT. NPERYR(I,2)) GO TO 340	10003600
C	MAKE SURE THE VEHICLE IS USED	10003610
	DO 320 K=1,NU	10003620
	KT=NPTASK(I9,K)	10003630
	NA=NTSK(KT)	10003640
	DO 310 K2=1,NA	10003650
	IF(U(J,K2,KT) .NE. 0.0) GO TO 330	10003660
310	CONTINUE	10003670
320	CONTINUE	10003680
	GO TO 340	10003690
330	WRITE(6,1260) NP(JJ), NP(I9)	10003700
1260	FORMAT(2H *,5H E X,A2,1HP,A2)	10003710
	WRITE(4,1261) NP(JJ), NP(I9)	10003720
	NROW=NROW+1	10003730
1261	FORMAT (5H E X,A2,1HP,A2)	10003740
340	CONTINUE	10003750
	DO 345 K=1,NU	10003760
	KT=NPTASK(I9,K)	10003770
	WRITE(6,1270) NP(KT), NP(I9)	10003780
1270	FORMAT(2H *, 5H E T,A2,1HP,A2)	10003790
	WRITE(4,1271) NP(KT), NP(I9)	10003800
	NROW=NROW+1	10003810
1271	FORMAT (5H E T,A2,1HP,A2)	10003820
345	CONTINUE	10003830
350	CONTINUE	10003840
C		10003850
C	COMPUTE UPPER BOUNDS	10003860
	DO 390 I1=1,NVR	10003870
	UB(IT)=0.0	10003880
	I2=NAMFN(I1)	10003890
	IA=NTNHP+1	10003900
	DO 390 I=IA,NPT	10003910
	NU=NPERYR(I,3)	10003920
	I1=I - NTNHP	10003930
	IF (YAVL(I2) .GT. NPERYR(I,2)) GO TO 380	10003940
	DO 375 J=1,NU	10003950
	JJ=NPTASK(I1,J)	10003960
	TF= PTASK(I1,J)	10003970
	NA=NTSK(JJ)	10003980
	UMAX=0.0	10003990
	DO 370 K=1,NA	10004000
	IF(UMAX .GT. U(I2,K,JJ)) GO TO 370	10004010
	UMAX=U(I2,K,JJ)	10004020
370	CONTINUE	10004030
	UB(I1)=UB(I1) - TF*UMAX*YRINT(I)	10004040
375	CONTINUE	10004050

380	CONTINUE		10004160
390	CONTINUE		10004170
	WRITE(6,1220)		10004180
1220	FORMAT(2H *, 4H N COST)		10004190
	WRITE(4,1221)		10004100
	NRQW=NRQW+1		10004110
1221	FORMAT (4H N COST)		10004120
C			10004130
	WRITE(6,1280)		10004140
1280	FORMAT(2H *, 7HCOLUMNS)		10004150
	WRITE(6,1290)		10004160
1290	FORMAT(2H *, 4X, *(PARTIAL LISTING)*)		10004170
	WRITE(4,1291)		10004180
1291	FORMAT (7HCOLUMNS)		10004190
C	NOW GENERATE THE MATRIX ELEMENTS.		10004200
C			10004210
C	THE XNM COLUMNS.		10004220
	DO 420 I=1, NVP		10004230
	IT=NAMEN(I)		10004240
	WRITE(6,1300) NP(I), NP(I), ONE		10004250
1300	FORMAT(2H *, 4X, 1HX, A2, 7X,	4HSUMX, A2, 4X, F12.4)	10004260
	WRITE(4,1301) NP(I), NP(I), ONE		10004270
	MCOL=MCOL+1		10004280
1301	FORMAT (4X, 1HX, A2, 7X,	4HSUMX, A2, 4X, F12.4)	10004290
420	CONTINUE		10004300
C			10004310
C	THE PNM COLUMNS		10004320
C			10004330
	IA=NOT-MINHP		10004340
	DO 430 I=1, IA		10004350
	WRITE (4,1311) NP(I), NP(I), ONE		10004360
1311	FORMAT (4X, 1HP, A2, 7X, 2HP, A2, 6X, F12.4)		10004370
	WRITE (6,1310) NP(I), NP(I), ONE		10004380
1310	FORMAT (2H *, 4X, 1HP, A2, 7X, 2HP, A2, 6X, F12.4)		10004390
	MCOL=MCOL+1		10004400
	IF (I.EQ. IA) GO TO 430		10004410
	IF (I.ZPLM1.EQ.1) GO TO 430		10004420
	WRITE (4,1313) NP(I), NP(I+1), ONE		10004430
	WRITE (6,1312) NP(I), NP(I+1), ONE		10004440
1312	FORMAT (2H *, 4X, 1HP, A2, 7X, 2HP, A2, 6X, F12.4)		10004450
1313	FORMAT (4X, 1HP, A2, 7X, 2HP, A2, 6X, F12.4)		10004460
430	CONTINUE		10004470
C			10004480
C	GENERATE THE WJLLMM COLUMNS		10004490
C			10004500
440	IF (NTV .EQ. 0) GO TO 490		10004510
	DO 470 IT=1, NTV		10004520
	JJ=THV(IT)		10004530
	J=NAMEN(JJ)		10004540
	CALL YPCOST(J)		10004550
	DO 460 I=1, NTNHP		10004560
	MAXL=VLTFE(J)		10004570
	IF (YAVL(J) .GT. NPERYP(I, 2)) GO TO 460		10004580
C	IT IS ASSUMED ALL THE VEHICLES INHERITED FROM A PERIOD WERE PURCHASED		10004590
C	IN THE FIRST YEAR OF THE PERIOD.		10004600
	IA=MAX0(YAVL(J), NPERYP(I, 1)) - YAVL(J) + 1		10004610
	IB=NPERYP(I, 2) - YAVL(J) + 1		10004620
	DO 445 K=IA, IB		10004630

Reproduced from
best available copy.

	IF(TNH(J,K) .GT. 0) GO TO 448	10004640
445	CONTINUE	10004650
	GO TO 460	10004660
448	NAGE=SY-NPERYR(I,1)	10004670
	C= COSTS(NAGE,2)	10004680
	LIFER=MAXL-NAGE	10004690
	IF(C .EQ. 0.0) GO TO 449	10004700
	C=-C	10004710
	WRITE(6,1330) NP(JJ),NPM(I),NZ, C	10004720
1330	FORMAT(2H *,4X,1HW,A2,A2,A2,3X,4HCOST,6X,F12.4)	10004730
	WRITE(4,1331) NP(JJ),NPM(I),NZ, C	10004740
1331	FORMAT(4X,1HW,A2,A2,A2,3X,4HCOST,6X,F12.4)	10004750
449	WRITE(6,1340) NP(JJ),NPM(I),NZ, NP(JJ),NPM(I),ONE	10004760
1340	FORMAT(2H *,4X,1HW,A2,A2,A2,3X,2HIW,A2,1HP,A2,3X,F12.4)	10004770
	WRITE(4,1341) NP(JJ),NPM(I),NZ, NP(JJ),NPM(I),ONE	10004780
	MCOL=MCOL+1	10004790
1341	FORMAT(4X,1HW,A2,A2,A2,3X,2HIW,A2,1HP,A2,3X,F12.4)	10004800
	IA=NINHP+1	10004810
	DO 455 K=IA,NPT	10004820
C	MAKE SURE THE VECILF IS USED	10004830
	KY=K-NINHP	10004840
	NU=NPERYR(K,3)	10004850
	DO 451 KK=1,NU	10004860
	KT=NPTASK(KY,KK)	10004870
	NA=NTSK(KT)	10004880
	DO 450 K2=1,NA	10004890
	IF(U(J,K2,KT) .NE. 0.0) GO TO 4511	10004900
450	CONTINUE	10004910
451	CONTINUE	10004920
	GO TO 455	10004930
4511	IF(SY+LIFER .LE. NPERYR(K,1)) GO TO 460	10004940
	IY=NPERYR(K,2)-NPERYR(I,1)+1	10004950
	IX=NPERYR(K,2) -SY + 1	10004960
	C=-COSTS(IY,2)	10004970
	IF(K .EQ. NPT) C=-COSTS(IY,3)	10004980
	DO 452 KK=1,IX	10004990
	KKK=KK+NAGE	10005000
	C= C + COSTS(KKK,1)/VCOST(J,4)**KK	10005010
452	CONTINUE	10005020
	WRITE(6,1330) NP(JJ),NPM(I),NP(KY), C	10005030
	WRITE(4,1331) NP(JJ),NPM(I),NP(KY), C	10005040
	WRITE(6,1340) NP(JJ),NPM(I),NP(KY), NP(JJ),NPM(I), ONE	10005050
	WRITE(4,1341) NP(JJ),NPM(I),NP(KY), NP(JJ),NPM(I), ONE	10005060
	MCOL=MCOL+1	10005070
	C=1.0	10005080
	ALPHA=VCOST(J,4)	10005090
	LLL3=0	10005100
	DO 4521 L3=IA,K	10005110
	L4=L3 - NINHP	10005120
	C=-C	10005130
	WRITE(6,1350) NP(JJ),NPM(I),NP(KY), NP(JJ),NP(L4), C	10005140
1350	FORMAT(2H *,4X,1HW,A2,A2,A2,3X,1HX,A2,1HP,A2,4X,F12.4)	10005150
	WRITE(4,1351) NP(JJ),NPM(I),NP(KY), NP(JJ),NP(L4), C	10005160
1351	FORMAT(4X,1HW,A2,A2,A2,3X,1HX,A2,1HP,A2,4X,F12.4)	10005170
	LLL3=LLL3 + (NPERYR(L3,2)-NPERYR(L3,1)) + 1	10005180
	C=ALPHA**LLL3	10005190
4521	CONTINUE	10005200
455	CONTINUE	10005210

460	CONTINUE	10005220
470	CONTINUE	10005230
C		10005240
C	GENERATE THE P, X, AND C COLUMNS FOR EACH PERIOD	10005250
480	IA=MINHPT + 1	10005260
	DO 490 LL=IA,NPT	10005270
C	IF YPINT(LL).EQ. 1... IT IS ASSUMED ALL THE VEHICLES USED ARE	10005280
C	AVAILABLE. HENCE NO CHECK IS MADE.	10005290
	IF (YPINT(LL).EQ. 1.) GO TO 491	10005300
	NVP=NDFRYP(LL,2)	10005310
C	THE SUBROUTINE YINTERP SETS THE ARRAY ALTER TO INDICATE THE	10005320
C	ALTERNATIVES THAT ARE NOT AVAILABLE FOR USE IN PERIOD LL.	10005330
C	IF ALTER(K,J)=C THEN ALTERNATIVE J OF TASK K IS NOT AVAILABLE FOR	10005340
C	USE.	10005350
	CALL YINTERP (NVP,NT2,NVP)	10005360
491	L=LL-NTHPT	10005370
	DO 499 J=1,NVP	10005380
C		10005400
C	GENERATE THE PIKKLL COLUMNS	10005410
499	MVEHU(J)=1	10005390
	NU=NDFRYP(LL,3)	10005420
	DO 500 IT=1,NU	10005430
	ID=NOTASK(L,IT)	10005440
	KA=NTSK(ID)	10005450
	KA=0	10005460
	DO 510 KK=1,NA	10005470
	IF (YPINT(LL).EQ. 1.) GO TO 491	10005480
	IF (ALTER(KK,IT).EQ. 0) GO TO 510	10005490
491	KA=KA+1	10005500
	DO 500 JJ=1,NVP	10005510
	J=NAMEN(JJ)	10005520
	IF (U(J,KK,IT).EQ. 0.0) GO TO 500	10005530
	IF (YAVL(J).GT.NDFRYP(LL,2)) GO TO 500	10005540
	MVEHU(JJ)=2	10005550
	C=PTASK(L,IT)*U(J,KK,IT)*YPINT(LL)	10005560
	WRITE(4,1351) NP(ID),NP(KA),NP(L), NP(JJ),NP(L), C	10005570
1361	FORMAT (4X,1HP,3A2,3X, 1HX,A2,1HP,A2, 4X, F12.4)	10005580
	IF (LL.NE.5) GO TO 510	10005590
	IF (KA.GT.10) GO TO 500	10005600
	WRITE(5,1350) NP(ID),NP(KA),NP(L), NP(JJ),NP(L), C	10005610
1360	FORMAT(2H *,4X,1HP,3A2, 3X, 1HX,A2,1HP,A2, 4X, F12.4)	10005620
500	CONTINUE	10005630
	WRITE(4,1371) NP(ID),NP(KA),NP(L), NP(ID),NP(L), ONE	10005640
	NCOL=NCOL+1	10005650
1371	FORMAT (4X,1HP,3A2, 3X, 1HT,A2,1HP,A2, 4X, F12.4)	10005660
	IF (LL.NE.5) GO TO 510	10005670
	IF (KA.GT.10) GO TO 510	10005680
	WRITE(5,1370) NP(ID),NP(KA),NP(L), NP(ID),NP(L), ONE	10005690
1370	FORMAT(2H *,4X,1HP,3A2, 3X, 1HT,A2,1HP,A2, 4X, F12.4)	10005700
510	CONTINUE	10005710
520	CONTINUE	10005720
	LEND=NDFRYP(LL,2)-NDFRYP(LL,1)+1	10005730
C		10005740
C	NOW GENERATE THE XJLMM COLUMNS	10005750
	DO 570 JJ=1,NVP	10005760
	IS=MVEHU(JJ)	10005770
	GO TO(573,525),IS	10005780
C	IS=2 INDICATES VEHICLE JJ IS USED IN PERIOD L	10005790

525	J=NAMEN(JJ)	10005800
	CALL YRCOST(J)	10005810
	C=0	10005820
	DO 526 IS=1,LENP	10005830
526	C=C + COSTS(TS,1)	10005840
	IF(NPERYR(LL,2).EQ. LY) GO TO 529	10005850
	C=C - COSTS(LENP,2)	10005860
	GO TO 527	10005870
528	C=C - COSTS(LENP,3)	10005880
527	WRITE(4,1391) NP(JJ),NP(L),NP(L), NP(JJ), ONE	10005890
1391	FORMAT (4X,1HX,3A2, 3X, 4HSUMX,A2, 4X, F12.4)	10005900
	IF (LL.NE.5) GO TO 530	10005910
	WRITE(6,1390) NP(JJ),NP(L),NP(L), NP(JJ), ONE	10005920
1390	FORMAT(2H *,4X,1HX,3A2, 3X, 4HSUMX,A2, 4X, F12.4)	10005930
530	WRITE(4,1401) NP(JJ),NP(L), NP(L), NP(JJ),NP(L), ONEM	10005940
1401	FORMAT (4X,1HX,3A2, 3X, 1HX,A2,1HP,A2,4X, F12.4)	10005950
	IF (LL.NE.5) GO TO 531	10005960
	WRITE(6,1400) NP(JJ),NP(L), NP(L), NP(JJ),NP(L), ONEM	10005970
1400	FORMAT(2H *,4X,1HX,3A2, 3X, 1HX,A2,1HP,A2,4X, F12.4)	10005980
531	IF (LL.NE.5) GO TO 529	10005990
	WRITE (6,1384) NP(JJ),NP(L),NP(L),NP(L),VCOST(J,5)	10006000
1384	FORMAT (2H *,4X,1HX,3A2,3X,2HPC,A2,6X,F12.4)	10006010
	WRITE(6,1380) NP(JJ),NP(L),NP(L), C	10006020
1380	FORMAT(2H *,4X,1HX,3A2, 3X, 4HCOST,6X,F12.4)	10006030
529	WRITE (4,1385) NP(JJ),NP(L),NP(L),NP(L),VCOST(J,5)	10006040
1385	FORMAT (4X,1HX,3A2,3X,2HPC,A2,6X,F12.4)	10006050
	WRITE (4,1381) NP(JJ),NP(L),NP(L),C	10006060
	MCOL=MCOL+1	10006070
1381	FORMAT (4X,1HX,3A2, 3X, 4HCOST,6X,F12.4)	10006080
	LP1=LL + 1	10006090
	IF (LP1 .GT. NPT) GO TO 570	10006100
	DO 545 L1=LP1,NPT	10006110
C		10006130
C	MAKE SUPE VEHICLE JJ IS USED IN PERIOD L1.	10006140
	IF(VLIFE(J) .LE. (NPERYR(L1,1) - NPERYR(LL,1))) GO TO 545	10006120
	NU=NPERYR(L1,3)	10006150
	DO 540 II=1,NU	10006160
	L2=L1-NINHP	10006170
	TD=NPTASK(L,II)	10006180
	NA=NTSK(ID)	10006190
	DO 535 KK=1,NA	10006200
	IF(U(J,KK,ID) .NE. 0.0) GO TO 5411	10006210
535	CONTINUE	10006220
540	CONTINUE	10006230
	GO TO 545	10006240
5411	ALPHA=VCOST(J,4)	10006250
	C=1.0	10006260
	LLL3=0	10006270
	DO 5442 L3=LL,L1	10006280
	C=-C	10006290
	L4=L3-NINHP	10006300
	IF (LL.NE.5) GO TO 5443	10006310
	WRITE(6,1400) NP(JJ),NP(L),NP(L2), NP(JJ),NP(L4), C	10006320
5443	WRITE(4,1401) NP(JJ),NP(L),NP(L2), NP(JJ),NP(L4), C	10006330
	LLL3=LLL3+ (NPERYR(L3,2)-NPERYR(L3,1)) + 1	10006340
	C=ALPHA**LLL3	10006350
5442	CONTINUE	10006360
	C=0	10006370

```

LLL1=NPEPYR(L1,2)-NPEPYR(LL,1) + 1
DO 542 IS=1,LLL1
542 C=C + COSTS(IS,1)
IF(NPEPYR(L1,2) .EQ. LY) GO TO 543
C=C - COSTS(LLL1,2)
GO TO 544
543 C=C - COSTS(LLL1,3)
544 WRITE (4,1385) NP(JJ),NP(L),NP(L2),NP(L),VCOST(J,5)
WRITE (4,1381) NP(JJ),NP(L),NP(L2),C
MCOL=MCOL+1
IF (LL.NE.5) GO TO 5441
WRITE (6,1384) NP(JJ),NP(L),NP(L2),NP(L),VCOST(J,5)
WRITE(6,1380) NP(JJ),NP(L),NP(L2),C
5441 WRITE(4,1331) NP(JJ),NP(L),NP(L2), NP(JJ), ONE
IF (LL.NE.5) GO TO 545
WRITE(6,1390) NP(JJ),NP(L),NP(L2), NP(JJ), ONE
545 CONTINUE
C
C NOW GENERATE THE SJLL COLUMN
C
CALL MOTH(J)
WRITE(4,1411) NP(JJ),NP(L),C
1411 FORMAT (4X,1HS,2A2,5X,4HCOST,6X,F12.4)
MCOL=MCOL+1
WRITE (4,1412) NP(JJ),NP(L),NP(JJ),NP(L),ONE
1412 FORMAT (4X,1HS,2A2,5X,1HX,A2,1HP,A2,4X,F12.4)
IF (LL.NE.5) GO TO 570
WRITE(6,1410) NP(JJ),NP(L),C, NP(JJ),NP(L), ONE
1410 FORMAT(2H *,4X,1HS,2A2,5X,4HCOST,6X,F12.4,3X,1HX,A2,1HP,A2,
* 4X,F12.4)
570 CONTINUE
530 CONTINUE
C
C NOW GENERATE THE RIGHT-HAND-SIDE ELEMENTS
WRITE(6,1420)
1420 FORMAT(2H *,2HPHS)
WRITE(4,1421)
MCOL=MCOL+1
1421 FORMAT (3HPHS)
C
C
C GENERATE THE PHS FOR PROCUREMENT CONSTRAINTS
C
IA=NPT-NINHP
DO 610 I=1,IA
IR=I+NINHP
WRITE (4,1435) NP(I),BUDG(IR)
1435 FORMAT (4X,4HPHS1,6X,2HPC,A2,6X,F12.4)
WRITE (6,1434) NP(I),BUDG(IR)
1434 FORMAT (2H *,4X,4HPHS1,6X,2HPC,A2,6X,F12.4)
610 CONTINUE
C
C GENERATE THE PHS FOR INHERITED FLEET ROWS
615 IF(NIV .EQ. 0) GO TO 650
DO 640 I=1,NIV
JJ=I+VN(I)
J=NAMEN(JJ)
DO 630 I=1,NINHP

```

```

10006380
10006390
10006400
10006410
10006420
10006430
10006440
10006450
10006460
10006470
10006480
10006490
10006500
10006510
10006520
10006530
10006540
10006550
10006560
10006570
10006580
10006590
10006600
10006610
10006620
10006630
10006640
10006650
10006660
10006670
10006680
10006690
10006700
10006710
10006720
10006730
10006740
10006750
10006760
10006770
10006780
10006790
10006800
10006810
10006820
10006830
10006840
10006850
10006860
10006870
10006880
10006890
10006900
10006910
10006920
10006930
10006940
10006950

```

```

IF(YAVL(J) .GT. NPERYR(I,2)) GO TO 630
ISUM=0
IA=MAX0(YAVL(J),NPERYR(I,1)) - YAVL(J) + 1
IB=NPERYR(I,2) - YAVL(J) + 1
DO 620 K=IA,IB
620 ISUM=ISUM + INH(J,K)
IF(ISUM .EQ. 0) GO TO 630
C=FLOAT(ISUM)
WRITE(6,1440) NP(JJ), NPM(I), C
1440 FORMAT(2H *,4X, 4HPHS1,6X, 2HIW,A2,1HP,A2,3X, F12.4)
WRITE(4,1441) NP(JJ), NPM(I), C
1441 FORMAT (4X, 4HPHS1,6X, 2HIW,A2,1HP,A2,3X, F12.4)
630 CONTINUE
640 CONTINUE
C
C NOW GENERATE THE RHS FOR THE TASK ROWS
650 IA=NINHP+1
DO 700 LL=IA,NPT
L=LL-NINHP
NU=NPERYR(LL,3)
DO 690 K=1,NU
KT=NPTASK(L,K)
WRITE(6,1450) NP(KT), NP(L), ONE
1450 FORMAT(2H *,4X, 4HRHS1,6X, 1HT,A2,1HP,A2,4X,F12.4)
WRITE(4,1451) NP(KT), NP(L), ONE
1451 FORMAT (4X, 4HRHS1,6X, 1HT,A2,1HP,A2,4X,F12.4)
690 CONTINUE
700 CONTINUE
WRITE(6,1460)
1460 FORMAT(2H *, 6HENDATA)
WRITE(4,1461)
1461 FORMAT ( 6HENDATA)
END FILE 4
CALL MATFILL(NROW,NCOL,UB,NVR)
WRITE (6,3000) NROW,NCOL,(UB(I),I=1,NVR)
3000 FORMAT (*0 IMPORTANT DATA ITEMS FOR INPUT TO B9CAVLP * /
A * NUMBER OF ROWS (INCLUDING COST) IS *,I4 /
B * NUMBER OF COLUMNS (INCLUDING PHS) IS *,I7 /
C * UPPER BOUNDS FOR VEHICLES IN ORDER FROM X1 THRU XN ARE */
D (1H ,10X,F12.4))
C
C PRODUCE OUTPUT LISTING FOR DOCUMENTATION OF RUN
C
WRITE (6,2010)
WRITE (6,2020)
2010 FORMAT (*1 VEHICLE VARIABLE PURCHASE O AND M R AND D
* RETENTION YEAR FIRST LIFE IN*)
2020 FORMAT (* NAME NAME COST COST COST
* RATE AVAILABLE YEARS*)
IY=SY
C
C LIST VEHICLE VARIABLE NAME, AND COST DATA
C
DO 800 I=1,NVR
II=NAMEN(I)
WRITE (6,2030) VNAME(II),NP(I),(VCOST(II,J),J=1,4),YAVL(II),
*VLIFE(II)
2030 FORMAT (1H0,4X,A2,7X,1HX,A2,4(F9.4,4X),2X,I4,8X,I2)

```

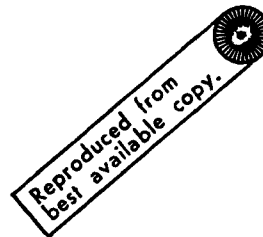
	IF (YAVL(II).LT.IY) IY=YAVL(II)	10007540
	830 CONTINUE	10007550
C		10007560
C	DESCRIBE THE INHERITED FLEET	10007570
C		10007580
	IF (IY.EQ.SY) GO TO 821	10007590
	WRITE (6,2040)	10007600
2040	FORMAT (*- COMPONENTS OF THE INHERITED FLEET*)	10007610
	IF ((SY-IY).GT.20) IY=SY-20	10007620
	DO 810 I=IY,SY	10007630
	II=I-IY+1	10007640
810	YEAR(II)=I	10007650
	INHYP5=SY-IY	10007660
	WRITE (6,2050) (YEAR(I),I=1,INHYP5)	10007670
2050	FORMAT (1H0,20X,20(I5))	10007680
	NA=NVP-NTV+1	10007690
	DO 820 I=1,NVP	10007700
	J=NAMEN(I)	10007710
	IF (YAVL(J).GE.SY) GO TO 820	10007720
	KK=YAVL(J)-IY	10007730
	DO 815 K=1,INHYP5	10007740
	IF (KK.LT.K) GO TO 814	10007750
	YEAR(K)=0	10007760
	GO TO 815	10007770
814	K1=K-KK	10007780
	YEAR(K)=INH(J,K1)	10007790
815	CONTINUE	10007800
	WRITE (6,2060) NP(I), (YEAR(K),K=1,INHYP5)	10007810
2060	FORMAT (15H NUMBER OF X,A2,4X,20(I5))	10007820
	820 CONTINUE	10007830
C		10007840
C	FOR EACH PERIOD, LIST ALL OF THE APPLICABLE TASK MATRICES	10007850
C		10007860
	821 IA=NTNHP+1	10007870
	DO 830 I=IA,NPT	10007880
	WRITE (6,2070) NPDPY(I,1),NPDPY(I,2)	10007890
2070	FORMAT (35H- TASKS REQUIRED IN PERIOD FROM ,I4,9H THROUGH ,I4)	10007900
	M=NPDPY(I,2)	10007910
	DO 845 J=1,M	10007920
	IM=I-NTNHP	10007930
	JJ=NPTASK(IM,J)	10007940
	WRITE (6,2080) NP(JJ),PTASK(IM,J), YPRINT(I)	10007950
2080	FORMAT (1H0,6X,*TASK *,A2,* - PERFORMED BY *,F5.2,* FORCE FLEMEN	10007960
	*T(S), WITH SCALE FACTOR EQUAL *,F5.3)	10007970
	TT=C	10007980
	IF (YPRINT(I).NE.1.0) GO TO 845	10007990
	WRITE (6,2090)	10008000
2090	FORMAT (1H ,6X,1H*)	10008010
C		10008020
C	DETERMINE WHICH VEHICLES ARE USED IN EACH TASK, JJ.....	10008030
C		10008040
C	(I=PERIOD, K=VEHICLE, II=NUMBER OF VEHICLES USED,	10008050
C	KK=NUMBER OF ALTERNATIVES)	10008060
C		10008070
	KK=NTSK(JJ)	10008080
	DO 830 K=1,NVP	10008090
	N=NAMEN(K)	10008100
	DO 829 L=1,KK	10008110

IF (U(N,L,JJ).EQ.0) GO TO 829	10008120
II=II+1	10008130
NL(II)=NP(K)	10008140
NN(II)=N	10008150
NAMES(II)=K	10008160
GO TO 830	10008170
829 CONTINUE	10008180
830 CONTINUE	10008190
WRITE (6,2100) (NL(K),K=1,II)	10008200
2100 FORMAT (1H ,7X,11H* VARIABLE ,10(3X,1HX,A2))	10008210
WRITE (6,2110)	10008220
2110 FORMAT (1H ,8X,9H*****)	10008230
WRITE (6,2120)	10008240
2120 FORMAT (1H ,6X,11HALTERNATIVE)	10008250
C	10008260
C FILL IN TASK MATRIX	10008270
C	10008280
DO 844 L=1,KK	10008290
DO 840 K=1,IT	10008300
N=NN(K)	10008310
GO TO (831,832,833,834,835,836,837,838,839),K	10008320
831 WRITE (6,2131) L,U(N,L,JJ)	10008330
2131 FORMAT (1H ,15X,T2,2X,F5.0)	10008340
GO TO 840	10008350
832 WRITE (6,2132) U(N,L,JJ)	10008360
2132 FORMAT (1H+,25X,F5.0)	10008370
GO TO 840	10008380
833 WRITE (6,2133) U(N,L,JJ)	10008390
2133 FORMAT (1H+,31X,F5.0)	10008400
GO TO 840	10008410
834 WRITE (6,2134) U(N,L,JJ)	10008420
2134 FORMAT (1H+,37X,F5.0)	10008430
GO TO 840	10008440
835 WRITE (6,2135) U(N,L,JJ)	10008450
2135 FORMAT (1H+,43X,F5.0)	10008460
GO TO 840	10008470
836 WRITE (6,2136) U(N,L,JJ)	10008480
2136 FORMAT (1H+,49X,F5.0)	10008490
GO TO 840	10008500
837 WRITE (6,2137) U(N,L,JJ)	10008510
2137 FORMAT (1H+,55X,F5.0)	10008520
GO TO 840	10008530
838 WRITE (6,2138) U(N,L,JJ)	10008540
2138 FORMAT (1H+,61X,F5.0)	10008550
GO TO 840	10008560
839 WRITE (6,2139) U(N,L,JJ)	10008570
2139 FORMAT (1H+,67X,F5.0)	10008580
840 CONTINUE	10008590
844 CONTINUE	10008600
845 CONTINUE	10008610
850 CONTINUE	10008620
STOP	10008630
END	10008640


```

SUBROUTINE MATFILL(N,M,UB,NVR)
DIMENSION RVAL(120),PNAME(120)
DIMENSION IROWTP(100)
DIMENSION UP(1)
DATA IT,II / 1HT,1HI /
DATA C / 7HCOLUMNS /,R / 3RHS /
I=0
J=0
DO 400 K=1,100
400 IROWTP(K)=0
PEWTD 4
WRITE(9,7000) M,N,(UB(I),I=1,NV)
7000 FORMAT(2I6/(6F12.4))
READ(4,4000) DUM1,DUM2
IF (EOF,4) 120,1
1 WRITE(3,4000) DUM1,DUM2
4000 FORMAT (A4,10X,A9)
READ(4,4100) DUM3
4100 FORMAT (A4)
DO 10 I=1,N
READ(4,4200) PNAME(I)
4200 FORMAT (4X,A7)
ENCODE(1,9000,ITEMP) PNAME(I)
IF (ITEMP.EQ.IT.OP.ITEMP.EQ.II) IROWTP(I)=4
9000 FORMAT(A1)
IF (EOF,4) 120,1
10 CONTINUE
READ(4,4300) DUM4
4300 FORMAT (A7)
IF (DUM4.EQ.C)GO TO 20
WRITE(6,4400) DUM4
4400 FORMAT(* INCORRECTLY READ FILE----COLUMNS READ AS *,A7)
RETURN
20 READ(4,4500) CNAME,PTEMP,VAL
4500 FORMAT (4X,A7,3X,A7,3X,F12.4)
WRITE(6,5000)
5000 FORMAT (*1 REFERENCE LIST FOR COLUMN NUMBERS AND NAMES*)
WRITE(6,6100) (IROWTP(K),K=1,N)
WRITE(9,6000) (IROWTP(K),K=1,N)
6000 FORMAT(I12)
6100 FORMAT(1H ,100I1)
L=1
DO 100 J=1,M
GO TO (21,22,23,24,25),L
21 WRITE(6,5100) J,CNAME
5100 FORMAT (1H ,4X,I5,4X,A7)
GO TO 25
22 WRITE(6,5200) J,CNAME
5200 FORMAT (1H+,24X,I5,4X,A7)
GO TO 25
23 WRITE(6,5300) J,CNAME
5300 FORMAT (1H+,44X,I5,4X,A7)
GO TO 25
24 WRITE(6,5400) J,CNAME
5400 FORMAT (1H+,64X,I5,4X,A7)
GO TO 25
25 WRITE(6,5500) J,CNAME

```



```

10008650
10008660
10008670
10008680
10008690
10008700
10008710
10008720
10008730
10008740
10008750
10008760
10008770
10008780
10008790
10008800
10008810
10008820
10008830
10008840
10008850
10008860
10008870
10008880
10008890
10008900
10008910
10008920
10008930
10008940
10008950
10008960
10008970
10008980
10008990
10009000
10009010
10009020
10009030
10009040
10009050
10009060
10009070
10009080
10009090
10009100
10009110
10009120
10009130
10009140
10009150
10009160
10009170
10009180
10009190
10009200
10009210

```

5500	FORMAT (1H+,84X,I5,4X,A7)	10009220
26	L=L+1	10009230
	IF(L.GT.5)L=1	10009240
	WRITE(7,5700) J,CNAME	10009250
5700	FORMAT(I5,4X,A7)	10009260
	DO 30 I=1,N	10009270
30	RVAL(I)=0.0	10009280
40	DO 50 I=1,N	10009290
	IF (RTEMP.NE.RNAME(I)) GO TO 50	10009300
	RVAL(I)=VAL	10009310
	GO TO 60	10009320
50	CONTINUE	10009330
60	IF (J.NE.(M-1)) GO TO 80	10009340
	IF (I.NE.N) GO TO 80	10009350
	READ (4,4600) DUM5	10009360
4600	FORMAT (A3)	10009370
	IF (EOF,4) 120,70	10009380
70	IF (DUM5.EQ.R) GO TO 80	10009390
	WRITE (6,4700) CNAME	10009400
4700	FORMAT(* THE M-1 COLUMN WAS *,A7,* ,UNABLE TO FIND RHS MARK*)	10009410
	RETURN	10009420
80	READ (4,4500) CTEMP,RTEMP,VAL	10009430
	IF (EOF,4) 120,90	10009440
90	IF (CTEMP.EQ.CNAME) GO TO 40	10009450
	CNAME=CTEMP	10009460
	WRITE (9,4800) (RVAL(K),K=1,N)	10009470
4800	FORMAT (F12.4)	10009480
100	CONTINUE	10009490
	END FILE 9	10009500
	END FILE 7	10009510
	RETURN	10009520
120	WRITE (6,4900) J,I	10009530
4900	FORMAT(* REACHED EOF WHILE WRITING COLUMN *,I7,* AND ROW *,I4)	10009540
	RETURN	10009550
	END	10009560

```

SUBROUTINE VPCOST(J)
C A SUBROUTINE TO COMPUTE THE OPERATING, SALVAGE, AND TRUNCATION
C COSTS YEAR BY YEAR. ALSO THE YEARLY MOTHBALLING SAVING IS COMPUTED.
COMMON /VECSTG/ VNAME(10), C,LENP, VLIFE(10), INH(10,16),
* VPCOST(10,5), NAMEN(10), COSTS(30,3)
INTEGER VNAME,VLIFE
C ASSUME THE OPERATING AND MAINTANCE COST INCREASES AT R*100 PER-CENT
C A YEAR (NOT A COMPOUND RATE INCREASE)
R=0.0
C
C LET X= THE 1ST YEAR I. AND M. COST. THEN
C  $X+(1+P)*X+(1+2*R)*X+\dots+(1+9*R)*X=VPCOST(J,2)$ 
X= VPCOST(J,2)/(10.0 + 45.0*R)
C ASSUME NO PERIOD IS LONGER THAN 6 YEARS.
I9=VLIFE(J) +10
DO 10 I=1,I9
COSTS(I,1)=(1.0 + FLOAT(I-1)*R)*X*(VPCOST(J,4)**(I-1))
10 CONTINUE
C
C ASSUME THE SALVAGE VALUE OF A VEHICLE AFTER I YEARS OF SERVICE IS
C (ALPHA)**I *PURCHASE COST.
ALPHA=0.5
Y=VPCOST(J,1)
DO 20 I=1,I9
Y= ALPHA*Y
COSTS(I,2)=Y
20 CONTINUE
C
C ASSUME TRUNCATION AFTER IYEARS OF SERVICE IS
C (VLIFE-I)*(PURCHASE COST)/VLIFE
Y=VPCOST(J,1)/VLIFE(J)
DO 30 I=1,I9
IX=VLIFE(J)-I
IF (IX.LT.0) IX=0
COSTS(I,3)=IX*Y
30 CONTINUE
RETURN
ENTRY MOTH
C ASSUME THE MOTHBALLING SAVING IS R1*100 PER CENT OF THE FIRST YEAR COST-X
R1=0.90
C
C DO 546 IL=1,LENP
C 546 C=C-0.1*R1*VPCOST(J,2)*VPCOST(J,4)**(IL-1)
C C =-X * R1
C =-VPCOST(J,2)/(10.0 + 45.0*R) * R1
RETURN
END
10009630
10009640
10009650
10009660
10009670
10009680
10009690
10009700
10009710
10009720
10009730
10009740
10009750
10009760
10009770
10009780
10009790
10009800
10009810
10009820
10009830
10009840
10009850
10009860
10009870
10009880
10009890
10009900
10009910
10009920
10009930
10009940
10009950
10009960
10009970
10009980
10009990
10009000
10009010
10009020
10009030
10009040
10009050
10009060
10009070
10009080
10009090
10009100
10009110
10009120
10009130
10009140
10009150
10009160
10009170
10009180
10009190
10009200
10009210
10009220
10009230
10009240
10009250
10009260
10009270
10009280
10009290
10009300
10009310
10009320
10009330
10009340
10009350
10009360
10009370
10009380
10009390
10009400
10009410
10009420
10009430
10009440
10009450
10009460
10009470
10009480
10009490
10009500
10009510
10009520
10009530
10009540
10009550
10009560
10009570
10009580
10009590
10009600
10009610
10009620
10009630
10009640
10009650
10009660
10009670
10009680
10009690
10009700
10009710
10009720
10009730
10009740
10009750
10009760
10009770
10009780
10009790
10009800
10009810
10009820
10009830
10009840
10009850
10009860
10009870
10009880
10009890
10009900
10009910
10009920
10009930
10009940
10009950
10009960
10009970
10009980
10009990

```

	SUBROUTINE YINTERP.(NVR,NTR,NYR)	10010040
	COMMON /TSKSTG/ U(7,288,9) ,NTSK(9)	10010050
	COMMON / ALTSTG / ALTER(288,9),YAVL(10)	10010060
	INTEGER ALTER	10010070
	INTEGER JSUB(10),YAVL	10010080
	DO 20 I=1,NTR	10010090
	N=NTSK(I)	10010100
	DO 10 J=1,N	10010110
10	ALTER(J,I)=1	10010120
20	CONTINUE	10010130
	DO 30 I=1,NVR	10010140
	IF (YAVL(I).LE.NYR) GO TO 30	10010150
	IVR=I	10010160
	GO TO 40	10010170
30	CONTINUE	10010180
	RETURN	10010190
40	L=0	10010200
	DO 50 J=IVR,NVR	10010210
	IF (YAVL(J).LE.NYR) GO TO 50	10010220
	L=L+1	10010230
	JSUB(L)=J	10010240
50	CONTINUE	10010250
		10010260
C	THE SET OF VEHICLES WHICH WILL NOT EXIST IN YEAR NYR	10010270
C	HAS BEEN DEFINED ---- NOW WE WILL ORDER THE SET	10010280
C	IN THE REVERSE OF THE ORDER IN WHICH THEY WILL	10010290
C	BE DEVELOPED.....	10010300
C		10010310
	DO 70 I=1,L	10010320
	N=I	10010330
	K=JSUB(I)	10010340
	DO 60 J=N,L	10010350
	M=JSUB(J)	10010360
	IF (YAVL(M).LE.YAVL(K)) GO TO 60	10010370
	JSUB(J)=K	10010380
	JSUB(I)=M	10010390
	K=M	10010400
60	CONTINUE	10010410
70	CONTINUE	10010420
		10010430
C	FOR EACH TASK, WE WILL DEFINE THE SET OF ALTERNATIVES	10010440
C	WHERE THE #NON-EXISTENT# VEHICLES ARE DOING ONLY	10010450
C	THOSE TASKS WHICH ARE THEIR PRIMARY RESPONSIBILITY,	10010460
C	THAT IS, WHERE THE REQUIREMENT FOR THEM IS A MINIMUM.....	10010470
C		10010480
	DO 150 I=1,NTR	10010490
	N=NTSK(I)	10010500
	DO 140 JJ=1,L	10010510
	J=JSUB(JJ)	10010520
	VMIN=9999.	10010530
	DO 100 K=1,N	10010540
	IF (ALTER(K,I).EQ.0) GO TO 100	10010550
	IF (U(J,K,I).LT.VMIN) VMIN=U(J,K,I)	10010560
100	CONTINUE	10010570
	DO 130 K=1,N	10010580
	IF (ALTER(K,I).EQ.0) GO TO 130	10010590
	IF (U(J,K,I).EQ.VMIN) GO TO 130	10010600

ALTP(K,T)=0
130 CONTINUE
140 CONTINUE
150 CONTINUE
RETURN
END

10010610
10010620
10010630
10010640
10010650
10010660

	PROGRAM 9BCAV2 (INPUT, OUTPUT, TAPEA, TAPE1, TAPE2,	20000010
	1 TAPE3, TAPE7, TAPE8, TAPE5=INPUT, TAPE6=OUTPUT,	20000020
	2 TAPF9=TAPFA)	20000030
C		20000040
C	LABELLED COMMON	20000050
	COMMON / CV1 / IP(12), RP(12), TMP(10)	20000060
	COMMON / CV2 / T(100,10), B0(100), BLO(10), ULO(10), CO(10)	20000070
	COMMON / CV3 / M, N, NCF, PHIT, UZ, USP, USM, EKO, MPLUS	20000080
	COMMON / CV4 / IX(110), X(110), IXZ(110), XZ(110), XCON(10), COST	20000090
	COMMON / CV5 / SIGMA(100,4), TSIG, LSTMAX	20000100
	COMMON / CV7 / NPHASE, NF1, CFX, IOPT, NOP, NOPS, NEWXZ	20000110
	COMMON / CV8 / NXPK, XK, NOBOL, EKBL(25)	20000120
	COMMON / CV9 / PSIGL(25), NXBL(25), XNXBL(25), BLIST(25,131)	20000130
	COMMON/THX/TMO, EXT, TITLE(4)	20000140
C		20000150
	INTEGER UB, CI, BV	20000160
	DIMENSION TST0(130), LSTFRE(25)	20000170
	DIMENSION BLT(10), ULT(10), CT(10)	20000180
C		20000190
C		20000200
	7 READ(5,4448) (TITLE(I), I=1,4)	20000210
	IF(EOF,5)1,2	20000220
	1 END FILE 0	20000230
	STOP0003	20000240
	2 CALL PARAMS	20000250
	RP(12)=0.0	20000260
	IP(9)=25	20000270
	NFREF=0	20000280
	CT = 4	20000290
	UB = 3	20000300
	LB = 2	20000310
	BV = 1	20000320
	MNC = (-1) * NCF	20000330
	MNX = (-1) * N	20000340
	EPSI = RP(1)	20000350
	NORA = IP(2)	20000360
	MPLUS=NORA+NCF	20000370
	NOPS = 1	20000380
	NCF4 = NCF * 3 + NORA	20000390
C		20000400
	CALL READIN	20000410
	CALL BOX1	20000420
C		20000430
C	SOLVE 1ST LP-PROBLEM	20000440
C		20000450
	55 CONTINUE	20000460
	US = USP	20000470
	IF(UZ .LT. 0.0) US = USM	20000480
	IF (NOP.GE.IP(12)) GO TO 4444	20000490
	LSTMAX=MAX0(LSTMAX,NOBOL)	20000500
	PMIN=1.E20	20000510
	DO 3000 I=1,NOBOL	20000520
	IF(PSIGL(I).GE.PMIN) GO TO 3000	20000530
	PMIN=PSIGL(I)	20000540
	NMIN=I	20000550
3000	CONTINUE	20000560
	IF(PMIN.LT.US) GO TO 3020	20000570

```

      WRITE(6,3010)
3010  FORMAT(1H-,14HOROPLEM SOLVED)
      GO TO 4446
3020  PSIGL(NMTN)=1.E20
      NPEEF=NPEEF+1
      LSTERR(NPEEF)=NMTN
      NXBK=NXBL(NMTN)
      EKQ=EKRL(NMTN)
      XK=XXBL(NMTN)
      DO 3030 J=1,NCF
      J1=NOPA+J
      J2=NCF+J1
      J3=NCF+J2
      RLO(J)=RLIST(NMTN,J1)
      ULO(J)=RLIST(NMTN,J2)
3030  CO(J)=RLIST(NMTN,J3)
      DO 3040 J=1,NOPA
3040  RO(J)=RLIST(NMTN,J)
      INDTG=1
      RBK = BLO(NXBK)
      URK = ULO(NXBK)

C
C
      DO 10 I = 1,NCF
      TMP(I) = 0.0
      BLT(I) = 0.0
      ULT(I) = 0.0
      OT(I) = 0.0
10  CONTINUE
      DO 30 I = 1,NCF4

C
C  REDEFINE SIGMA FOR 1ST-LP-PR FROM KC-DATA
C
30  TSTG(T) = 0.0
      NMTN=NOPA-1
      DO 552 T=1,NMTN
552  SIGMA(I,RV) = RO(T)
      DO 553 I = 1, NCF
      SIGMA(I,LR) = RLO(T)
      SIGMA(I,UR) = ULO(I)
      SIGMA(I,CI) = CO(T)
553  CONTINUE
      XKSTG = XK

C
      XK = XK - RBK

C
      SIGMA(NXBK,UR) = XK
      TSTG = EKQ
      SIGMA(NOPA,RV) = -TSTG
      RO(NOPA) = -TSTG

C
      BLT(NXBK) = SIGMA(NXBK,LR)
      ULT(NXBK) = XK + BLT(NXBK)

C
C
      RLO(NXBK) = BLT(NXBK)
      ULO(NXBK) = XK
      CALL GETC (NXBK,PLT,ULT,OT)

```

Reproduced from
best available copy.

```

20000580
20000590
20000600
20000610
20000620
20000630
20000640
20000650
20000660
20000670
20000680
20000690
20000700
20000710
20000720
20000730
20000740
20000750
20000760
20000770
20000780
20000790
20000800
20000810
20000820
20000830
20000840
20000850
20000860
20000870
20000880
20000890
20000900
20000910
20000920
20000930
20000940
20000950
20000960
20000970
20000980
20000990
20001000
20001010
20001020
20001030
20001040
20001050
20001060
20001070
20001080
20001090
20001100
20001110
20001120
20001130
20001140
20001150

```

	SIGMA(NXBK,CI) = CT(NXBK)	20001160
	CO(NX9K)=CT(NXBK)	20001170
	NOP = NOP + 1	20001180
	SOLVE K PRIME LP PROBLEM	20001190
C	DO 5555 IND=1,MPLUS	20001200
	X(IND)=0	20001210
5555	IX(IND)=0	20001220
	CALL TABOUT (1)	20001230
	NCF1=NCF	20001240
	NF1=0	20001250
	CALL LP (NOPA,N,NCF1)	20001260
	CALL TABOUT (2)	20001270
	CALL TIMEC	20001280
	COST1 = COST	20001290
	IF(NF1 .NE. 1)GO TO 90	20001300
57	CONTINUE	20001310
	DO 6665 J=1,NCF	20001320
	TMP(J)=0	20001330
6665	XCON(J)=0	20001340
	DO 6666 IND=1,MPLUS	20001350
	IF (IX(IND).GT.NCF .OR. IX(IND).EQ.0) GO TO 6666	20001360
	ICOL=IX(IND)	20001370
	TMP(ICOL)=X(IND)	20001380
	X(IND)=X(IND)+BLO(ICOL)	20001390
	XCON(ICOL)=X(IND)	20001400
	DEFINE X(K) FROM Y(K)	20001410
C	6666 CONTINUE	20001420
	IND=0	20001430
	DO 6677 J=1,NCF	20001440
	IF (BLO(J).EQ. 0.0) GO TO 6677	20001450
	IF (XCON(J).GT.0.0) GO TO 6677	20001460
	XCON(J)=BLO(J)	20001470
	IX(MPLUS-IND)=J	20001480
	X(MPLUS-IND)=BLO(J)	20001490
	IND=IND+1	20001500
6677	CONTINUE	20001510
	RP(12)=COST-TSIG	20001520
	DO 6667 J=1,NCF	20001530
6667	RP(12)=RP(12)-TMP(J)*CO(J)	20001540
	NOPS = NOPS + 1	20001550
	MNY = (-N)	20001560
	CALL TIMEC	20001570
	EVALUATE OBJECTIVE F(X)	20001580
C	CALL GETPHI (MNC,XCON,TMP,PHIT)	20001590
	CALL TIMEC	20001600
C	WRITE(6,573) PHIT	20001610
573	FORMAT(1H0,11HPHI(XANDJ) =,1PE10.7)	20001620
	IF (TP(11).EQ.1)	20001630
	*WRITE (6,575) (IX(I),X(I),I=1,MPLUS)	20001640
575	FORMAT (1H0,5(7H COL ,I4,2H =,F12.4))	20001650
C		20001660
C		20001670
C		20001680
C	IF(PHIT .GE. UZ)GO TO 70	20001690
C	PHIT .LT. UZ FOR 1ST-PROBLEM	20001700
C		20001710
	U7 = PHIT	20001720
	DO 5A I=1,MPLUS	20001730

IXZ(I)=IX(I)	20001740
59 X7(I) = X(I)	20001750
NFWX7=1	20001760
USP = (U7/(1.0 + EPS1))	20001770
USM = (U7/(1.0 - EPS1))	20001780
US = USP	20001790
IF(U7 .LT. 0.0)US = USM	20001800
70 CONTINUE	20001810
IF(COST1 .GE. US)GO TO 95	20001820
CALL NXPRN(XCON, SIGMA, NXP)	20001830
1990 IF(NRFEF.LE.0) GO TO 2000	20001840
NQL=LSTPEF(NRFEF)	20001850
NRFEF=NRFEF-1	20001860
GO TO 2010	20001870
2000 NNRQL=NRQL+1	20001880
NQL=NRQL	20001890
IF(NNRQL.LE.IP(9)) GO TO 2010	20001900
WRITE(6,2020)	20001910
2020 FORMAT(14-,'ALIST SIZE EXCEEDED')	20001920
GO TO 4446	20001930
2010 PSIGL(NQL)=COST	20001940
NXQL(NQL)=NXP	20001950
FXQL(NQL)=TSTG	20001960
XNXQL(NQL)=XCON(NXP)	20001970
DO 2030 J=1,NCF	20001980
J1=NRQA+J	20001990
J2=NCF+J1	20002000
J3=NCF+J2	20002010
ALIST(NQL,J1)=SIGMA(J,LB)	20002020
ALIST(NQL,J2)=SIGMA(J,UB)	20002030
2030 ALIST(NQL,J3)=SIGMA(J,CI)	20002040
DO 2040 J=1,NRQA	20002050
2040 ALIST(NQL,J)=SIGMA(J,RV)	20002060
IF(INDIC.F0.2) GO TO 55	20002070
90 CONTINUE	20002080
INDIC=2	20002090
DO 91 I = 1,NCF	20002100
RLT(I) = 0.0	20002110
ULT(I) = 0.0	20002120
CT(I) = 0.0	20002130
TMP(I) = 0.0	20002140
91 CONTINUE	20002150
C	20002160
C REDEFINE SIGMA FOR 2ND-LP-PB FROM KD-DATA	20002170
C	20002180
DO 95 I=1,NMTN	20002190
SIGMA(I,BV) = B0(I)	20002200
SIGMA(I,RV) = SIGMA(I,RV) - (T(I,NXAK)*YK)	20002210
T(I)=SIGMA(I,RV)	20002220
95 CONTINUE	20002230
DO 96 I = 1,NCF	20002240
SIGMA(I,LR) = PL0(I)	20002250
SIGMA(I,UR) = UL0(I)	20002260
SIGMA(I,CI) = C0(I)	20002270
C	20002280
C	20002290
96 CONTINUE	20002300
C	20002310
	DEFINE LOWER BOUND OF X(K)
	IF P9K = 0
	SET UPPER BOUND OF Y(K).

Reproduced from
 best available copy.

	THIS IS THE ONLY BOUND FOR	20002320
	THIS VARIABLE SENT TO THE LP CODE	20002330
C	BRK2 = BRK + XK	20002340
C	UBK2 = UBK - XK	20002350
	SIGMA(NXBK,UB) = UBK2	20002360
	SIGMA(NXBK,LB) = BRK2	20002370
	RLT(NXBK) = BRK	20002380
	CALL GETPHI(NXBK,BLT,TMP,OMY)	20002390
	PH1 = TMP(NXBK)	20002400
	RLT(NXBK) = BRK2	20002410
	CALL GETPHI(NXBK,RLT,TMP,OMY)	20002420
	PH2 = TMP(NXBK)	20002430
	IP2 = 0	20002440
	TSIG = EKO - PH1 + PH2	20002450
	SIGMA(NORA,RV) = -TSIG	20002460
	BO(NORA) = -TSIG	20002470
	RLT(NXBK) = BRK2	20002480
	ULT(NXBK) = BRK2 + UBK2	20002490
C	SET SLOPE OF X(K), IF BRK = 0	20002500
		20002510
		20002520
		20002530
		20002540
		20002550
		20002560
C	SOLVE K DOUBLE PRIME LP PROBLEM	20002570
		20002580
		20002590
		20002600
		20002610
		20002620
		20002630
		20002640
		20002650
		20002660
		20002670
104	CONTINUE	20002680
	NOPS = NOPS + 1	20002690
	DO 8887 J=1,NCF	20002700
	TMP(J)=0	20002710
8887	XCON(J)=0	20002720
	DO 8888 IND=1,MPLUS	20002730
	IF (IX(IND).GT.NCF .OR. IX(IND).EQ.0) GO TO 8888	20002740
	ICOL=IX(IND)	20002750
	TMP(ICOL)=X(IND)	20002760
	X(IND)=X(IND)+BL0(ICOL)	20002770
	YCON(ICOL)=X(IND)	20002780
8888	CONTINUE	20002790
	IND=0	20002800
	DO 8899 J=1,NCF	20002810
	IF (RLO(J).EQ.0.0) GO TO 8899	20002820
	IF (XCON(J).GT.0.0) GO TO 8899	20002830
	XCON(J)=RLO(J)	20002840
	IX(MPLUS-IND)=J	20002850
	X(MPLUS-IND)=RLO(J)	20002860
	IND=IND+1	20002870
8899	CONTINUE	20002880
	RP(12)=COST-TSIG	20002890

00	9999	J=1, MCF	20002900
9999		PP(12)=PP(12)-TMP(J)*CO(J)	20002910
		CALL GETPHI (MNC, YCON, TMP, PHIT)	20002920
		WRITE(6,573) PHIT	20002930
		IF (IP(11).EQ.1)	20002940
		*WRITE (6,575) (IX(I), X(I), I=1, MPLUS)	20002950
		IF (PHIT .GE. UZ) GO TO 109	20002960
		U7 = PHIT	20002970
		DO 1(7 I=1, MPLUS	20002980
		IX7(I)=IX(I)	20002990
107		XZ(I) = X(I)	20003000
		NEWX7=1	20003010
		USP = (U7 / (1.0 + EPSI))	20003020
		USM = (U7 / (1.0 - EPSI))	20003030
		US = USP	20003040
		IF (U7 .LT. 0.0) US = USM	20003050
109		CONTINUE	20003060
		IF (COST2.GE.US) GO TO 55	20003070
		CALL NXCON(YCON, SIGMA, NXP)	20003080
		GO TO 1990	20003090
4444		WRITE (6,4445)	20003100
4445		FORMAT (* HAVE SOLVED MAX. NO. OF LP PROBS. SET BY IP(12)*)	20003110
4446		WRITE(9,4448) (TITLE(I), I=1,4)	20003120
4448		FORMAT(4A10)	20003130
		WRITE(8,4447) (IXZ(I), X7(I), I=1, MPLUS)	20003140
4447		FORMAT(T4,4X,F12.4)	20003150
		WRITE(9,4447) MNC, U7	20003160
		NEWXZ=1	20003170
		CALL TABOUT (7)	20003180
		GO TO 7	20003190
25		CALL EXIT	20003200
		END	20003210
			20003220
			20003230

	SURPOUTIVE BOX1	20003240
C		20003250
C	LABELLED COMMON	20003260
	COMMON / CV1 / IP(12),RP(12),TMP(10)	20003270
	COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)	20003280
	COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS	20003290
	COMMON / CV4 / IX(110),X(110),IXZ(110),X7(110),XCON(10),COST	20003300
	COMMON / CV5 / SIGMA(100,4),TSIG ,LSTMAX	20003310
	COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ	20003320
	COMMON / CV8 / NYBK,XK,NOBOL,EKBL(25)	20003330
	COMMON / CV9 / PSIGL(25),NXBL(25),XNXBL(25),RLTST(25,131)	20003340
C		20003350
C		20003360
	INTEGER UB,CI,BV	20003370
C		20003380
C	BOX NO. 1 (NOP = 1)	20003390
C		20003400
	CT = 4	20003410
	UB = 3	20003420
	LB = 2	20003430
	BV = 1	20003440
	NORA = TP(2)	20003450
	MNC = (-1)*NCF	20003460
	MNX = (-1)*N	20003470
	CALL GETC (MNC,BLO,ULO,CO)	20003480
	CALL INITA (NCF,N,NOPA)	20003490
	CALL GETPHI(MNC,PL0,TMP,ESIG)	20003500
	EKO = ESIG	20003510
C	SET ISIGMA FOR 1ST LP PROR.	20003520
	DO 10 I = 1,NCF	20003530
	TMP(I) = 0.0	20003540
	SIGMA(I,LB)=BLO(I)	20003550
	SIGMA(I,UB)=ULO(I)	20003560
	SIGMA(I,CI)=CO(I)	20003570
10	CONTINUE	20003580
	DO 15 I = 1,NOPA	20003590
	SIGMA(I,BV) = BO(I)	20003600
15	CONTINUE	20003610
	TSIG = EKO	20003620
C		20003630
	NOP = 1	20003640
	DO 5555 IND=1,MPLUS	20003650
	XZ(IND)=0	20003660
	IXZ(IND)=0	20003670
	X(IND)=0	20003680
5555	IX(IND)=0	20003690
	CALL TABOUT (1)	20003700
	NCF1=NCF	20003710
	NF1=0	20003720
	CALL LP (NORA,N,NCF1)	20003730
	CALL TABOUT (2)	20003740
	IF(NF1.NE.1)GO TO 7	20003750
23	CONTINUE	20003760
	DO 31 J=1,NCF	20003770
31	XCON(J)=0	20003780
	DO 6666 IND=1,MPLUS	20003790
	IF (IX(IND).GT.NCF .OR. IX(IND).EQ.J) GO TO 6666	20003800

	ICOL=IX(INO)	20003810
	Y(INO)=X(INO)+RLO(ICOL)	20003820
	XCON(ICOL)=X(INO)	20003830
6556	CONTINUE	20003840
	DO 35 J=1,NPLUS	20003850
	IXZ(J)=IX(J)	20003860
	XZ(J) = X(J)	20003870
33	CONTINUE	20003880
	NEWX7=1	20003890
	RP(12)=COST	20003900
	DO 6567 J=1,NCF	20003910
6567	RP(12)=RP(12)-XCON(J)*CO(J)	20003920
	CALL GETPHT (MNC,XCON,TMP,UZ)	20003930
	EPST = RP(1)	20003940
	USP= (U7 / (1.0 + EPSI))	20003950
	USM= (U7 / (1.0 - EPSI))	20003960
	EKO = TSIG	20003970
C		20003980
C	10 SEP 6A	20003990
	CALL NXPRN (XCON,SIGMA,NXB)	20004000
	LSTMAX=1	20004010
	NOROL=1	20004020
	PSIGL(1)=COST	20004030
	XK=XCON(NXB)	20004040
	XNXBL(1)=XCON(NXP)	20004050
	NXBL(1)=NXB	20004060
	FKBL(1)=TSIG	20004070
51	CONTINUE	20004080
	DO 52 T = 1,NCF	20004090
	RLO(I) = SIGMA(T,LR)	20004100
	CO(I) = SIGMA(T,CI)	20004110
	ULO(I) = SIGMA(T,UR)	20004120
	I1=NOPA+T	20004130
	I2=NCF+I1	20004140
	I3=NCF+I2	20004150
	RLIST(1,I1)=RLO(T)	20004160
	RLIST(1,I2)=ULO(T)	20004170
	RLIST(1,I3)=CO(I)	20004180
52	CONTINUE	20004190
	DO 53 I = 1,NOPA	20004200
	RO(I) = SIGMA(T,OV)	20004210
	RLIST(1,I)=RO(I)	20004220
53	CONTINUE	20004230
777	RETURN	20004240
7	CALL JRCV2	20004250
	END	20004260

```

SUBROUTINE GETASQ(NOES,ELM,JSQ)
C...SHELL METHOD OF HALVING
C
C GETASQ(NOES,ELM,JSQ) SORTS ELM(J),J=1,NOES IN AN ASCENDING SEQUENCE
C PRESET INITIAL POSITION CODE OF ELM(J)
C JSQ(J) PRESET TO (-1) WHEN ELM(J) IS UNDEFINED (I.E. INFINITE)
C
  DIMENSION ELM(1), JSQ(1)
  L = 1
  7  L = 2 * L
    IF( L.LE. NOES) GO TO 7
    L = L - 1
  10 L = L / 2
    DO 20 K2 = 1, NOES
      K1 = K2
  15  K3 = K1 + L
      IF( K3 .GT. NOES) GO TO 30
      IF( ELM(K1).LE. ELM(K3) ) GO TO 20
      RT = ELM(K1)
      ELM(K1) = ELM(K3)
      ELM(K3) = RT
      JSQ(K1) = JSQ(K3)
      JSQ(K3) = RT
      K1 = K1 - L
      IF( K1 .GE. 1) GO TO 15
  20 CONTINUE
  30 IF( L .GT. 1) GO TO 10
    RETURN
  END

```

	SUBROUTINE GETC (KCX,RLT,ULT,CT)	20004610
	COMMON / CV1 / IP(12),RP(12),TMP(10)	20004610
	COMMON / CV3 / M,N,NCF,PHIT,UZ,UTP,USM,EKO,MPLUS	20004620
	DIMENSION RLT(01),ULT(01),CT(01),FX1(10),FX2(10)	20004630
C		20004640
C		20004650
C	IF (KCX) 1. .GT. 0, EVALUATE KCX(TH) C(X)-SLOPE.	20004660
C	2. .LT. 0, EVALUATE CX(1) TO CX(IFX), (IFX = -KFX).	20004670
C	3. .EQ. 0, INVALID KCX **** UEP.	20004680
		20004690
	IF (IP(6) .EQ. 1) WRITE(6,999)	20004700
999	FORMAT(14-,124X,FMGETC)	20004710
	IF (KCX .GT. N) GO TO 770	20004720
	IF (KCX) 200,770,100	20004730
100	FX1(KCX) = 0.0	20004740
	FX2(KCX) = 0.0	20004750
	CALL GETPHI(KCX,RLT,FX1,DMY)	20004760
	CALL GETPHI(KCX,ULT,FX2,DMY)	20004770
	NOX1 = KCX	20004780
	NOX2 = KCX	20004790
	GO TO 220	20004800
200	ICX = (-1) * KCX	20004810
	IF (ICX .GT. N) GO TO 770	20004820
	DO 210 I = 1, ICX	20004830
	FX1(I) = 0.0	20004840
	FX2(I) = 0.0	20004850
	CT(I) = 0.0	20004860
210	CONTINUE	20004870
	CALL GETPHI(KCX,RLT,FX1,DMY)	20004880
	CALL GETPHI(KCX,ULT,FX2,DMY)	20004890
	NOX1 = 1	20004900
	NOX2 = ICX	20004910
220	DO 225 J = NOX1,NOX2	20004920
	DIF = ULT(J) - RLT(J)	20004930
	IF (DIF .EQ. 0.0) GO TO 225	20004940
	CT(J) = (FX2(J) - FX1(J)) / DIF	20004950
225	CONTINUE	20004960
	GO TO 777	20004970
770	WRITE(6,771)KCX	20004980
771	FORMAT(1H1,13HINVALID KCX =,I3,10H IN GETC)	20004990
	CALL EXIT	20005000
C		20005010
	777 CONTINUE	20005020
	888 RETURN	20005030
C		20005040
	END	

	SUBROUTINE GETPHI(KFX,XPHI,PHI,SUMPHI)	20005050
	DIMENSION XPHI(61),PHI(61)	20005060
	COMMON / CV1 / IP(12),RP(12),TMP(10)	20005070
	COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)	20005080
	COMMON / CV3 / M,N,NC,F,PHI,T,UZ,USP,USH,EKO,MPLUS	20005090
C		20005100
C	IF (KFX) 1. .GT. 0, EVALUATE KFX(TH) F(X).	20005110
C	2. .LT. 0, EVALUATE FX(1) TO FX(IFX), (IFX = -KFX).	20005120
C	3. .EQ. 0, INVALID KFX **** UEP.	20005130
C		20005140
	IF(IP(6) .EQ. 1) WRITE(6,999)	20005150
999	FORMAT(1H-,124X,6HGETPHI)	20005160
C		20005170
	IF(KFX)100,300,500	20005180
100	SUMPHI=PP(12)	20005190
	I = 1	20005200
160	IF(I+KFX) 150,150,400	20005210
150	IF(I.GT.4)GO TO 140	20005220
	GO TO (101,102,103,104) I	20005230
101	IF(XPHI(I).LT..00(1) GO TO 140	20005240
	PHI(I) = 0.30 + 0.006*XPHI(I)**0.95	20005250
	GO TO 200	20005260
102	PHI(I) = 0.0034*XPHI(I)**0.96	20005270
	GO TO 200	20005280
103	PHI(I) = 0.006*XPHI(I)**0.90	20005290
	GO TO 200	20005300
104	PHI(I) = 0.015*XPHI(I)**0.909	20005310
	GO TO 200	20005320
140	PHI(I) = 0.0	20005330
200	SUMPHI = SUMPHI + PHI(I)	20005340
	IF(KFX.GT.0) RETURN	20005350
	I = I+1	20005360
	GO TO 160	20005370
500	SUMPHI = 0.0	20005380
	I = KFX	20005390
	GO TO 150	20005400
300	WRITE(6,301)	20005410
301	FORMAT(1H1,25HKFX = 0 IN GETPHI	20005420
	CALL EXIT	20005430
400	RETURN	20005440
	END	20005450

	SUBROUTINE INITA(NCF,N,M)	20005460
		20005470
	THIS SUBROUTINE COPIES THE A MATRIX FROM TAPE TO DISC	20005480
	AND STORES THE R0 AND C0 ARRAYS IN CORE. TAPE9 IS ASSUMED	20005490
	TO BE THE TAPE AND TAPE3 IS THE DISC FILE.	20005500
		20005510
	COMMON / CV2 / T(100,10),R0(100),BLO(10),ULO(10),CO(10)	20005520
	COMMON /POWTP/ IPWTP(101)	20005530
	DIMENSION AJ(100)	20005540
	REWIND 3	20005570
	REWIND 9	20005580
	READ (9,100) NUM1,NUM2	20005590
100	FORMAT (A4,10X,A9)	20005600
	READ (9,400) (IPWTP(J),J=1,M)	20005610
400	FORMAT (I12)	20005620
	DO 10 T=1,N	20005630
	READ (9,200) (AJ(J),J=1,M)	20005640
200	FORMAT (F12.4)	20005650
	IF (EOF,9) GO TO 20	20005660
20	IF (T.NE.N) GO TO 40	20005670
	DO 30 J=1,M	20005680
30	R0(J)=AJ(J)	20005690
40	IF (T.GT.NCF) GO TO 60	20005700
	AJ(M)=CO(T)	20005710
	DO 55 J=1,M	20005720
55	T(J,T)=AJ(J)	20005730
60	WRITE (3) (AJ(J),J=1,M)	20005740
	WRITE (7,1) (AJ(J),J=1,M)	20005750
1	FORMAT (5F15.5)	20005760
2	WRITE (6,2) (AJ(J),J=1,M)	20005770
	FORMAT (1X,5F15.5)	20005780
10	CONTINUE	20005790
	IPWTP(M)=3	20005800
	END FILE 3	20005810
	RETURN	20005820
1000	WRITE (6,300) T	20005830
300	FORMAT (* PREMATURE EOF ON A MATRIX TAPE AT COLUMN *,I5)	20005840
	STOP0002	20005850
	END	20005960

	SUBROUTINE NXBRN(XT, SIGMAT, NXB)	20005870
	COMMON / CV1 / IP(12), RP(12), TMP(10)	20005880
	COMMON / CV3 / M, N, NCF, PHIT, UZ, USP, USM, EKO, MPLUS	20005890
	DIMENSION XT(001), RLT(10), CT(10), YT(10), SIGMAT(100,4)	20005900
	DIMENSION FX1(10), FX2(10), DIF(10), NDX(10)	20005910
10	CONTINUE	20005920
	IF(IP(6).EQ.1) WRITE(6,999)	20005930
999	FORMAT(1H-,124X,6HNXB)	20005940
	NXB = 0	20005950
C	NXB RN GIVES BEST BRNCH-CANDIDATE FOR XT(J)	20005960
	DO 5 J = 1, NCF	20005970
	FX2(J) = 0.0	20005980
	FX1(J) = 0.0	20005990
	DIF(J) = 0.0	20006000
	NDX(J) = 0	20006010
	RLT(J) = SIGMAT(J,2)	20006020
	CT(J) = SIGMAT(J,4)	20006030
5	CONTINUE	20006040
	DO 20 J = 1, NCF	20006050
	YT(J) = XT(J) - RLT(J)	20006060
20	CONTINUE	20006070
	NFX = (-1) * NCF	20006080
	CALL GETPHI(NFX, XT, FX2, DMY)	20006090
	CALL GETPHI(NFX, RLT, FX1, DMY)	20006100
40	CONTINUE	20006110
	IF (RP(4).NE.0)	20006120
	*WRITE (6,55)	20006130
55	FORMAT(1H0,10X,*DIFFERENCE =*,10X,*PHI(X) - PHI(LOWER BOUND)	20006140
	* - (*,8X,4HC(X),4X,1H*,12X,1HX,6X,1H)	20006150
	DO 30 J = 1, NCF	20006160
	DIF(J) = FX2(J) - FX1(J) - CT(J)*YT(J)	20006170
	IF (RP(4).NE.0)	20006180
	* PRINT 50, J, DIF(J), FX2(J), FX1(J), CT(J), YT(J)	20006190
50	FORMAT(1H0, I5, 6F20.6)	20006200
	NDX(J) = J	20006210
30	CONTINUE	20006220
	CALL GETASQ(NCF, DIF, NDX)	20006230
	NXB = NDX(NCF)	20006240
	RETURN	20006250
1000	CONTINUE	20006260
	END	20006270

	SUBROUTINE PARAMS	20006280
C		20006290
C	LABELLED COMMON	20006300
	COMMON / CV1 / IP(12),OP(12),TMP(10)	20006310
	COMMON / CV2 / T(100,10),BO(100),RLO(10),ULO(10),CO(10)	20006320
	COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS	20006330
	COMMON / CV4 / TX(110),X(110),TX7(110),X7(110),XCON(10),COST	20006340
	COMMON / CV5 / STOMA(100,4),TSIG,IFIL	20006350
	COMMON / CV7 / NPHASE,NF1,CFX,IOP1,NOP,NOPS,NEWXZ	20006360
	COMMON / CV8 / NXPX,XK,NOROL,EK91(25)	20006370
	COMMON / CV9 / PSTGL(25),NXBL(25),XNXBL(25),BLIST(25,131)	20006380
C		20006390
	READ(5,10) (IP(I),I=1,12)	20006400
10	FORMAT(12I6)	20006410
	REWIND 9	20006420
	READ(9,11) IP(1), IP(2)	20006430
11	FORMAT(2I6)	20006440
	IF(EOF, 5) 77777, 15	20006450
15	CONTINUE	20006460
C		20006470
C	IP1=N, IP2=NORA, IP3=NCF, IP4=MMAX, IP5=NMAX, IP6=LPIN, IP7=LPOUT	20006480
C	IP8=ICTU, IP9=TBMAX, IP10=JRMAY, IP11=ICK, IP12=MXNDP	20006490
C		20006500
C	N - TOTAL NO. OF VARIABLES	20006510
C	NORA - NO. OF ROWS IN A-MATRIX	20006520
C	NCF - NO. OF VARIABLES W/CONCAVE-F(X)	20006530
C	MMAX - MAX. NO. OF CONSTRAINTS FOR JPLP	20006540
C	NMAX - MAX. NO. OF VARIABLES FOR JPLP	20006550
C	LPIN - IF (1) WRITE LP INPUT FOR EACH PROBLEM	20006560
C	LPOUT - IF (1) WRITE LP OUTPUT FOR EACH PROBLEM	20006570
C	ICTU - IF (1) CONSTRAIN COST-F(X) .LT. UO	20006580
C	TBMAX - MAX NO. OF ROWS IN BLIST	20006590
C	JRMAY - MAX NO. OF COLUMNS IN BLIST	20006600
C	ICK - IF (1) SET PRINT = .TRUE. IN JPLP	20006610
C	MXNDP - MAX. NO. OF LP PROBS. SOLVED BEFORE CALLING EXIT	20006620
C		20006630
	READ(5,20) (OP(I),I=1, 6)	20006640
20	FORMAT(6F12.0)	20006650
C		20006660
C	OP1=EPST, OP2=TMAY, OP3=THETA, OP4=TRACE	20006670
C	EPSI - ADJUSTMENT FACTOR FOR UO	20006680
C	TMAY - MAX. RR-EXCT TIME IN SECONDS	20006690
C	THETA - X(I) ZERO RNDOFF	20006700
C	TRACE - IF (1) TRACE SOLUTION, USING LPIN, LPOUT, AND ICK CODES	20006710
C	IF(0) SKIP ALL INTERMEDIATE PRINT OUT	20006720
C		20006730
	TMAY = RP(2)	20006740
	CALL SFT(TMAY)	20006750
C		20006760
	CALL PRSET	20006770
C		20006780
	WRITE(6,30) (IP(I),I=1,12)	20006790
30	FORMAT(14I,20HINTEGER PARAMETERS =,12I6)	20006800
	WRITE(6,40) (OP(I),I=1,6)	20006810
40	FORMAT(1H-,17HREAL PARAMETERS =,6F18.8)	20006820
	N = IP(1)	20006830
	NORA = IP(2)	20006840

	NCF = IP(3)	20006850
	M=NCF + NORA	20006860
C		20006870
	IBMAX = IP(9)	20006880
C		20006890
55	READ(9,20) (ULO(J), J = 1,NCF)	20006900
	DO 60 J=1,NCF	20006910
	IF (ULO(J).LT.0) ULO(J) = - ULO(J)	20006920
60	CONTINUE	20006930
C	READ(5,20) (BLO(J),J=1,NCF)	
	DO 61 J=1,NCF	
C		20006950
61	BLO(J) = 0.0	
	WRITE(6,90)	20006960
90	FORMAT(14-,25H X(J) LOWER-UPPER BOUNDS)	20006970
	DO 100 J = 1,NCF	20006980
	WRITE(6,95) J,BLO(J),ULO(J)	20006990
95	FORMAT(1H0,2X,I3,3X,2E12.4)	20007000
100	CONTINUE	20007010
	NOROL = 0	20007020
777	RETURN	20007030
77777	CONTINUE	20007040
	STOP 00001	20007050
	END	20007060

SUBROUTINE DRESET

C

C LABELLED COMMON

COMMON / CV1 / IP(12),PP(12),TMP(1)
 COMMON / CV2 / T(10,10),B0(10),RL0(10),UL0(10),C0(10)
 COMMON / CV3 / M,N,NCF,PHIT,U7,USP,USM,FK0,MPLUS
 COMMON / CV4 / IX(110),X(110),IX7(110),X7(110),XCON(10),COST
 COMMON / CV5 / SIGMA(10,4),TSTG
 COMMON / CV7 / NPHASE,NF1,CFX,IORT,NOP,NOPS,NEWX7
 COMMON / CV8 / NYRK,XK,NOPOL,FKRI(2F)
 COMMON / CV9 / PSTGL(25),NYRL(25),XNYRL(25),RLIST(25,171)

IF (IP(5) .EQ. 1) WRITE(6,999)
 112 FORMAT(1H-,124X,6HPRESET)
 V=IP(1)
 NOPA = IP(2)
 NCF=IP(3)
 IRMAX = IP(4)
 JRMAX = IP(10)

C

DO 13 J=1,NCF
 C0(J) = 0.0
 RL0(J) = 0.0
 UL0(J) = 0.0

C

13 CONTINUE

C

DO 15 I = 1,NOPA
 DO 14 K=1,4

14 SIGMA(I,K)=0.0
 B0(I) = 0.0

C

15 CONTINUE

C

NEWX7=0
 PHIT=0
 U7 = 1.E+35
 USP = 1.E+36
 USM = 1.E+36
 COST=0.0
 TSTG=0.0
 NF1=0
 CFX=0.0
 IORT=0
 NOP=0
 NYRK=0
 XK = 0.0
 NOPOL=0

C

DO 20 I = 1,IRMAX
 FKRI(I) = 0.0
 PSTGL(I) = 0.0
 NYRL(I) = 0
 XNYRL(I)=0.0
 DO 20 J=1,JRMAX
 RLIST(I,J) = 0.0

C

20 CONTINUE

20007070
 20007080
 20007090
 20007100
 20007110
 20007120
 20007130
 20007140
 20007150
 20007160
 20007170
 20007180
 20007190
 20007200
 20007210
 20007220
 20007230
 20007240
 20007250
 20007260
 20007270
 20007280
 20007290
 20007300
 20007310
 20007320
 20007330
 20007340
 20007350
 20007360
 20007370
 20007380
 20007390
 20007400
 20007410
 20007420
 20007430
 20007440
 20007450
 20007460
 20007470
 20007480
 20007490
 20007500
 20007510
 20007520
 20007530
 20007540
 20007550
 20007560
 20007570
 20007580
 20007590
 20007600
 20007610
 20007620
 20007630



RETURN
END

20007640
~~20007650~~

```

SUBROUTINE READIN
COMMON / CV1 / IP(12),PP(12),TMP(12)
DATA ENDEF / 5HEND /
REWIND 7
NC=PP(3)
IF(NC.EQ.0) GO TO 20
DO 10 J=1,NC
READ (5,100) (TMP(J),J=1,4)
WRITE (7,100) (TMP(J),J=1,4)
100 FORMAT (4A10)
10 CONTINUE
20 WRITE(7,100) ENDEF
RETURN
END

```

```

20007660
20007670
20007680
20000210
20007690
20007700
20007710
20007720
20007730
20007740
20007750
20007760
20007770
20007780

```

Reproduced from
best available copy. 

	SUBROUTINE SET(TMMAX)	20007790
	COMMON/TMX/TMO,EXT	20007800
C		20007810
C	SECOND GIVES JOB CPU EXECUTION TIME IN 1/1000 OF A SECOND	20007820
C		20007830
	CALL SECOND(TMO)	20007840
	FXT = TMMAX + TMO	20007850
	RETURN	20007860
	END	20007870


```

SUBROUTINE TAPOUT(IPT)
C LABELLED COMMON
COMMON / CV1 / IP(12),PP(12),TMP(12)
COMMON / CV2 / T(100,10),RQ(100),BLQ(10),ULQ(10),CQ(10)
COMMON / CV3 / M,N,NCF,PHIT,UZ,UCP,USM,EKQ,MPLUS
COMMON / CV4 / IX(11),X(110),IX7(110),X7(110),XCON(11),COST
COMMON / CV5 / SIGMA(100,4),TSIG
COMMON / CV7 / NPHASE,NE1,CFX,TOPT,NOP,NOPS,NEWX7
COMMON / CV8 / NYBK,YK,NORQL,EKQL(2F)
COMMON / CV9 / PSIGL(25),NXRL(25),YXRL(25),PLTST(25,131)

IF(NOP.EQ.1) GO TO 90
GO TO 100
90 CONTINUE
IF(PP(4).EQ.0.0) GO TO 777
100 CONTINUE
WRITE(6,101)
101 FORMAT(1H1,30HTAPOUT - GENERAL - INFORMATION)
IF (IPT.NE.3) CALL TIMEQ
WRITE(6,105)UZ,USP
105 FORMAT(1H0,6HUZ=,F18.7,6Y,6HUSP =,F18.7)
IF (NEWX7.EQ.1)
*WRITE (6,570) (IX7(I),X7(I),I=1,MPLUS)
570 FORMAT (7H0X7ERO // (7X,5(7H COL ,I4,2H =,F12.4)))
IF (IPT.EQ.3) GO TO 777
NEWX7=0
WRITE(6,109)
109 FORMAT(1H0,10HSIGMA(T,J),13X,5H NPH-R,12X,6HLW-RND,12X,6HUP-RND,
111X,7HC-SLOPE)
C
C
DO 115 I=1,NCF
WRITE(6,113) I, (SIGMA(T,J),J=1,4)
113 FORMAT(1H ,5X,I2,7X,4F19.5)
115 CONTINUE
C
WRITE(6,117)TSIG
117 FORMAT(1H0,6HE(K) =,F18.6)
C
IF(IPT.NE.1)GO TO 145
C
IF(NORQL.EQ.0) GO TO 145
IT = NORQL
DO 131 I=1,IT
WRITE(6,121) I,PSIGL(I),NXRL(I),YXRL(I),EKQL(I)
121 FORMAT(1H-,6HNODE =,I4,6X,6HCOST =,F20.6,6Y,6HNX-RRN =,I4,6X,
1 7HY-RPN =,F20.6,6Y,6HE(K) =,F20.6)
131 CONTINUE
145 CONTINUE
777 RETURN
END
20007880
20007890
20007900
20007910
20007920
20007930
20007940
20007950
20007960
20007970
20007980
20007990
20008000
20008010
20008020
20008030
20008040
20008050
20008060
20008070
20008080
20008090
20008100
20008110
20008120
20008130
20008140
20008150
20008160
20008170
20008180
20008190
20008200
20008210
20008220
20008230
20008240
20008250
20008260
20008270
20008280
20008290
20008300
20008310
20008320
20008330
20008340
20008350
20008360
20008370
20008380

```

	SUBROUTINE TIMEC	20008390
	COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USH,EKO,MPLUS	20008400
	COMMON / CV4 / IX(110),X(110),IXZ(110),XZ(110),XCON(10),COST	20008410
	COMMON / CV7 / NPHASE,NF1,CFX,IOP,T,NOP,NOPS,NEWXZ	20008420
	COMMON /TSW/ NSWW	20008430
	COMMON/TMX/TMO,EXT,TITLE(4)	20008440
C		20008450
C	SECOND GIVES JOB CPU EXECUTION TIME IN 1/1000 OF A SECOND	20008460
C		20008470
	CALL SECON0(SECS)	20008480
	XX= SECS - TMO	20008490
	WRITE(6,666) XX	20008500
666	FORMAT(12H0FYCT-TIME =,F9.3,8H SECONDS)	20008510
	IF(SECS .LT. EXT) GO TO 100	20008520
	WRITE(6,667)	20008530
667	FORMAT(37H TIME IS UP...CYCLING TO NEXT PROBLEM)	20008540
	MNC=(-1)*NCF	20008550
4446	WRITE(8,4448) (TITLE(I),I=1,4)	20008560
4448	FORMAT(4A10)	20008570
	WRITE(8,4447) (IX7(I),XZ(I),I=1,MPLUS)	20008580
4447	FORMAT(I4,4X,F12.4)	20008590
	WRITE(8,4447)MNC,117	20008600
	NEWXZ=1	20008610
	CALL TABOUT (3)	20008620
	CALL BRCAV2	20008630
100	RETURN	20008640
	END	20008650

```

SUBROUTINE LP(MPOWS,NCOLS,NCHGS)                20008660
COMMON / CV1 / IP(12),PP(12),TMP(10)           20008670
COMMON / CV2 / T(10,10),RO(100),RLO(1.),URS(10),CO(10) 20008680
COMMON / CV4 / IX(110),X(110),IX7(110),X7(110),XCON(10),COST 20008690
COMMON / CV7 / NPHASE,NF1,CFX,ICPT,NOP,NOPS,NEWXZ 20008700
C-----SET PHS TO INPUTM+1                       20008710
COMMON /PHS/ PHS(100)                          20008720
C-----SET AJ(AS MUCH AS POSSIBLE) OVER INPUTM+1**2 FOR CORE COLUMNS 20008730
COMMON /CORE/ JAPEJ(101),JA(101),JAK(101),AJ(11000) 20008740
COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101) 20008750
COMMON /DJS/ DJ(101)                          20008760
C-----SET IPWTP(INPUTM+1) NAME(INPUTM+INPUTM+1) 20008770
COMMON /IPWTP/ IPWTP(101) /NAMES/ NAME(600)      20008780
COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,TPHASE,JPHS,IP  20008790
COMMON /LIMS/ MAXTPY,NTRY,JNCOPE,NORMAX,NSCAN    20008800
COMMON /STATE/ JPOS,TPHW,JCOL,JOUT,ITRN,NREJ,NPIF,NOJS 20008810
COMMON /FILES/ TA1,TA2,IMAP                   20008820
COMMON /INPUT/INPUT,INPUTM,INPUTN             20008830
COMMON /ROUNDS/ ROUNDS(100),JBS(100),NRDS     20008840
COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5 20008850
COMMON /IXX/IXX(100) /XX/ XX(100)            20008860
C-----THE R ORIGIN IS MOVED DOWN THE AJ SPACE --IGNORE SIZE 20008870
REAL R(100)                                   20008880
EQUIVALENCE (AJ,R)                           20008890
C                                              20008900
CALL MSSG(40HLP/LONG/5-GUB CYCLIC           20008910
C-----FILE DEFINITIONS                          20008920
IA1=1                                         20008930
IA2=2                                         20008940
INPUT=3                                       20008950
IMAP=7                                        20008960
REWIND INPUT                                 20008970
CALL FTNPRN(1,1,IA1)                       20008980
CALL FTNPRN(1,1,IA2)                       20008990
C-----SET LENGTH OF AJ SPACE IN NWAJ          20009000
NWAJ=11000                                  20009010
C                                              20009020
C                                              20009030
C                                              20009040
C                                              20009050
C                                              20009060
C                                              20009070
C                                              20009080
C                                              20009090
C                                              20009100
ICOST=INPUTM=MPOWS                          20009110
JPHS=INPUTN=NCOLS                          20009110
NRDS=NCHGS                                  20009120
DO 10 I=1,NRDS                              20009130
  IPDS(I)=I                                  20009140
10 ROUNDS(I)=URS(I)                         20009150
C-----MAP OUT AFTER TMAX, QUIT AFTER K5 CYCLES 20009160
TMAX=200.                                    20009170
K5=200                                       20009180
C-----XCHECK BETWEEN CYCLES NS00 TO NNNAT INCREMENTS K2 20009190
K2 = 1                                       20009200
K4=100                                       20009210
K4=0                                         20009220

```

```

      K4=1
C-----PRINT CONTROL K3
      K3 = 0
      K3=5*7*11*13
      IF(IP(8).NE.1) K3=K3/5
      IF(IP(7).EQ.0) K3=13*7
      IF(IP(4).EQ.0.0) K3=1
C
C
C
C
C
C
C
C
C
C
C
C
C-----PROGRAM ERRS
100 CALL SETUP
      WRITE(6,999) IPQWTP
999  FORMAT(* DUMP IPQWTP*/(1X,50T1))
C-----M IS NOW ACTUAL NON-GUR ROWS, L IS NO. OF GUR ROWS NWAJ IS AJ SPAC
      MPL=M+L
C-----OPTIMIZE CORE COLUMN STORAGE
      IORG=NWAJ-M*M
      NCPMAX=MIN0( 99,IORG/M) - 3
200 CALL MAPIN(R(IORG))
250 FORMAT(/** LP PROBLEM DATA FOR THIS RUN *
+          /* NON-GUR ROWS * I6
+          /*   GUR-ROWS * I6
+          /
+          /* LOGICALS      * I6
+          /* TOTAL COLUMNS* I6
+          /* MAX IN CORE  * I6
+          /* INVERT FREQU.* I6 * CYCLES*
+          /* MAX RUN TIME * F6 * SECONDS*
+          /* MAX CYCLES   * I6 * ITERATIONS*
+          /**/)
      WRITE(6,250) M,L,M0,NT,NCPMAX,INVE,TMAX,K5
      IF (IORG.GE.2*M) GO TO 300
      CALL ERROR(40HLP--INSUFFICIENT SPACE STATED IN NWAJ )
      CALL ESCAPE(R(IORG))
300 CALL INVERT(R(IORG))
400 CALL PRIMAL(R(IORG))
      ITNTNV=0
      CALL INVERT(R(IORG))
      IPT=IORG+IPT-1
      DO 500 J=1,NT
      CALL IN(J,AJ,0)
500  DJ(J)=DOT(M,R(IPT),AJ)
      IPT=IPT-IORG+1
      WRITE(6,501) (J,DJ(J),NAME(J),J=1,NT)
501  FORMAT(*00J VALUES FOR FINAL SOLUTION COLUMNS*/(T10,E12.4,T10))
C-----END PHASE 2, OR UNBOUNDED OR NO FEASIBLE SOLUTION
900 CONTINUE
      CALL MAPOUT(R(IORG))

```

```

20009230
20009240
20009250
20009260
20009270
20009280
20009290
20009300
20009310
20009320
20009330
20009340
20009350
20009360
20009370
20009380
20009390
20009400
20009410
20009420
20009430
20009440
20009450
20009460
20009470
20009480
20009490
20009500
20009510
20009520
20009530
20009540
20009550
20009560
20009570
20009580
20009590
20009600
20009610
20009620
20009630
20009640
20009650
20009660
20009670
20009680
20009690
20009700
20009710
20009720
20009730
20009740
20009750
20009760
20009770
20009780
20009790
20009800

```

```

      NVAR=INPIJTM+NCHGS
      DO 900 I=1,NVAR
      IX(I)=TXX(I)
33)  Y(I)=XX(I)
      COST=-BETA(ICOST)
7776 RETURN
      END

```

```

20009810
20009820
20009830
20009840
20009850
20009860
20009870

```

FUNCTION BOUND(J)	20009880
COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS	20009890
COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI	20009900
COMMON /NAMES/ NAME(100)	20009910
C-----PICKS UP BOUND FROM PACKED LIST,EITHER FINITE OR 10**35	20009920
BOUND=1.E70	20009930
IF(J.GT.NT) RETURN	20009940
IB=NAME(J)/100000	20009950
IF(IB.LE.0.OR.IB.GT.NBDS) RETURN	20009960
BOUND = BOUNDS(IA)	20009970
RETURN	20009980
END	20009990

```

SUBROUTINE COLUMN(JCOL,B)
C-----GUB VERSION APRIL 20-71
COMMON /MOVES/ THETA,RNDJ,DMAX,PRMLR,QUALEP
COMMON /STATE/ JPOS,TROW,JKOL,JOUT,TTRN,NREJ,NPTF,NDJS
COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DELTOL
COMMON /T/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPT
COMMON /LIMS/ MAXTRY,NTRY,JNCOPE,NORMAX,NSCAN
COMMON /A/ ALPHA(101) /B/ BETA(1 1) /C/ GAMMA(101) /D/ DELTA(101)
COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
COMMON /NAMES/ NAME(100)
COMMON /DJS/ DJ(100)
LOGICAL BASIC,ATRND,NULL,KEY
REAL R(1)
KEY(I)=MOD(NAME(I),10).EQ.4
NPKT(J)=MOD(NAME(J),10000)/10
C
C-----CHECKS COLS IN CORE, IF NONE GETS SOME, IF SOME FINDS BEST
NTRY = 1+NTRY
MAXTRY=NORMAX
IF( NTRY.GT. MAXTRY) GOTO 1
IF( JNCOPE.NE.0) GOTO 5
C-----CHECK FOR MORE COLUMNS ON DISC
1 CALL DISC(9)
NTRY = 0
NDJST=NDJS
IF( JNCOPE.EQ.0) GOTO 100
C
C-----RE-PRICE VECTORS IN CORE
5 JORG = 1
C-----NO PRICING IF REJECTS OR JUST PRICED DISC
IF(NREJ.NE.0 .OR. NTRY.EQ.0) GOTO 50
C-----PRICE OUT COLUMN
DO 40 J= 1,JNCOPE
DJ(J)=DOT(M,R(IPT),AJ(JORG))
40 JORG = JORG+M
C
C-----NOW FIND BEST COLUMN IN CORE, NON-BASIC OR ROUNDED
50 DMAX=0
NDJS=0
JPKT=0
PIKEY=0.
DO 60 J=1,JNCOPE
IF(JAREJ(J).EQ.1) GOTO 60
JPOS = JA(J)
JTYPE=MOD(NAME(JPOS),10)
IF(JTYPE.EQ.2) GOTO 60
IF(JTYPE.EQ.4) GOTO 60
IF(JTYPE.EQ.0) GOTO 60
JPKT=NPKT(JPOS)
IF(JPKT.EQ.0) GOTO 55
IF(JPKT.EQ.JPKT) GOTO 55
JKEY=KEYFND(JPKT)
C-----NEW PACKET STARTED, FINE KEY AND KEY PRICE
IF(JKEY.EQ.0) GOTO 60
PIKEY=DJ(JKEY)
JPKT=JPKT
55 D=DJ(J)

```

```

20010000
20010010
20010020
20010030
20010040
20010050
20010060
20010070
20010080
20010090
20010100
20010110
20010120
20010130
20010140
20010150
20010160
20010170
20010180
20010190
20010200
20010210
20010220
20010230
20010240
20010250
20010260
20010270
20010280
20010290
20010300
20010310
20010320
20010330
20010340
20010350
20010360
20010370
20010380
20010390
20010400
20010410
20010420
20010430
20010440
20010450
20010460
20010470
20010480
20010490
20010500
20010510
20010520
20010530
20010540
20010550
20010560

```

	IF(JTYPE.EQ.3) D=-DJ(J)	20010570
	IF(JPKT.NE.0) D=0-PIKEY	20010580
	IF(D.LT.-ZERO) NDJS=1+NDJS	20010590
	IF(D.GE. DMAX) GOTO 60	20010600
	DMAX =D	20010610
	JCOL = J	20010620
60	CONTINUE	20010630
	NCORE=JNCORE	20010640
C----	RESTORE COUNT OF NDJS FROM CHECK IF JUST DONE	20010650
	IF(NTRY.EQ.0 .AND. NT.GT.NCRMAX) NDJS=NDJST	20010660
	IF(DMAX.LT.-DJTOL) GOTO 70	20010670
C----	CURRENT COLS NO-GOOD, QUIT IF THESE ARE BEST	20010680
	IF(NTRY.EQ.0) GOTO 100	20010690
	GOTO 1	20010700
C		20010710
C----	RETURN WITH COLUMN INDEX	20010720
	70 RETURN	20010730
C		20010740
C----	NO GOOD COLS, OPTIMUM	20010750
	100 JCOL=0	20010760
C----	SAVE OLD COLUMNS	20010770
	JNCORE=NCORE	20010780
	RETURN	20010790
C		20010800
	END	20010810

	SUBROUTINE DISC(R)	20010820
C	REVISED 10/71	20010830
C-----	CHECKS DISC FOR COLUMNS, ACCEPTING 1/NBCH, IF NOT ALL IN CORE.	20010840
C	RETURNS JNCORE COLUMNS AND PRICES, OR JNCORE=0	20010850
C-----	PACKETS CAN IN BE INTER-MIXED WITH SOME LOSS OF EFFICIENCY	20010860
C	DUE TO MULTIPLE KEY SEARCHES	20010870
	COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IPI	20010880
	COMMON /STATE/ JPOS,TRON,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS	20010890
	COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DEPTOL	20010900
	COMMON /PARAMS/ TMAX,ITNINV,INVE,K1,K2,K3,K4,ITNCHK	20010910
	COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCPMAX,NSCAN	20010920
	COMMON /CORE/ JAPFJ(101),JA(101),JAK(101),AJ(100)	20010930
	COMMON /BASIS/ IRSTS(101),KEYS(101)	20010940
	COMMON /DJS/ DJ(100)	20010950
	COMMON /NAMES/ NAME(100)	20010960
	INTEGER PKT,PKTO	20010970
	REAL R(1)	20010980
	LOGICAL BASIC,ATRND ,NULL,CHEK	20010990
	NPKT(J)=MOD(NAME(J),100000)/10	20011000
	NULL(I)=MOD(NAME(I),10).EQ.0	20011010
C		20011020
C-----	CHECK FOR AN INVERT (ITRN.GE.ITNINV)	20011030
	CALL INVERT(R)	20011040
C		20011050
C-----	ALL IN CORE, NCPMAX SET IN LP	20011060
	IF(NT.LT.NCPMAX) GOTO 200	20011070
C-----	ACCEPT 1 COL/NBCH COLS, BEST AT IORG, NEW AT JORG, (ICOL,JCOL)	20011080
	NBCH=NT/NCPMAX/4+1	20011090
	NDJS=PKTO-JNCORE=0	20011100
	JORG=0	20011110
	JCOL=1	20011120
	IORG=M	20011130
	ICOL=2	20011140
	JNCORE=2	20011150
	CALL INPOS(JNT)	20011160
	NBCHS = (NT+NBCH-1)/NBCH	20011170
C		20011180
	DO 1000 JACH =1, NBCHS	20011190
	DJOLD = 1.E35	20011200
	DO 100 JFACH= 1,NBCH	20011210
C-----	BATCH CYCLE, NEXT COLUMN JNT	20011220
	JNT= 1+MOD(JNT,NT)	20011230
	JTYPE = MOD(NAME(JNT),10)	20011240
C-----	SKIP NULL, BASIC OR KEY COLUMNS	20011250
	IF(JTYPE.EQ.2) GOTO 100	20011260
	IF(JTYPE.EQ.4) GOTO 100	20011270
	IF(JTYPE.EQ.0) GOTO 100	20011280
C		20011290
C-----	IF IN A GUR PACKET, GET KEY	20011300
	PKT= NPKT(JNT)	20011310
	IF(PKT.EQ.0) GOTO 20	20011320
	IF(PKT.EQ.PKTO) GOTO 20	20011330
C-----	USE AN UNUSED KEY SLOT	20011340
	JKEY=KEYEND(0)	20011350
	IF(JKEY.NE.J) GOTO 15	20011360
10	JNCORE=1+JNCORE	20011370
	JKEY=JNCORE	20011380

15	KORG= M*JKEY-M	20011390
	CALL INPCKD(KEYS(PKT1)/100,AJ(KORG+1),JKEY)	20011400
	DJ(JKEY)= DOTS(M,B(IPI),AJ(KORG+1))	20011410
	PKTO=PKT	20011420
C		20011430
C-----	NOW GET COLUMN AND DJ.	20011440
20	CONTINUE	20011450
	CALL IN(JNT, AJ(JORG+1), JCOL)	20011460
	DJ(JCOL)= DOTS(M,B(IPI),AJ(JORG+1))	20011470
C		20011480
C-----	CORRECT FOR PACKET AND BOUND EFFECTS	20011490
	DJNEW = DJ(JCOL)	20011500
	IF(PKT.NE.0) DJNEW = DJNEW - DJ(JKEY)	20011510
	IF(JTYPE.EQ.3)DJNEW =-DJNEW	20011520
	IF(DJNEW.LT.-ZERO) NDJS=1+NDJS	20011530
C		20011540
C-----	SELECTION STAGE INTERCHANGE BEST FOR NEW	20011550
	IF(DJNEW.GE.DJOLD) GOTO 100	20011560
	DJOLD=DJNEW	20011570
	I=ICOL	20011580
	ICOL=JCOL	20011590
	JCOL=I	20011600
	I=IORG	20011610
	IORG=JORG	20011620
	JORG=I	20011630
100	CONTINUE	20011640
C		20011650
	IF(DJOLD.GT.-DJTOL) GOTO 999	20011660
C		20011670
C-----	PRESERVE THE BEST	20011680
	JNCORE=1+JNCORE	20011690
	NCORE=1+NCORE	20011700
	ICOL = NCORE	20011710
	IORG = M*ICOL-M	20011720
999	IF(NCORE.GE. NCRMAX) GOTO 110	20011730
1000	CONTINUE	20011740
110	CONTINUE	20011750
	IF(JNCORE.NE.0) JNCORE=NCORE-1	20011760
	GOTO 500	20011770
C		20011780
C-----	ALL IN CORE CASE, READ AND PRICE .	20011790
200	IF(JNCORE.EQ.0) GOTO250	20011800
	JNCORE=0	20011810
	GOTO 500	20011820
C		20011830
250	CONTINUE	20011840
	JORG=0	20011850
	DO 300 JNT=1,NT	20011860
	CALL IN(JNT,AJ(JORG+1),JNCORE+1)	20011870
	IF(NULL(JNT))GOTO 300	20011880
	JNCORE=1+JNCORE	20011890
	DJ(JNCORE)= DOTS(M,B(IPI), AJ(JORG+1))	20011900
	JORG = M+JORG	20011910
300	CONTINUE	20011920
	GOTO 500	20011930
C		20011940
C-----	DIAGNOSTICS IF K3*23	20011950
500	CONTINUE	20011960

```
IF( MOD(K3,23).NE.0 ) RETURN
WRITE(6,501) (JA(J), DJ(J), J=1, JNCORF )
501 FORMAT(* DISC-PROVIDED*/(8( I5, F10.2 ) )
RETURN
END
```

```
20011970
20011980
20011990
20012000
20012010
```

	FUNCTION DOT(M,X,Y)	20012020
C-----	INNER PRODUCT OF X AND Y	20012030
	DOUBLE PRECISION SUM	20012040
	REAL X(1),Y(1)	20012050
	SUM=0.0	20012060
	DO 100 I=1,M	20012070
	IF(Y(I).EQ.0.0) GOTO 100	20012080
	SUM=SUM+X(I)*Y(I)	20012090
100	CONTINUE	20012100
	DOT = SUM	20012110
	RETURN	20012120
C		20012130
C-----	SINGLE PRECISION VERSION FOR SPEED	20012140
	ENTRY DOTS	20012150
	DOT=0.0	20012160
	DO 200 I=1,M	20012170
200	DOT=DOT+X(I)*Y(I)	20012180
	RETURN	20012190
	END	20012200

	SUBROUTINE ESCAPE(R)	20012210
C-----	GUP VERSTON APRIL/71	20012220
	COMMON /PAPAMS/ TMAX, ITNINV, INVF, K1, K2, K3, K4, K5	20012230
	COMMON /LIMS/ MAXTOP, MTPV, JNCOPE, NCPMAX, NSCAN	20012240
	COMMON /T/ M, L, MPL, MC, NT, ICOST, IC, IPHASE, JPHS, IPT	20012250
	COMMON /BASIS/ IBASIS(101), KEYS(101)	20012260
	COMMON /NAMES/ NAME(100)	20012270
	COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)	20012280
	COMMON /CORE/ JAPFJ(101), JA(101), JAK(101), AJ(1000)	20012290
	COMMON /DJJ/ DJ(100)	20012300
	DATA AALPHA/5HALPHA/, ABETA/4HBETA/, AGAMMA/5HGAMMA/, ADT/2HOT/	20012310
	DATA AJAPEJ/5HJAPEJ/, AJA/2HJA/, ANAME/4HNAME/, ARBASIS/5HRBASIS/	20012320
	DATA ADDELTA/5HDELTA/, ADJ/2HDJ/, AKEY/3HKEY/	20012330
	COMMON /PHS/ PHS(100)	20012340
	MPKT(J)=MOD(NAME(J), 100000)/1)	20012350
	CALL MESSG(4) (HESCAPED AFTER DUMP OF LP SYSTEM)	20012360
	WRITE(6, 3) ANAME	20012370
	WRITE(6, 2) (NAME(J), J=1, NT)	20012380
	WRITE(6, 3) ARBASIS	20012390
	WRITE(6, 2) (IBASIS(J), J=1, M)	20012400
	WRITE(6, 3) AKEY	20012410
	WRITE(6, 2) (KEYS(I), I=1, L)	20012420
	WRITE(6, 3) AJA	20012430
	WRITE(6, 2) (JA(J), J=1, JNCOPE)	20012440
	WRITE(6, 3) AJAPEJ	20012450
	WRITE(6, 2) (JAPFJ(I), I=1, JNCOPE)	20012460
	WRITE(6, 3) AALPHA	20012470
	WRITE(6, 1) (ALPHA(J), J=1, MPL)	20012480
	WRITE(6, 3) ABETA	20012490
	WRITE(6, 1) (BETA(J), J=1, MPL)	20012500
	WRITE(6, 3) AGAMMA	20012510
	WRITE(6, 1) (GAMMA(J), J=1, M)	20012520
	WRITE(6, 3) ADDELTA	20012530
	WRITE(6, 1) (DELTA(J), J=1, M)	20012540
	WRITE(6, 3) ADJ	20012550
	WRITE(6, 1) (DJ(J), J=1, JNCOPE)	20012560
1	FORMAT(14, 10F12.5)	20012570
2	FORMAT(14, 10I12)	20012580
3	FORMAT(140, A10)	20012590
	CALL MAPOUT(R)	20012600
C-----	CAUSE A DUMP	20012610
	I=0	20012620
	WRITE(1) I	20012630
	RETURN	20012640
	END	20012650

	SUBROUTINE EXITS	20012660
	COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ	20012670
C-----	PRIMAL CALLS THESE ENTRY POINTS AT THE END OF EACH PHASE	20012680
C-----		20012690
	ENTRY OPT1	20012700
	NPHASE=1	20012710
	RETURN	20012720
C-----		20012730
	ENTRY OPT2	20012740
	NPHASE=2	20012750
	NF1 = 1	20012760
	CALL STATUS(40HPRIMAL--END OF PHASE 2--OPTIMAL)	20012770
	RETURN	20012780
C-----		20012790
	ENTRY UNBND	20012800
	NPHASE=4	20012810
	CALL STATUS(40HPRIMAL--UNBOUNDED SOLUTION)	20012820
	RETURN	20012830
C-----		20012840
	ENTRY NOFEAS	20012850
	NPHASE = 5	20012860
	CALL STATUS(40HPRIMAL--NO FEASIBLE SOLUTION)	20012870
	RETURN	20012880
C-----		20012890
	END	20012900

```

SUBROUTINE FFASCH(P)
C-----GJR VERSION APRIL/71
C-----GIVEN CURRENT INVERSE R, PHS, KEY AND ROUNDS IN GAMMA
C-----COMPUTES CURRENT SOLUTION BETA, NO. OF INFEASIBLES NPIF.
C-----IF BETA INFEAS, ADDS ARTIFICIALS AND REVERTS TO PHASE-1
COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
COMMON /STATE/ JPOS,TRON,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
COMMON /BASIS/ IBASIS(101),KEYS(101)
COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,TPI
COMMON /COPE/ JAPFJ(101),JA(101),JAK(101),AJ(1000)
COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DEXTOL
COMMON /ROUNDS/ ROUNDS(100),IBDS(100),NRDS
COMMON /NAMES/ NAME(100)
COMMON /PHS/ PHS(100)
LOGICAL KEY
REAL R(1)
NPKT(I)=MOD(NAME(I),100000) / 10
ITP(J)=MOD(IBASIS(J),100)
KEY=.FALSE.
DO 10 T=1,M
10 DELTA(T)=0.0
DELTA(M)=1.0
C-----COMPUTE EFFECTIVE PHS IN GAMMA USING KEY + ROUNDS ALREADY THERE
DO 20 T=1,M
20 GAMMA(T)=+GAMMA(T)+PHS(T)
C-----CYCLE ELEMENTS OF SOLUTION
NPIF=0
SUMIF=0.
RNDJ=1.E70
IORG=1
MM1=M-1
DO 100 T=1,MM1
SUM = DOT(M, R(IORG), GAMMA )
BETA(T)=SUM
IORG= M+IORG
JOUT= IBASIS(T)/100
IF(NRDS.EQ.0) GOTO 50
C-----CHECK XJOUT LESS THAN ROUND
RNDJ = ROUND(JOUT)
IF( SUM.LE.RNDJ+ZERO) GOTO 50
C-----ROUND VIOLATED, REMOVE ROUND AND TREAT AS INFEAS XJOUT
IF(MOD(K3,13).EQ.0) WRITE(6,101) I,JOUT,SUM,RNDJ
CALL SETBND (JOUT)
BETA(I)=BETA(I)-RNDJ
GOTO 60
C-----CHECK XJOUT POSITIVE FOR NON-FREE POWS
50 IF( ITP (I).EQ.7 ) GOTO 100
IF( BETA(I).GE.-ZERO) GOTO 100
C-----INFEASIBLE, ADD ARTIFICIAL TO KILL ERROR AND PIVOT IN
IF(MOD(K3,13).EQ.0) WRITE(6,101) I,JOUT,SUM,RNDJ
55 CONTINUE
CALL SETBND(-JOUT)
BETA(I) = -BETA(I)
60 NPIF=1+NPIF
JPKT=0
IF(JOUT.LE.NT) JPKT=NPKT(JOUT)

```

	IF(JPKT.NE.0) KEYS(JPKT)=KEYS(JPKT)-1	20013480
	SUMIF=SUMIF+BETA(I)	20013490
	IBASIS(I)=100*(100+JOUT+NT)+MOD(IBASIS(I),100)	20013500
	DELTA(I)=-1.	20013510
	IF(JOUT.GT.NT) DELTA(M)=2.	20013520
	CALL PIVOT(I,R,DELTA)	20013530
	DELTA(I)=0.	20013540
	DELTA(M)=1.	20013550
	IF(KEY) GOTO 150	20013560
100	CONTINUE	20013570
101	FORMAT(* INFEAS--ROW*I5* COL*I5* VALUE*E12.4* BOUND*E12.4)	20013580
C----	CONSTRUCT COMPLETE SOLUTION FOR KEYS IN BETA(M+1).....	20013590
	IF(L.EQ.0) GOTO 151	20013600
	BNDJ=1.E70	20013610
	KEY=.TRUE.	20013620
	DO 150 K=1,L	20013630
C----	CHECK PACKET HAS BASIC COLS	20013640
	SUM=0.0	20013650
	NR = MOD (KEYS(K),100)	20013660
	IF(NR .EQ.0) GO TO 115	20013670
C----	SUM BASIC COL VALUES IN PACKET K	20013680
	DO 110 I=1,M	20013690
	JPOS = IBASIS(I)/100	20013700
	IF(NPKT(JPOS).EQ.K) SUM=SUM+BETA(I)	20013710
110	CONTINUE	20013720
115	BETA(M+K) = RHS(M+K)-SUM	20013730
	IF(BETA(M+K).GE.-ZERO) GOTO 150	20013740
C----	INFEASIBLE GUB, MUST BE ESSENTIAL, CHANGE TO BASIC ROW I	20013750
	JOUT=KEYS(K)/100	20013760
	IF(MOD(K3,13).EQ.0) WRITE(6,111) K,JOUT,BETA(M+K),BNDJ	20013770
111	FORMAT(* INFEAS--GUB*I5* COL*I5* VALUE*E12.4* BOUND*E12.4)	20013780
	CALL KEYCH(JOUT,I,R)	20013790
	GOTO 55	20013800
150	CONTINUE	20013810
151	CONTINUE	20013820
C----	TOTAL INFEASIBILITY	20013830
	BETA(M)=0.	20013840
	DO 160 I=1,MM1	20013850
	IF(IRASIS(I)/100.LE.NT) GOTO 160	20013860
	BETA(M)=BETA(M)-BETA(I)	20013870
160	CONTINUE	20013880
C----	INDICATE PHASE 1	20013890
	IF(ABS(BETA(M)).LT.CTOL) GOTO 200	20013900
	IPHASE = 1	20013910
	IF(MOD(K3,13).EQ.0) WRITE(6,199) SUMIF,BETA(M)	20013920
199	FORMAT(* TOTAL INFEASIBILITY*E12.4* PHASE1 COST*E12.4)	20013930
C----	ADJUST COST POW IC AND ORIGIN IPI	20013940
200	IC = ICOST	20013950
	IF(IPHASE.EQ.1) IC = M	20013960
	IPI=1+M*IC-M	20013970
C----	PHASE 1 COST IS EQUALITY ZERO IN PHASE 2	20013980
	IBASIS(M)=100*(IBASIS(M)/100)	20013990
	IF(IPHASE.EQ.1) IBASIS(M)=IBASIS(M)+3	20014000
	RETURN	20014010
	END	20014020

	SUBROUTINE INVERT(R)	20014030
C-----	GUR VERSION APRIL/71	20014040
	COMMON /ROUNDS/ ROUNDS(100),IBDS(100),NRDS	20014050
	COMMON /STATE/ JPOS,TRPW,JCOL,JOUT,ITRN,NREJ,NPTE,NDJS	20014060
	COMMON /TOLS/ DJTOL,TEPO,PIVTOL,CTOL,PERTOL,DEPTOL	20014070
	COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,ITNCHK	20014080
	COMMON /BASIS/ IBASIS(101),KEYS(101)	20014090
	COMMON /A/ ALPHA(101) /B/ BETA(1 1) /C/ GAMMA(101) /D/ DELTA(101)	20014100
	COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NORMAX,ISCAN	20014110
	COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,JPHASE,JPHS,IPI	20014120
	COMMON /CORE/ JAPFJ(101),JA(101),JAK(101),AJ(100)	20014130
	COMMON /NAMES/ NAME(100)	20014140
	COMMON /PHS/ PHS(100)	20014150
	REAL R(1)	20014160
	INTEGER PKT,PKTO	20014170
	LOGICAL BASIC,ATRND	20014180
	ITP(J)=MOD(IBASIS(J),100)	20014190
	NPKT(J)=MOD(NAME(J),100000)/10	20014200
C-----	INVERTS CURRENT BASIS VECTORS AND ADJUSTS FOR ROUNDS	20014210
	IF(ITRN.LT.ITNINV) RETURN	20014220
	CALL MESSG(4)HTINVERT	20014230
C-----	ITERATION OF NEXT INVERT	20014240
	ITNINV=ITRN+INVF	20014250
C		20014260
	IF(L.EQ.0) GOTO 15	20014270
C-----	COUNT MISSING KEYS TO NKM AND CLEAR BASIC COUNT	20014280
	NKM=0	20014290
0	DO 5 I=1,L	20014300
	K=KEYS(I)/100	20014310
	IF(K.EQ.0) NKM=NKM+1	20014320
	KEYS(I)=100*K	20014330
5	CONTINUE	20014340
	IF(NKM.EQ.0) GOTO 15	20014350
C-----	FIRST SCAN BASIC COLS FOR KEY CANDIDATES	20014360
	JTAG=2	20014370
6	CONTINUE	20014380
	DO 10 J=MC,NT	20014390
	K=NPKT(J)	20014400
	IF(K.EQ.0) GOTO 11	20014410
	JTYPE=MOD(NAME(J),10)	20014420
	IF(JTYPE.NE.JTAG) GOTO 10	20014430
	IF(KEYS(K)/100.NE.0) GOTO 10	20014440
C-----	SET COLUMN J KEY IN PACKET K AND REDUCE COUNT NKM	20014450
	KEYS(K)=100*J	20014460
	CALL SETKEY(J)	20014470
	NKM=NKM-1	20014480
10	CONTINUE	20014490
	IF(NKM.EQ.0) GOTO 15	20014500
	IF(JTAG.NE.2) GOTO 11	20014510
C-----	NOW EXAMINE FREE COLUMNS	20014520
	JTAG=1	20014530
	GOTO 6	20014540
11	CONTINUE	20014550
	CALL ERROR(4)HTINVERT--SEVERAL KEYS UNMARKED--DATA ERR.	20014560
	CALL ESCAPE(R)	20014570
C		20014580
C-----	NULL BASIS EXCEPT FOR FREE POWS SAVING TYPES	20014590

15	CONTINUE	20014600
	IORG=J	20014610
	DO 20 I=1,M	20014620
	GAMMA(I)=0.0	20014630
	ITYPE=MOD(IBASIS(I),100)	20014640
	IF(ITYPE.NE.3) IBASIS(I)=ITYPE	20014650
C-----	SET UP UNIT BASIS AND ZERO RHS	20014660
	DO 19 J=1,M	20014670
19	B(IORG+J)=0.	20014680
	B(IORG+I)=1.0	20014690
20	IORG=IORG+M	20014700
C-----	RESTORE PHASE1 LOGICAL	20014710
	IBASIS(M)=100*MC+ITYPE	20014720
C		20014730
C-----	CYCLE COLUMN NAMES, KEY COLUMNS TO KORG, OTHERS TO JORG	20014740
	JORG=M*JNCORE	20014750
	KORG=JORG+M	20014760
C-----	PKT IS CURRENT GUP PACKET, PKTO IS PACKET OF LAST KEY	20014770
	PKTO=0	20014780
	DO 200 JNT=1,NT	20014790
	JTYPE= MOD(NAME(JNT),10)	20014800
	IF(JTYPE.LE.1) GOTO 200	20014810
C-----	GET THIS BASIC/ROUNDED/KEY COL TO CORE	20014820
	JPOS=JNT	20014830
30	CALL IN(JPOS,AJ(JORG+1),JNCORE+1)	20014840
	PKT=NPKT(JNT)	20014850
	IF(JTYPE.GE.3) GOTO 150	20014860
C-----	BASIC COLUMN, IS KEY NEEDED	20014870
	IF(PKT.EQ.0) GOTO 120	20014880
	IF(PKT.EQ.PKTO) GOTO 100	20014890
C-----	GET KEY AND RECORD	20014900
	CALL INPKD(KEYS(PKT)/100,AJ(KORG+1),JNCORE+2)	20014910
	PKTO=PKT	20014920
C-----	REMOVE KEY COMPONENT FROM COL	20014930
100	DO 110 I=1,M	20014940
110	AJ(JORG+I)=AJ(JORG+I)-AJ(KORG+I)	20014950
C-----	TRANSFORM TO CURRENT BASIS	20014960
120	IORG=1	20014970
	DO 130 I=1,M	20014980
	ALPHA(I)=DOT(M,B(IORG),AJ(JORG+1))	20014990
130	IORG=IORG+M	20015000
C-----	FIND BEST ROW TO PIVOT	20015010
	IROW=0	20015020
	CALL PIVOT(IROW,B,ALPHA)	20015030
	IF(IROW.EQ.0) GOTO 200	20015040
C-----	INCREASE COUNT OF BASIC COLS IN PACKET	20015050
	IF(PKT.NE.0) KEYS(PKT)=KEYS(PKT)+1	20015060
	GOTO 200	20015070
C		20015080
C-----	PICK UP ROUND OR PH5 OF PACKET	20015090
150	IF(PKT.NE.0) GOTO 155	20015100
	BNDJ=BOUND(JPOS)	20015110
	GOTO 156	20015120
155	BNDJ=RHS(PKT+M)	20015130
156	DO 160 J=1,M	20015140
160	GAMMA(J) = GAMMA(J) - AJ(JORG+J)*BNDJ	20015150
200	CONTINUE	20015160
C		20015170

C-----COMPLETE BASIS WITH ARTIFICIALS	20015180
C-----COUNT LOGICALS IN JL	20015190
JL=0	20015200
DO 210 I=1,M	20015210
IF(MOD(IRASTS(I),100).NE.0) JL=JL+1	20015220
IF(IRASIS(I)/100.NE.0) GOTO 210	20015230
IF(MOD(IRASTS(I),100) .NE.1) GOTO 205	20015240
C-----MAKE A LOGICAL BASIC INSTEAD	20015250
IRASTS(I)=100*JL+IRASTS(I)	20015260
GOTO 210	20015270
205 CONTINUE	20015280
IRASIS(I)=100*(I+NT)+IRASIS(I)	20015290
210 CONTINUE	20015300
C-----ADD ARTIFICIALS NEEDED	20015310
DO 220 I=1,M	20015320
220 DELTA(I)=0.0	20015330
DELTA(M)=1.0	20015340
DO 240 I=1,M	20015350
IF(IRASIS(I)/100.LE.NT) GOTO 240	20015360
C-----ONE NEEDED ROW I	20015370
DELTA(I)=1.0	20015380
IOPG=1	20015390
DO 230 J=1,M	20015400
ALPHA(J)= DDT(M,R(IOPG),DELTA)	20015410
230 IOPG=IOPG+M	20015420
DELTA(I)=0.0	20015430
CALL PIVOT(I,R,ALPHA)	20015440
240 CONTINUE	20015450
C	20015460
C-----NOW USE R AND GAMMA TO GET SOLUTION TO BETA	20015470
CALL FEASCH(R)	20015480
RETURN	20015490
END	20015500

```

SUBROUTINE IO(KOL, ALPHA, NAME) 20015510
C-----GUR VERSION--APRIL-20-1971 20015520
C-----WRITES TWO FILES OF A MATRIX TO DISC IN STRAIGHT OR PACKED FORM 20015530
COMMON /FILES/ IA1,IA2,INAP 20015540
COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI 20015550
COMMON /ROWTYP/ IROWTP(101) 20015560
COMMON /LIMS/ MAXTPY,NTRY,JNCORE,NORMAX,NSCAN 20015570
COMMON /CORE/ JAPEJ(101),JA(101),JAK(101),AJ(1000) 20015580
DIMENSION ALPHA(1),ID(100),D(100) 20015590
COMMON /B/ ALPHB(100) 20015600
DATA ZERO/1.E-10/ 20015610
C 20015620
ENTRY OUT 20015630
C-----TO DISC FILES IA1/IA2 20015640
C 20015650
IF (KOL.GT.1) GOTO 10 20015660
REWIND IA1 20015670
REWIND IA2 20015680
KOL1=KOL2=0 20015690
C 20015700
C-----STRIP GUBS AND PACK FOR TWO FILES 20015710
10 J=K=0 20015720
DO 20 I=1, M 20015730
IF( IROWTP(I).EQ.4 ) GOTO 20 20015740
J=J+1 20015750
ALPHB(J)=ALPHA(I) 20015760
IF(ABS(ALPHA(I)).LT.ZERO) GOTO 20 20015770
K=K+1 20015780
ID(K)=J 20015790
D(K)=ALPHA(I) 20015800
20 CONTINUE 20015810
NAME=KOL 20015820
WRITE(IA1) KOL,NAME,(ALPHB(I),I=1,J) 20015830
WRITE(IA2) KOL,NAME,K,(ID(I),D(I),I=1,K) 20015840
RETURN 20015850
C 20015860
C 20015870
ENTRY IN 20015880
C-----FOR NORMAL COLUMNS FROM DISC IA1 20015890
DO 100 JNT=1,NT 20015900
IF(MOD(KOL1,NT).NE.0) GOTO 110 20015910
99 REWIND IA1 20015920
REWIND IA2 20015930
NSCAN =1+NSCAN 20015940
110 READ(IA1) KOL1,NAAM,(ALPHA(I),I=1,M) 20015950
IF(KOL.LT.KOL1) GOTO 99 20015960
IF(KOL.EQ.KOL1) GOTO 101 20015970
100 CONTINUE 20015980
GOTO 300 20015990
101 CONTINUE 20016000
C-----UPDATE RECORDS AND TRACK DISC LOCATION IN KOL 20016010
120 CONTINUE 20016020
JCOL=NAME 20016030
JA(JCOL)=KOL 20016040
JAK(JCOL)= NAAM 20016050
JAREJ(JCOL)=0 20016060
RETURN 20016070

```

C		20016080
C		20016090
	ENTRY INPOS	20016100
C-----	TO GET THE INPUT FILE POSITION	20016110
	KOL=KOL1	20016120
	RETURN	20016130
C		20016140
C		20016150
	ENTRY INPKO	20016160
C-----	AUXILIARY FILE FOR KEYS	20016170
	DO 200 JNT=1,NT	20016180
	IF(MOD(KOL2,NT).NE.0) GOTO 199	20016190
195	REWIND TA2	20016200
199	READ(IA2) KOL2,NAAM,K,(ID(I),D(I),I=1,K)	20016210
	IF(KOL.LT.KOL2) GOTO 195	20016220
	IF(KOL2.EQ.KOL) GOTO 201	20016230
200	CONTINUE	20016240
	GOTO 300	20016250
201	CONTINUE	20016260
C-----	UNPACK D TO ALPHA	20016270
	DO 210 I=1,M	20016280
210	ALPHA(I)=0.	20016290
	IF(K.EQ.0) GOTO 120	20016300
	DO 220 I=1,K	20016310
	J=ID(I)	20016320
220	ALPHA(J)=D(I)	20016330
	GOTO 120	20016340
C		20016350
C-----	TROUBLE	20016360
300	CALL ERROR (4CHIO--COLUMN NOT LOCATED IN NT READS)	20016370
	CALL ESCAPE(R)	20016380
	END	20016390

	SUBROUTINE KEYCH(JCOL,IROW,B)	20016400
C-----	GUR VERSION APRIL/71	20016410
	COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI	20016420
	COMMON /BASIS/ IBASIS(101),KEYS(101)	20016430
	COMMON /NAMES/ NAME(100)	20016440
	COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)	20016450
	REAL B(1)	20016460
	NPKT(J)=MOD(NAME(J),100000)/10	20016470
	ITP(J)=MOD(IBASIS(J),100)	20016480
C		20016490
J-----	INTERCHANGE KEY WITH FIRST BASIC COL IN KEYS PACKET	20016500
C	RETURNING ROW OF NEW BASIC COL (OLD KEY)	20016510
	JCOLPK= NPKT(JCOL)	20016520
C-----	FIRST BASIC COL	20016530
	DO 10 I=1,M	20016540
	JKEY = IBASIS(I)/100	20016550
	IF(JKEY.GT.NT) GOTO 10	20016560
	IF(NPKT(JKEY).EQ.JCOLPK) GOTO 20	20016570
10	CONTINUE	20016580
	CALL ERROR(40HKEYCH--ESSENTIAL PACKET NO BASIC COL)	20016590
	CALL ESCAPE	20016600
C		20016610
C-----	RE-DIFFERENCE BASIS INVERSE TO MAKE JKEY KEY	20016620
20	IROW = I	20016630
	JORG = M*IROW-M	20016640
	DO 30 J=1,M	20016650
30	B(JORG+J) =-B(JORG+J)	20016660
	IORG = 0	20016670
	DO 50 I=1,M	20016680
	IF(I.EQ.IROW) GOTO 50	20016690
	IB=IBASIS(I)/100	20016700
	IF(IB.GT.NT) GOTO 50	20016710
	IF(NPKT(IB).NE.JCOLPK) GOTO 50	20016720
	DO 40 J=1,M	20016730
40	B(JORG+J)=B(JORG+J)-B(IORG+J)	20016740
50	IORG=IORG+M	20016750
C		20016760
C-----	NEW KEY IS NOW JKEY	20016770
	CALL SETKEY(JKEY)	20016780
	KEYS(JCOLPK)= 100*JKEY+ MOD(KEYS(JCOLPK),100)	20016790
C-----	COL JCOL IS NOW BASIC	20016800
	CALL SETBNB(JCOL)	20016810
	IBASIS(IROW)=100*JCOL+ITP(IROW)	20016820
C-----	REARRANGE SOLUTION	20016830
	MPK=M+JCOLPK	20016840
	SUM=ALPHA(IROW)	20016850
	ALPHA(IROW)=ALPHA(MPK)	20016860
	ALPHA(MPK)=SUM	20016870
	SUM= BETA(IROW)	20016880
	BETA(IROW)=BETA(MPK)	20016890
	BETA(MPK)=SUM	20016900
	RETURN	20016910
	END	20016920

	FUNCTION KEYFND(PKT)	20016930
	COMMON /CORE/ JAPRJ(101),JA(101),JAK(101),AJ(1000)	20016940
	COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN	20016950
	COMMON /NAMES/ NAME(100)	20016960
	COMMON /BASIS/ IPASIS(101),KEYS(101)	20016970
	INTEGER PKT	20016980
	NPKT(I)=MOD(NAME(I),100000)/10	20016990
C		20017000
C-----	GIVEN PACKET NO. PKT, FIND ITS KEY IN CORE	20017010
	IF(PKT.EQ.0) GOTO 100	20017020
	KEY=KEYS(PKT)/100	20017030
	DO 20 K=1,JNCORE	20017040
	IF(JA(K).EQ.KEY) GOTO 30	20017050
20	CONTINUE	20017060
	KEYFND=0	20017070
	RETURN	20017080
30	KEYFND=K	20017090
	RETURN	20017100
C		20017110
C-----	FIND THE FIRST KEY WITH NO COLUMNS IN CORE FOR CHECK	20017120
100	DO 130 K=1,JNCORE	20017130
	JAK=JA(K)	20017140
	JTYPE=MOD(NAME(JAK),10)	20017150
	IF(JTYPE.NE.4) GOTO 130	20017160
	JPKT=NPKT(JAK)	20017170
	DO 120 J=1,JNCORE	20017180
	JAJ=JA(J)	20017190
	JTYPE=MOD(NAME(JAJ),10)	20017200
	IF(JTYPE.EQ.4) GOTO 120	20017210
	IF(JPKT.EQ.NPKT(JAJ)) GOTO 130	20017220
120	CONTINUE	20017230
	GOTO 30	20017240
130	CONTINUE	20017250
	KEYFND=0	20017260
	RETURN	20017270
	END	20017280

```

SUBROUTINE MAPIN(R)
C-----GUB VERSION APRIL 20/71
C-----ADDS SPECS FOR BOUND/BASIC/NULL/KEY VARIABLES AND INVERSE IF PRESE
C-----OPTIONALLY CALLED BEFORE INVERT
COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
COMMON /BASIS/ IRASIS(101),KEYS(101)
COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
COMMON /FILES/ IA1,IA2,IMAP
COMMON /INPUT/ INPUT,INPUTM,INPUTN
COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DEXTOL
COMMON /NAMES/ NAME(100)
COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS
COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
REAL B(1)
DIMENSION CARD(8)
INTEGER NAMES(5)
INTEGER PKT
REAL NULL,KEE,INVERS
DATA BASIC/5HBASIC/,ATBND/5HATBND/,ENDER/5HEND /,ROWS/4HPOMS/
+ ,NULL/4HNULL/,KEE/3HKEY/,INVERS/5HINVER/
+ ,REWIND/6HREWIND/
ITP(J)=MOD(IRASIS(J),100)
NPKT(I)=MOD(NAME(I),100000)/10
C-----SETS BASIC COLUMNS AND LOGICALS.
CALL MESSG(40HMAPIN )
REWIND IMAP
10 READ(IMAP,11) TYPE1,TYPE2,(NAMES(J),J=1,4)
11 FORMAT(2A5,4I10)
IF(MOD(K3,3).EQ.0) WRITE(6,12) TYPE1,TYPE2,(NAMES(J),J=1,4)
12 FORMAT(X,2A5,4I10)
IF(TYPE1.EQ.BASIC) GOTO 30
IF(TYPE1.EQ.KEE) GOTO 50
IF(TYPE1.EQ.ATBND) GOTO 15
IF(TYPE1.EQ.NULL) GOTO 80
IF(TYPE1.EQ.INVERS) GOTO 95
IF(TYPE1.EQ.ENDER) RETURN
CALL ERROR(40HMAPIN--UNRECOGNIZED TYPE CARD IN DATA )
RETURN
C
C-----ADD AT BOUND COLUMN SPECS
15 DO 20 J=1,4
ID=NAMES(J)
IF(ID.EQ.0) GOTO 20
ID=ID+MC
BNDJ=BOUND(ID)
IF(BNDJ.LT.1.E8) GOTO 19
CALL ERROR(40HMAPIN--ATBND COLUMN NOT ROUNDED IBDS/BDS )
CALL DUMP(IBDS(1),IBDS(NBDS),2,BOUNDS(1),BOUNDS(NBDS),1)
GOTO 20
19 CONTINUE
CALL SETBND(ID)
20 CONTINUE
GOTO 10
C
C-----BASIC COLUMNS ADDED
30 IF(TYPE2.EQ.ROWS) GOTO 60

```


DO 40 J=1,4	200178F0
ID=NAMES(J)	2001787J
IF(ID.EQ.0) GOTO 40	20017880
ID=ID+MC	20017890
CALL SETPNR(ID)	20017900
40 CONTINUE	20017910
GOTO 10	20017920
C	20017930
C-----ENTER KEY COLUMNS IF A SUB PROBLEM	20017940
50 IF(L.EQ.0) GOTO 30	20017950
DO 55 I=1,4	20017960
ID=NAMES(I)	20017970
IF (ID.EQ.0) GOTO 10	20017980
ID=ID+MC	20017990
PKT=NPKT(ID)	20018000
IF(PKT.EQ.0) GOTO 55	20018010
JOUT=KEYS(PKT)/100	20018020
IF(JOUT.NE.0) CALL SETKEY(-JOUT)	20018030
CALL SETKEY(ID)	20018040
KEYS(PKT)=100*ID	20018050
55 CONTINUE	20018060
GOTO 10	20018070
C	20018080
C-----BASIC ROW-COL DATA FOR ENTRY OF ROW LOGICALS	20018090
60 DO 70 J=1,4	20018100
ID=NAMES(J)	20018110
IF(ID.EQ.0) GOTO 70	20018120
CALL SETPNR(ID)	20018130
70 CONTINUE	20018140
GOTO 10	20018150
C	20018160
C-----SET NULL COLUMNS	20018170
80 DO 90 J=1,4	20018180
ID=NAMES(J)	20018190
IF(ID.EQ.0) GOTO 90	20018200
ID=ID+MC	20018210
CALL SETNNN(ID)	20018220
90 CONTINUE	20018230
GOTO 10	20018240
C	20018250
C-----CHECK FOR INVERSE AT END OF INPUT TAPE OR SKIP	20018260
95 MM=MM	20018270
READ(INPUT) (R(J),J=1,MM)	20018280
IF(ENDFILE INPUT) 10,96	20018290
96 READ(INPUT) (TRASTS(J),PETA(J),J=1,M)	20018300
IF(ENDFILE INPUT) 10,97	20018310
97 IF(L.NE.0) READ(INPUT) (KEYS(J),PETA(J+M),J=1,L)	20018320
IF(ENDFILE INPUT) 10,98	20018330
C-----SUCCESSFULL, SUPPRESS INVERT	20018340
98 ITHINV=INVF/2	20018350
CALL MSSG(40HSTARTED FROM GIVEN INVERSE ON INPUT	20018360
C-----RESET INPUT FILE FOR MAPOUT TO OVERWRITE LAST INVERSE	20018370
BACKSPACE INPUT	20018380
BACKSPACE INPUT	20018390
IF(L.EQ.0) GOTO 10	20018400
BACKSPACE INPUT	20018410
GOTO 10	20018420
C	20018430

C	ENTRY INMAP	20018440
C-----	READS MAP CARDS FROM INPUT TO FILE IMAP AND TERMINATES THEM	20018450
	CALL MESSG(40HINMAP LOOKED FOR MAP	20018460
	DO 200 I=1,1000	20018470
	READ(5,2) CARD	20018480
	IF(ENDFILE 5) 201,199	20018490
199	CONTINUE	20018500
2	FORMAT(8A10)	20018510
	IF (CARD(1).EQ.ENDER) GOTO 201	20018520
	IF(CARD(1).EQ.REWTND) GOTO 202	20018530
	WRITE(IMAP,2) CARD	20018540
1999	CONTINUE	20018550
200	CONTINUE	20018560
C-----	ENDS THE MAPOUT CARDS	20018570
201	WRITE(IMAP,2) ENDFR	20018580
	RETURN	20018590
202	REWIND IMAP	20018600
	CALL MESSG(40HINMAP--DELETED EXISTING MAP, IF ANY	20018610
	GOTO 1999	20018620
	END	20018630
		20018640

	SUBROUTINE MAPOUT(R)	20018650
C-----	GUR VERSION APRIL-20-71	20018660
C-----	OUTPUTS THE FINAL BASIS FOR MARTIN USE	20018670
C		20018680
	COMMON /NAMES/ NAME(100)	20018690
	COMMON /BASIS/ BASIS(101),KEYS(101)	20018700
	COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)	20018710
	COMMON /IXX/ IY(100) /XX/ X(100)	20018720
	COMMON /INPUT/ INPUT, INPUTM, INPUTN	20018730
	COMMON /FILES/ IA1, IA2, IMAF	20018740
	COMMON /I/ M, L, MPI, MC, NT, ICOST, IC, IPHASE, JRHS, IPI	20018750
	COMMON /LIMS/ MAXTPY, NTPY, JNCOPE, NCPMAX	20018760
	COMMON /CORE/ JAPFJ(101), JA(101), JAK(101), AJ(1000)	20018770
	COMMON /PARAMS/ TMAX, ITNINV, TNVF, K1, K2, K3, K4, K5	20018780
	COMMON /ROUNDS/ ROUNDS(100), IRDS(100), NRDS	20018790
	EQUIVALENCE (MAPKEY, AJ)	20018800
	DIMENSION MAPRAS(100), MAP RND(100), MAPNLL(10), MAPKEY(1000)	20018810
	REAL B(1)	20018820
C		20018830
	CALL MESSG(40HMAPOUT)	20018840
	REWIND IMAF	20018850
	JNCOPE=0	20018860
	DO 50 I=1, MC	20018870
	IF(NAME(I).NE.2) GOTO 50	20018880
	WRITE(IMAF,1) I	20018890
1	FORMAT(10H BASICROWS ,I10)	20018900
50	CONTINUE	20018910
C		20018920
	K=INLL=IRND=IRAS=IKFY=0	20018930
C-----	CLEAR SOLUTION SPACE	20018940
	NVARS=TNPJTM+NRDS	20018950
	DO 60 I=1, NVARS	20018960
	IX(I)=0	20018970
60	X(I)=0.	20018980
	MPI=MC+1	20018990
	DO 40 I=MPI, NT	20019000
	JCOL=I-MC	20019010
	J=MOD(NAME(I),10)+1	20019020
	GOTO(10,40,20,30,35),J	20019030
C		20019040
10	INLL=INLL+1	20019050
	MAPNLL(INLL)=JCOL	20019060
	GOTO 40	20019070
20	IRAS=IRAS+1	20019080
	MAPRAS(IRAS)=JCOL	20019090
	DO 25 IP=1, M	20019100
	IF(IRASIS(IP)/100.FQ.T) GOTO 26	20019110
25	CONTINUE	20019120
26	K=K+1	20019130
	IX(K)=JCOL	20019140
	X(K)=BETA(IP)	20019150
	GOTO 40	20019160
30	IRND=IRND+1	20019170
	MAPRND(IRND)=JCOL	20019180
	K=K+1	20019190
	IX(K)=JCOL	20019200
	Y(K)=ROUND(I)	20019210

	GOTO 40	20019220
35	IKEY=IKEY+1	20019230
	MAPKEY(IKEY)=JCOL	20019240
	DO 36 IR=1,L	20019250
	IF(KEYS(IR)/100.EQ.I) GOTO 37	20019260
36	CONTINUE	20019270
37	K=K+1	20019280
	IX(K)=JCOL	20019290
	X(K)=BETA(IR+M)	20019300
40	CONTINUE	20019310
	IF(IBAS.NE.0) WRITE(IMAP,2) (MAPBAS(I),I=1,IBAS)	20019320
	IF(IRND.NE.0) WRITE(IMAP,3) (MAPRND(I),I=1,IRND)	20019330
	IF(INLL.NE.0) WRITE(IMAP,4) (MAPNLL(I),I=1,INLL)	20019340
	IF(IKEY.NE.0) WRITE(IMAP,6) (MAPKEY(I),I=1,IKEY)	20019350
2	FORMAT(104BASIC ,4I10)	20019360
3	FORMAT(104TBND ,4I10)	20019370
4	FORMAT(10HNULL ,4I10)	20019380
5	FORMAT(10HEND)	20019390
6	FORMAT(10HKEY ,4I10)	20019400
7	FORMAT(10HINVERSE)	20019410
	IF(MOD(K3,2).NE.0) GOTO 598	20019420
C-----	PLACE BASIS ON END OF INPUT TAPE, AFTER ANY THERE ALREADY	20019430
	MM=M*M	20019440
	WRITE(INPUT) (R(T),I=1,MM)	20019450
	WRITE(INPUT) (IBASIS(J),BETA(J),J=1,M)	20019460
	IF(L.NE.0) WRITE(INPUT) (KEYS(J),BETA(J+M),J=1,L)	20019470
	WRITE(IMAP,7)	20019480
598	WRITE(IMAP,5)	20019490
599	IF(MOD(K3,5).NE.0) RETURN	20019500
600	WRITE(6,601)	20019510
601	FORMAT(*OCURRENT SOLUTION*/*O BASIS VALUE -PI*)	20019520
	WRITE(6,602) (IBASIS(I),BETA(I),R(IPI+I-1),I=1,M)	20019530
602	FORMAT(I12,2E12.4)	20019540
	WRITE(6,603) (KEYS(I),BETA(I+M),I=1,L)	20019550
603	FORMAT(*O KEYS VALUE*/(I12,E12.4))	20019560
	WRITE(6,604) (IX(I),X(I),I=1,K)	20019570
604	FORMAT(*OSOLUTION VECTOR, PACKED*/(I12,E12.4))	20019580
C-----	PRICE OUT REMAINING VECTORS	20019590
	WRITE(6,701)	20019600
701	FORMAT(*OREMAINING VECTORS*)	20019610
	DO 700 J=MP1,NT	20019620
	JTYPE=MOD(NAME(J),10)	20019630
	IF(JTYPE.EQ.2) GOTO 700	20019640
	CALL IN(J,GAMMA,1)	20019650
	DJVAL=DOTS(M,R(IPI),GAMMA)	20019660
	WRITE(6,702) J,DJVAL,NAME(J)	20019670
702	FORMAT(I12,12X,E12.4,I12)	20019680
700	CONTINUE	20019690
	RETURN	20019700
	END	20019710

	SUBROUTINE PIVOT(IPOW,B,ALPHA)	20019720
C-----	PIVOT ALPHA INTO B ROW IPOW	20019730
	COMMON /NAMES/ NAME(100)	20019740
	COMMON /BASIS/ IBASIS(101),KEYS(101)	20019750
	COMMON /STATE/ JPOS	20019760
	COMMON /T/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,IRHS,IPI	20019770
	COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DEPTOL	20019780
	COMMON /PARAMS/ TMAX,ITNTNV,INVF,K1,K2,K3,K4,K5	20019790
	REAL ALPHA(1),B(1)	20019800
	NPKT(I)=MOD(NAME(I),100000)/10	20019810
C		20019820
C-----	CHECK ALPHA HAS A ROW	20019830
	IF(IPOW.EQ.0) GOTO 90	20019840
C-----	NORMALISE ROW IPOW	20019850
1	CONTINUE	20019860
	IORG=M*(IPOW-1)	20019870
	PIV=1.	20019880
	IF (ALPHA(IPOW).EQ.1.0) GOTO 20	20019890
	IF(ABS(ALPHA(IPOW)).GT.PIVTOL) GOTO 5	20019900
	CALL EPPOR(404PIVOT--PIVOT LESS THAN PIVTOL	20019910
	CALL ESCAPE(B)	20019920
5	PIV = 1.0/ALPHA(IPOW)	20019930
	DO 10 I=1,M	20019940
10	B(IORG+I)= B(IORG+I)*PIV	20019950
C-----	PIVOT OP FOR ROW I, LEAVE ALPHA.	20019960
20	DO 30 I=1,M	20019970
	IF(I.EQ.IPOW) GOTO 30	20019980
	IF(ABS(ALPHA(I)).LT.ZERO) GOTO 30	20019990
	JORG=M*(I-1)	20020000
	PIV=ALPHA(I)	20020010
	DO 25 J=1,M	20020020
25	B(JORG+J)=B(JORG+J)-PIV *B(IORG+J)	20020030
30	CONTINUE	20020040
	RETURN	20020050
C		20020060
C-----	FIND BEST ROW TO PIVOT ALPHA INTO B	20020070
90	CONTINUE	20020080
	PIV=PIVTOL	20020090
	DO 100 I=1,M	20020100
C-----	CHECK FOR FREE LOGICALS	20020110
	JP=IBASIS(I)/100	20020120
	IF(JP.NE.JPOS) GOTO 99	20020130
	IPOW=I	20020140
	GOTO 1	20020150
99	IF(JP.NE.0) GOTO 100	20020160
C-----	ZERO BASIS ENTRY AT I	20020170
	DIVOT=ABS(ALPHA(I))	20020180
	IF(DIVOT.LT.PIV) GOTO 100	20020190
	PIV=DIVOT	20020200
	IPOW=I	20020210
100	CONTINUE	20020220
	IF(IPOW.EQ.0) GOTO 150	20020230
C-----	BEST ROW TO ADD THIS COLUMN IS IPOW	20020240
101	CONTINUE	20020250
	IBASIS(IPOW)=100*JPOS+IBASIS(IPOW)	20020260
	GOTO 1	20020270
C-----	THE COLUMN IS NO GOOD ANYWHERE AT PRESENT, DROP FROM BASIC SET	20020280

```
150 CONTINUE
    CALL SETRNB(-JPOS)
    IF (MOD(K3,13).NE.0) RETURN
    WRITE(6,151) JPOS
151  FORMAT(1H ,*PIVOT DROPPED COLUMN* I6 )
    RETURN
    END
```

```

SUBROUTINE PRIMAL (R)
C-----GUR VERSION APRIL/71
COMMON /MOVES/ THETA,RNDJ,DMAX,PRMLER,DUALER
COMMON /TOLS/ DJTOL,ZEQN,PIVTOL,CTOL,PERTOL,DEPTOL
COMMON /STAT/ JPOS,TRW,JCOL,JOUT,ITRN,NRFJ,NPTF,NDJS
COMMON /PARAMS/ TMAX,ITNINV,INVE,K1,K2,K3,K4,K5
COMMON /LIMS/ MAXTRY,NTRY,JNCOE,NORMAX,NSCAN
COMMON /CORE/ JAPFJ(101),JA(101),JAK(101),AJ(100)
COMMON /T/ M,L,MPL,MC,NT,ICOST,IC,TPHASE,JRHS,IPJ
COMMON /A/ ALPHA(101) /B/ BETA(1 1) /C/ GAMMA(101) /D/ DELTA(101)
COMMON /BASIS/ IRASIS(101),KEYS(101)
COMMON /DJS/ DJ(100)
COMMON /NAMES/ NAME(100)
COMMON /RHS/ RHS(100)
REAL R(1)
LOGICAL BASTC,ATRND
ITP(I)=MOD(IRASIS(I),10)
ATRND(I)=MOD(NAME(I),10).EQ.7
NPKT(J)=MOD(NAME(J),100000)/10
C
C CALL MESSG(40HPRTMAL )
C
TRW=JCOL=NRFJ=NDFG=0
CALL STATUS(40HPRTMAL--BEGIN )
3000 CONTINUE
C-----FIND THE COST ROW
IC=ICOST
IF ( IPHASE.EQ.1 ) IC= M
C-----KEEP PHASE1 COST ZERO IN PHASE 2
C-----PHASE 1 COST IS FREE IN PHASE 1
IRASIS(M)=100*(IRASIS(M)/100)
IF (TPHASE.EQ.1) IRASIS(M)=IRASIS(M)+3
C-----PICK UP NEW PI
IPI=1+M*IC-M
C-----CUTOFF FOR DEGENERACY REJECTS
NDEGLY=0
C
C*****BASIC CYCLE OF 2 PHASE LP*****
1 ITRN=1+ITRN
THETA=RNDJ=IPW=JCOL=JOUT=JPOS=J
C
C-----LOCATE PIVOTAL COLUMN
30 CALL COLUMN( JCOL,R)
IF( JCOL.NE.0) GOTO 50
CALL XCHECK(4HPRIMAL,4HQUIT,P)
C
C-----OPTIMUM, CHECK MODE = PHASE1/ PHASE2/ NOFEAS
IF( TPHASE.EQ.2) GOTO 2000
IF( ABS(BETA(IC)).LT.CTOL) GOTO 1000
CALL NO FEAS
RETURN
C
C-----STEP PROCEDURE FOR IN CORE COLUMN JCOL
50 CONTINUE
C-----LOCATE COLUMN POSITION AND ROUND
JPOS = JA(JCOL)
RNDJ = ROUND(JPOS)

```

C		20020930
C	-----LOCATE PIVOTAL ROW IROW AND STEP THETA	20020940
	CALL ROW(THETA, IROW, JCOL, ITYPE, B)	20020950
	IF(IROW.NE.0) GOTO 60	20020960
	CALL XCHECK(6HPRIMAL, 4HQUIT, B)	20020970
	CALL UNBND	20020980
	RETURN	20020990
C		20021000
C		20021010
C	-----DEGENERACY AND PIVOT CHECKS	20021020
60	IF(ABS(ALPHA(IROW)).GE.PIVTOL) GOTO 65	20021030
	IF(NREJ.LT.2) GOTO 62	20021040
	WRITE(6,61) JA(JCOL), IROW, ALPHA(IROW), BETA(IROW)	20021050
61	FORMAT(20X, *REJECTED COLUMN*I5* ROW*I5* PIVOT*E12.4* RHS*E12.4)	20021060
62	CONTINUE	20021070
	JAREJ(JCOL)=1	20021080
	NREJ = 1+NREJ	20021090
	IF(NREJ.NE.5) GOTO 64	20021100
C	-----RE-INVERT, CLEAR REJECTS AND CONTINUE	20021110
	ITNINV=ITRN	20021120
	CALL INVERT(B)	20021130
	DO 63 J=1, JNCORE	20021140
63	JAREJ(J)=0	20021150
64	CONTINUE	20021160
	IF(NREJ.LT.100) GOTO 30	20021170
	CALL ERROR(40HPRIMAL--TOO MANY REJECT VECTORS)	20021180
C	-----TRY ENDING IF PHASE 2	20021190
	IF(IPHASE.EQ.2) GOTO 2000	20021200
	CALL ESCAPE(B)	20021210
C		20021220
65	IF(ABS(THETA*DJ(JCOL)).GE.CTOL) GOTO 70	20021230
	IF(NDJS.EQ.1) GOTO 70	20021240
	IF(NDEG.GE.NDEGLM) GOTO 70	20021250
	NDEG=1+NDEG	20021260
	JAREJ(JCOL)=1	20021270
	GOTO 30	20021280
C		20021290
C	-----CHECK EXCEED BOUND ON JPOS----- XJPOS MOVES TO OR OFF BOUND	20021300
70	CONTINUE	20021310
	CALL XCHECK(6HPRIMAL, 3HEND, B)	20021320
	IF(ABS(THETA)+ZERO.LT.BNDJ) GOTO 80	20021330
	ITYPE=1	20021340
C	-----SUPPRESS PRICING NEXT TIME	20021350
	NREJ=1	20021360
C	-----JOUT=0 KILLS STATUS PRINT	20021370
	JOUT=0	20021380
	IF(THETA.GE.0.0) GOTO 75	20021390
C	-----XJPOS COMES OFF BOUND, THETA NEG.	20021400
	CALL SETBND(-JPOS)	20021410
	THETA= -BNDJ	20021420
	GOTO 90	20021430
C	-----XJPOS GOES TO BOUND	20021440
75	CALL SETBND(JPOS)	20021450
	THETA = BNDJ	20021460
	GOTO 90	20021470
C		20021480
C	-----PICK UP REJECTED COL, C FOR KEY CHANGE	20021490
80	CONTINUE	20021500

JOUT=IRASIS(IPOW)/100	20021510
IF(IPOW.LE.M) GOTO 800	20021520
C-----KEY CHANGE, CORRECT REJECTED COL AND CHECK ESSENTIAL PACKET	20021530
JOUT=KEYS(IROW-M)/100	20021540
NBVPKT=MOD(KEYS(IPOW-M),100)	20021550
IF(NBVPKT.GT.0) GOTO 81	20021560
C-----CHANGE KEY FROM JOUT TO JPOS IN NON-ESSENTIAL PKT	20021570
KEYS(IPOW-M) = 100*JPOS	20021580
CALL SET KEY(JPOS)	20021590
CALL SFT KEY(-JOUT)	20021600
C-----SET PARAMS FOR KEY STEP	20021610
EPST=THETA	20021620
C-----SUPPRESS PRICING NEXT TIME	20021630
NPEJ=1	20021640
GOTO 90	20021650
C	20021660
C-----ESSENTIAL PACKET, CHANGE JOUT FROM KEY TO BASIC IN NEWROW	20021670
81 CALL KEY CH(JOUT,NEWROW,R)	20021680
IROW = NEWROW	20021690
C	20021700
C-----NORMAL PIVOT OPERATION	20021710
800 CALL PIVOT(IROW,R, ALPHA)	20021720
C-----UPDATE KEY BASIS COUNTS FOR JPOS AND JOUT	20021730
JPOSPK=NPKT(JPOS)	20021740
IF(JPOSPK.EQ.0) GOTO 82	20021750
KEYS(JPOSPK)= KEYS(JPOSPK)+1	20021760
82 JOUTPK=NPKT(JOUT)	20021770
IF(JOUT.GT.NT) GOTO 84	20021780
IF(JOUTPK.EQ.0) GOTO 84	20021790
KEYS(JOUTPK)= KEYS(JOUTPK)-1	20021800
84 CONTINUE	20021810
C	20021820
C-----CHECK JPOS COMING OFF A ROUND (THETA.LE.0)	20021830
EPST=THETA	20021840
IF(ATRND(JPOS)) EPST=BNDJ+THETA	20021850
C	20021860
C-----CHECK JOUT, MARK NEW AND UNMARK OLD BASIC COLS	20021870
JOUT = IRASIS(IPOW)/100	20021880
IRASIS(IPOW) = 100*JPOS+MOD(IRASIS(IPOW),100)	20021890
CALL SETRNB(JPOS)	20021900
CALL SETRNB(-JOUT)	20021910
C-----ITYPE=2 IMPLIES JOUT OFF ROUND, =3 IMPLIES JOUT TO ROUND	20021920
IF(ITYPE.EQ.3) CALL SETBND(JOUT)	20021930
C-----RELEASE REJECTED VECTORS AFTER A PIVOT	20021940
IF(NPEJ+NDFG.EQ.0) GOTO 90	20021950
NREJ=NDFG=0	20021960
DO 85 I=1,JNDRF	20021970
C	20021980
C	20022000
C-----STEP BETA AND CONDITION COMPLETE PROBLEM (GUR ROWS ARE LAST)	20022010
85 JAPEJ(I)=0	20021980
90 DO 100 I=1,MPL	20022020
BETA(I)=BETA(I)-THETA*ALPHA(I)	20022030
100 CONTINUE	20022040
IF(ITYPE.NE.1) BETA(IROW)= EPST	20022050
DO 110 I=1,M	20022060
IF(ITP(I).EQ.3) GOTO 110	20022070
IF(BETA(I).GE.0.) GOTO 110	20022080

	BETA(I)=0.0	20022090
110	CONTINUE	20022100
	DO 120 I=1,L	20022110
	IF(BETA(I+M).GE.0.0) GOTO 120	20022120
	BETA(I+M)=0.0	20022130
120	CONTINUE	20022140
	CALL STATUS(40HEND OF PRIMAL)	20022150
	GOTO 1	20022160
C	*****END OF BASIC PRIMAL CYCLE*****	20022170
C		20022180
C	-----OPTIMUM PHASE1 TERMINATION	20022190
	1000 CONTINUE	20022200
	CALL OPT1	20022210
	IPHASE=2	20022220
	BETA(M)=0.0	20022230
	GOTO 3000	20022240
C		20022250
C	-----OPTIMUM PHASE2 TERMINATION	20022260
	2000 CONTINUE	20022270
	CALL OPT2	20022280
C		20022290
	RETURN	20022300
	END	20022310

SUBROUTINE POW(THETA,IPRW,JCOL,ITYPE,R)	20022320
C-----FOR VERSION APRIL 20/71	20022330
C-----FINDS STEP TO BOUND AND BOUND ENCOUNTERED	20022340
COMMON /I/ M,L,MOL,MC,NT,ICOST,IC,IPHASE,JPHS,IPJ	20022350
COMMON /CORE/ JAPFJ(101),JA(111),JAK(111),AJ(100)	20022360
COMMON /BASIS/ IBASIS(101),KEYS(101)	20022370
COMMON /NAMES/ NAME(100)	20022380
COMMON /A/ ALPHA(101) /B/ BETA(101) /G/ GAMMA(101) /D/ DELTA(101)	20022390
COMMON /TOLS/ DJTOL,TFRO,PIVTOL,CTOL,PERTOL,DEPTOL	20022400
LOGICAL BASIC,ATBND	20022410
REAL B(1)	20022420
ITP(J)=MOD(IBASIS(J),100)	20022430
NPKT(J)=MOD(NAME(J),100000)/10	20022440
ATBND(I)=MOD(NAME(I),10).EQ.7	20022450
C	20022460
C-----TRANSFORM SELECTED COLUMN	20022470
JPOS=JA(JCOL)	20022480
C-----LOAD COLUMN TO DELTA (MAYBE NULL PACKET)	20022490
JORG=M*JCOL-M	20022500
DO 5 I=1,M	20022510
DELTA(I)=AJ(JORG+I)	20022520
C-----NULL ELEMENTS IN LOWER ALPHA FOR PACKET ROWS	20022530
DO 6 I=1,L	20022540
ALPHA(I+M)=0.	20022550
C-----PACKET ROW HAS A UNITY	20022560
JPKT=NPKT(JPOS)	20022570
IF (JPKT.EQ.0) GOTO 16	20022580
ALPHA(JPKT+M)=1.	20022590
C-----FIND KEY AND KORG	20022600
KORG=KEYFND(JPKT)	20022610
IF(KORG.NE.0) GOTO 14	20022620
CALL ERROR(4,IPRW--KEY NOT IN CORE	20022630
WRITE(6,998)JPKT,JCOL,JPOS	20022640
998 FORMAT(1H+,40X,*PACKET*15* POSITION*15* COLUMN*15)	20022650
CALL ESCAPE(P)	20022660
IPRW=0	20022670
RETURN	20022680
14 CONTINUE	20022690
KORG=M*KORG-M	20022700
DO 15 I=1,M	20022710
DELTA(I)=DELTA(I)-AJ(KORG+I)	20022720
C-----TRANSFORM REDUCED COLUMN IN LOWER ALPHA	20022730
16 IORG=1	20022740
DO 20 I=1,M	20022750
ALPHA(I) = DOT(M, P(IORG), DELTA)	20022760
20 IORG=IORG+M	20022770
C-----SUM PACKET BASIC ENTRIES TO ALPHA ELEMENTS	20022780
IF(L.EQ.0) GOTO 26	20022790
DO 25 J=1,M	20022800
IB = IBASIS(J)/100	20022810
IF(IA.GT.NI) GOTO 25	20022820
K= NPKT(IB)	20022830
IF(K.EQ.0) GOTO 25	20022840
K=K+M	20022850
ALPHA(K)=ALPHA(K)-ALPHA(I)	20022860
25 CONTINUE	20022870
26 CONTINUE	20022880

C	THETA=1.E35	20022890
	IR0W = 0	20022900
	ITYPE= 1	20022910
C		20022920
	IF(ATBND(JPOS)) GOTO 100	20022930
C		20022940
C-----	X(JPOS) ZERO, DJ NEGATIVE ---INCREASE X(JPOS)	20022950
	DO 50 I=1,MPL	20022960
	IF(I.LE.M .AND. ITP(I).EQ.3) GOTO 50	20022970
	IF (ALPHA(I).LT.-ZERO) GOTO 30	20022980
	IF (ALPHA(I).GT. ZERO) GOTO 10	20022990
	GOTO 50	20023000
C-----	POSITIVE PIVOT	20023010
10	STEP = BETA(I)/ALPHA(I)	20023020
	IF(STEP .GE. THETA) GOTO 50	20023030
	THETA = STEP	20023040
	IR0W = I	20023050
	ITYPE = 2	20023060
	GOTO 50	20023070
C-----	NEGATIVE PIVOT----- (ROUND(JOUT).GE.BETA(I))	20023080
30	JOUT = IBASIS(I)/100	20023090
	IF(I.GT.M) JOUT=KEYS(I-M)/100	20023100
	STEP = (BETA(I) - BOUND(JOUT)) / ALPHA(I)	20023110
	IF(STEP .GE. THETA) GOTO 50	20023120
	THETA = STEP	20023130
	IR0W = I	20023140
	ITYPE = 3	20023150
50	CONTINUE	20023160
	GOTO 200	20023170
C		20023180
C-----	X(JPOS) AT BOUND DJ POS.--DECREASE X(JPOS)	20023190
100	DO 150 I=1,MPL	20023200
	IF(I.LE.M .AND. ITP(I).EQ.3) GOTO 150	20023210
	IF(ALPHA(I).LT. - ZERO) GOTO 130	20023220
	IF(ALPHA(I).GT. ZERO) GOTO 110	20023230
	GOTO 150	20023240
C-----	POSITIVE PIVOT----- (ROUND(JOUT).GE. BETA(I))	20023250
110	JOUT = IBASIS(I)/100	20023260
	IF(I.GT.M) JOUT=KEYS(I-M)/100	20023270
	STEP= (BETA(I)-BOUND(JOUT)) / (-ALPHA(I))	20023280
	IF(STEP .GE. THETA) GOTO 150	20023290
	THETA = STEP	20023300
	IR0W = I	20023310
	ITYPE = 3	20023320
	GOTO 150	20023330
C-----	NEGATIVE PIVOT	20023340
130	STEP= BETA(I)/ (-ALPHA(I))	20023350
	IF(STEP .GE. THETA) GOTO 150	20023360
	THETA= STEP	20023370
	IR0W = I	20023380
	ITYPE= 2	20023390
150	CONTINUE	20023400
	THETA= -THETA	20023410
C-----	PIVOTS ON 2,3, 2 DRIVES JOUT TO ZERO , 3 MOVES JOUT TO BOUND	20023420
C		20023430
200	RETURN	20023440
	END	20023450
		20023460

	SUBROUTINE SETBND(T)	20023470
	COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IP	20023480
	COMMON /NAMES/ NAME(100)	20023490
	K=3	20023500
100	CONTINUE	20023510
	IF(I) 1,2,3	20023520
1	J=-T	20023530
	IF(J.LE.NT) NAME(J)=10*(NAME(J)/10)+1	20023540
2	RETURN	20023550
3	IF(T.LE.NT) NAME(T)=10*(NAME(T)/10)+K	20023560
	RETURN	20023570
0		20023580
	ENTRY SETBNA	20023590
	K=2	20023600
	GOTO 100	20023610
0		20023620
	ENTRY SETBNV	20023630
	K=0	20023640
	GOTO 100	20023650
0		20023660
	ENTRY SETKEY	20023670
	K=4	20023680
	GOTO 100	20023690
	END	20023700

SUBROUTINE SETUP	20023710
C-----GUB VERSION APRIL/71	20023720
INTFGER PKT,PKT1	20023730
COMMON /INPUT/INPUT,INPUTM,INPUTN	20023740
COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NORMAX,NSCAN	20023750
COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI	20023760
COMMON /A/ ALPHA(101) /R/ BETA(101) /C/ GAMMA(101) /Q/ DELTA(101)	20023770
COMMON /POWTP/ IPOWTP(101) /NAMES/ NAME(100)	20023780
COMMON /BASIS/ IBASIS(101),KEYS(101)	20023790
COMMON /COPE/ JARFJ(101),JA(101),JAK(101),AJ(1000)	20023800
COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,NWAJ	20023810
COMMON /RHS/ RHS(100)	20023820
COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DEPTOL	20023830
COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS	20023840
COMMON /BOUNDS/ ROUNDS(100),IRDS(100),NRDS	20023850
COMMON /CV2 / T(100,10),B0(100),BL0(10),UBS(10),CO(10)	20023860
COMMON /CV3/ MRBCAV,NRBCAV,NCHGS	20023870
C-----TAKES INPUT A MATRIX IN COLUMNS OFF INPUT FILE BY COLUMNS	20023880
C-----INITIAL SETUP FOR COMPLETE PROBLEM IS CHANGED AT END TO REDUCED PR	20023890
C-----OUT DROPS GUB ROWS AS FOUND	20023900
CALL MESSG(40HSETUP	20023910
C-----ACTUAL PROBLEM SIZES	20023920
M= INPUTM+1	20023930
C-----TRY TO START IN PHASE 2	20023940
IPHASE=2	20023950
IC=ICOST	20023960
C-----TOLERANCES	20023970
DJTOL=1.E-8	20023980
ZERO=1.E-12	20023990
PIVTOL=1.E-5	20024000
CTOL=1.E-4	20024010
PERTOL=1.E-5	20024020
DEPTOL=1.E-8	20024030
NSCAN=NTRY=MAXTRY=NDJS=ITRN=JNCOPE=0	20024040
C-----INVERT FREQUENCY INVF, ITERATION OF NEXT INVERT ITNINV	20024050
INVF=100	20024060
ITNINV=0	20024070
C	20024080
C	20024090
C-----FIRST M COLUMNS GIVE ROW LOGICALS AND TYPES	20024100
DO 5 I=1,M	20024110
KEYS(I)=0	20024120
5 IBASIS(I)=0	20024130
C-----MARK PHASE1 COST ROW FREE FOR POSSIBLE USE IN PHASE 1	20024140
IROWTP(M)=3	20024150
C-----SET UP LOGICAL VECTORS FOR THE ROWS	20024160
DO 10 I=1,M	20024170
10 ALPHA(I)=0.0	20024180
C-----NOW COUNT COLS WRITTEN	20024190
NT=0	20024200
DO 100 I=1,M	20024210
ID=IROWTP(I)	20024220
IF(ID.EQ.0) GOTO 20	20024230
IF(ID.EQ.1) GOTO 21	20024240
IF(ID.EQ.2) GOTO 22	20024250
IF(ID.EQ.3) GOTO 23	20024260
IF(ID.EQ.4) GOTO 24	20024270

19	CALL EPPOR(40H SETUP--ROW TYPE EPPOR--OUT OF RANGE)	20024280
	CALL ESCAPE		20024290
C----	EQUALITY ROW- NO LOGICAL COL.		20024300
20	GOTO 100		20024310
C----	LE. ROW- POSITIVE LOGICAL+SLACK		20024320
21	ALPHA(I)=1.0		20024330
	K=1		20024340
	GOTO 50		20024350
C----	GE. ROW- NEGATIVE LOGICAL+SLACK		20024360
22	ALPHA(I)=-1		20024370
	K=1		20024380
	GOTO 50		20024390
C----	FREE ROW-POSITIVE LOGICAL-BASIC		20024400
23	ALPHA(I)=+1.		20024410
	IRASIS(I)=NT+1		20024420
	K=2		20024430
	GOTO 50		20024440
C----	GUM POW-NO LOGICAL		20024450
24	CONTINUE		20024460
	GOTO 100		20024470
C----	PLACE COLUMN IN FILE IA1 AND IA2		20024480
50	NT=NT+1		20024490
	NAME(NT)=C		20024500
	CALL OUT(NT,ALPHA,COLNM)		20024510
	ALPHA(T)=0		20024520
100	CONTINUE		20024530
C----	KEEP PHASE 1 COST POW ZERO IN PHASE 2		20024540
	IROWTP(M)=0		20024550
C----	NO. OF LOGICAL COLS MC		20024560
	MC=NT		20024570
C			20024580
C----	CYCLE INPUT FILE COLUMNS		20024590
	REWIND INPUT		20024600
	DO 200 JNT=1,INPUTN		20024610
110	IF(JNT.NE.JPHS) GOTO 130		20024620
	READ (INPUT) (PHS(J),J=1,INPUTM)		20024630
C----	TAKE RHS FROM 80 AND SKIP TAPE VERTON		20024640
	DO 120 J=1,INPUTM		20024650
120	PHS(J)=80(J)		20024660
	PHS(M)=0.		20024670
	NT=NT+1		20024680
	NAME(NT)=0		20024690
	CALL OUT(NT,PHS,COLNM)		20024700
	GOTO 200		20024710
C----	GET NEXT COLUMN JNT		20024720
130	CONTINUE		20024730
	READ(INPUT)(ALPHA(J),J=1,INPUTM)		20024740
C----	INSERT COLUMN CHANGES TO PROBLEM		20024750
	IF(JNT.GT.NCHGS) GOTO 135		20024760
	ALPHA(ICOST)=CO(JNT)		20024770
135	CONTINUE		20024780
C----	CHECK FOR COL PACKET, GET PKT NO. OR 0		20024790
	PKT=0		20024800
	PKT1=0		20024810
	DO 140 I=1,INPUTM		20024820
	IF(IPROWTP(I).NE.4) GOTO 140		20024830
	PKT1=1+PKT1		20024840
	IF(ALPHA (I).NE.1.) GOTO 140		20024850

	PKT =PKT1	20024860
	GOTO 145	20024870
140	CONTINUE	20024880
C-----	CHECK FOR BOUND, GET BOUND NO. OR 0	20024890
145	IF(NBDS.EQ.0) GOTO 151	20024900
	DO 150 J=1,NBDS	20024910
	IF (IBDS(J).EQ.JNT) GOTO 155	20024920
150	CONTINUE	20024930
151	J=0	20024940
155	CONTINUE	20024950
C-----	SET NAME TO BOUND+PACKET + STATE AND MARK KEY COLUMN	20024960
	K=1	20024970
C-----	COUNT COLUMN AND WRITE TO FILE LESS GUB ELEMENTS	20024980
160	NT=1+NT	20024990
	NAME(NT)=K+10*PKT+100000*J	20025000
	CALL OUT(NT, ALPHA,COLNM)	20025010
200	CONTINUE	20025020
C-----	REMOVE GUB ROWS FROM IBASIS AND RHS	20025030
	INON=0	20025040
	L=0	20025050
	IKOST=ICOST	20025060
	DO 220 I=1,M	20025070
	IF(IROWTP(I).EQ.4) GOTO 210	20025080
C-----	NON-GUB ROW	20025090
	INON=INON+1	20025100
	IBASIS(INON)=100*IBASIS(I)+IROWTP(I)	20025110
	RHS(INON)=RHS(I)	20025120
	GOTO 220	20025130
C-----	GUB POW, STOPE RHS IN AJ	20025140
210	L=L+1	20025150
	AJ(L)=RHS(I)	20025160
C-----	MOVE DOWN USER COST POW	20025170
	IF(I.LT.ICOST) IKOST=IKOST-1	20025180
220	CONTINUE	20025190
C-----	NOW REPLACE RHS ON END OF RHS	20025200
	IF(L.EQ.0) GOTO 240	20025210
	DO 230 I=1,L	20025220
230	RHS(INON+I)=AJ(I)	20025230
C-----	NOW DROP COUNT OF GUB ROWS	20025240
	M=M-L	20025250
	ICOST=IKOST	20025260
C-----	REDUCED PROBLEM NOW COMPLETE	20025270
240	CONTINUE	20025280
	RETURN	20025290
	END	20025300


```

SUBROUTINE STATUS(NOTE)                                20025310
DIMENSION NOTE(4)                                     20025320
COMMON /LIMS/ MAXTRY, NTRY, JNCOPE, NCMAX, NSCAN      20025330
COMMON /I/  M, L, MPL, MC, NT, ICOST, IC, TPHASE, JPHS, TPI 20025340
COMMON /NAMES/ NAME(100)                             20025350
COMMON /CORE/ JAPFJ(101), JA(101), JAK(101), AJ(100)   20025360
COMMON /A/  ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101) 20025370
COMMON /BASIS/ IBASIS(101), KEYS(101)                20025380
COMMON /DJS/ DJ(101)                                  20025390
COMMON /STATE/ JPOS, IPOW, JCOL, JOUT, ITPN, NPEJ, NPIF, NDJS 20025400
COMMON /PAPAMS/ TMAX, ITNINV, INVF, K1, K2, K3, K4, K5  20025410
DATA JNTO/0/                                          20025420
LOGICAL BASIC, ATEND                                  20025430
IF(MOD(K3,11).NE.0) RETURN                            20025440
IF( MOD(ITPN ,5()).EQ.0) WRITE(6,1)                  20025450
1  FORMAT(1H1                                          20025460
+      ,10H      PHASE                                20025470
+      ,10H      TTR                                     20025480
+      ,10H      TRY                                     20025490
+      ,15H      VAL OBJECTIVE                         20025500
+      ,10H      NDJS                                  20025510
+      ,10H      NPTS                                  20025520
+      ,15H      VALUE DJ IN                           20025530
+      ,10H      COL IN                               20025540
+      ,10H      CODE                                  20025550
+      ,10H      COL OUT                              20025560
+      ,10H      CODE                                  20025570
+      ,10H      NSCAN                                20025580
C-----STOP CURRENT SOLUTION, QUIT OR CONTINUE      20025590
CALL SECOND(X)                                        20025600
IF(X.LT.TMAX.AND.ITPN.LT.K5) GOTO 999                20025610
CALL MAPOUT(R)                                       20025620
CALL EXIT                                           20025630
999  CONTINUE                                         20025640
C-----COUNT ACTIVE ARTIFICIALS FOR STATUS DATA  20025650
NPIF=0                                               20025660
DO 15 I=1,M                                          20025670
15  IF(IBASIS(I)/100.GT.NT) NPIF=1+NPIF              20025680
     COST=-BETA(IC)                                  20025690
     NCOLS=JNCOPE                                    20025700
     IF(NTRY.NE.0) GOTO 20                            20025710
     CALL INPOS(JNT)                                  20025720
     NCOLS=JNT-JNTO                                   20025730
     IF(NCOLS.LE.0) NCOLS=NCOLS+NT                   20025740
20  JNTO=JNT                                          20025750
     UNSCAN=10000*NSCAN+NCOLS                         20025760
     MTRY=1000*MAXTRY+NTRY                             20025770
     IF(JCOL.EQ.0) GOTO 10                             20025780
     IF(IPOW.EQ.0) GOTO 10                             20025790
     NJOUT=NAME(JOUT)                                  20025800
     IF(JOUT.EQ.0) NJOUT=                             20025810
     IF(JOUT.GT.NT) NJOUT=10000JOUT*IPOW            20025820
     WRITE(6,2) IPHASE, ITPN, MTRY, COST, NDJS, NPIF, DJ(JCOL) 20025830
+     , JPOS, NAME(JPOS), JOUT, NJOUT, JNSCAN        20025840
2  FORMAT(1H ,I10, F15.6, 2I10, F15.6, 5I10)         20025850
     RETURN                                           20025860
C-----WHEN NO COLUMN WAS SELECTED                  20025870

```

C-----	OR UNBOUNDED	20025880
10	WRITE(6,3) IPHASE,ITRN,MNTRY,COST,NDJS,NPIF,NOTE,JNSCAN	20025890
3	FORMAT(1H,3I10,F15.6,2I10,15H-----,4A10,I10)	20025900
	RETURN	20025910
C		20025920
C		20025930
	ENTRY ERROR	20025940
	WRITE(6,4) NOTE,NOTE,NOTE	20025950
4	FORMAT(1H/(1H+,4A10))	20025960
	RETURN	20025970
C		20025980
C		20025990
	ENTRY MESSG	20026000
	ENTRY MSSG	20026010
	IF(MOD(K3,7).NE.0) RETURN	20026020
	CALL SECOND(X)	20026030
	WRITE(6,5) NOTE,X	20026040
5	FORMAT((1H,4A10,6CX,F10.0,* SECONDS*)	20026050
	RETURN	20026060
	END	20026070

```

SUBROUTINE XCHECK(CALLER,AT,N)
PEAL R(1)
C-----GIVES QUICK CROSS CHECKS AND LOCATION
COMMON /PHS/ PHS(100)
COMMON /MOVES/ THETA,RNDJ,DMAX,PRMLER,DUALER
COMMON /PARAMS/ TMAX,ITNTNV,INVF,K1,K2,K3,K4,K5
COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
COMMON /BASIS/ IBASIS(101),KEYS(101)
COMMON /CORE/ JAPFJ(101),JA(101),JAK(1..1),AJ(1000)
COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IPT
COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NORMAX,NSCAN
COMMON /DJS/ DJ(100)
COMMON /NAMES/ NAME(100)
COMMON /TOLS/ DJTOL,ZERO,PIVTOL,GTOL,PERTOL,DEPTOL
COMMON /STATE/ JPOS,TROW,JCOL,JOUT,ITRN,NPEJ,NPIF,NDJS
COMMON /ROUNDS/ ROUNDS(100),IBDS(100),NRDS
INTEGER DELTA
IPRWT(I)=MOD(IBASIS(I),100)
C-----K4 IS 1000*START + STOP ITERATION FOR XCHECKS
IF(K4/1000.GT.ITRN.OR.MOD(K4,1000).LT.ITRN) RETURN
K4=1000*(ITRN+K2)+MOD(K4,1000)
WRITE(6,1) CALLER,TROW,THETA,RNDJ,JPOS
1  FORMAT(*0.....*/
+ * XCHECK CALLED BY *A6* PIVOT ROW---*I3,
+ * STEP*E12.5* ROUND*E11.4* COLUMN*I5)
6  WRITE(6,5) (J(J),J=1,JNCORE)
FORMAT(32X,10F10.4)
JKOL=JCOL
J1=MAX0(JKOL-5,1)
J2=MIN0(J1+9,JNCORE)
WRITE(6,5) (JA(K),K=J1,J2)
5  FORMAT(32X,10I10)
IORG=1
JEND=JNCORE*M
DO 40 I=1,M
JORG=J1*M-M+1
DO 30 J=J1,J2
AJ(JEND+J)=DOT(M,0(IORG),AJ(JORG))
IF(ABS(AJ(JEND+J)).LE.ZERO) AJ(JEND+J)=0.
30  JORG=JORG+M
WRITE(6,8) I,IBASIS(I),ALPHA(I),BETA(I),(AJ(JEND+J),J=J1,J2)
40  IORG=IORG+M
DO 45 J=J1,J2
JAJ=JA(J)
45  DELTA(J)=NAME(JAJ)
WRITE(6,5) (DELTA(J),J=J1,J2)
IF(L.EQ.J) GOTO 50
DO 50 I=1,L
WRITE(6,8) I,KEYS(I),ALPHA(I+M),BETA(I+M)
50  CONTINUE
60  CONTINUE
8  FORMAT(I3,I7,2E10.2,2X,10E10.2)
C-----ERROR CHECKING OPTION IN XCHECK
7) CONTINUE
IF(MOD(K3,17).NE.1) GOTO 555
C-----GAMMA SUMS LHS-PHS
DO 90 I=1,MPL

```

90	GAMMA(I)=-RHS(T)	20026650
C-----	CYCLE ALL BOOK KFFPING TO GET COLUMNS IN ORDER	20026650
	LAST=1	20026670
	NVARS=MPL+NBDS	20026680
	DO 500 NVAR=1,NVAPS	20026690
	NEXT=999999999	20026700
C-----	BASIC COLUMNS	20026710
	DO 200 I=1,M	20026720
	J=IBASIS(I)/100	20026730
C-----	MARK STRUCTURALS	20026740
	ITAG=2	20026750
	IF(J.LE.NT) GOTO 150	20026760
C-----	MARK ARTIFICIALS	20026770
	ITAG=5	20026780
	IF(J.LE.NT+100) GOTO 150	20026790
C-----	MARK NEGATIVE STRUCTURALS	20026800
	ITAG=6	20026810
	J=J-(NT+100)	20026820
	IF(J.LE.NT) GOTO 150	20026830
C-----	MARK NEGATIVE ARTIFICIALS	20026840
	ITAG=7	20026850
150	CONTINUE	20026860
	IF(J.LT.LAST) GOTO 200	20026870
	IF(J.GT.NEXT) GOTO 200	20026880
	NEXT=J	20026890
	JTYPE=2	20026900
	JTAG=ITAG	20026910
	X=BETA(I)	20026920
	IBAS=I	20026930
200	CONTINUE	20026940
C-----	BOUNDED COLUMNS	20026950
	IF(NBDS.EQ.0) GOTO 300	20026960
	NEXTJ=MIND(NT,NEXT)	20026970
	DO 250 J=LAST,NEXTJ	20026980
	IF(MOD(NAME(J),10).EQ.3) GOTO 290	20026990
250	CONTINUE	20027000
	GOTO 300	20027010
290	NEXT=J	20027020
	JTYPE=3	20027030
	X=ROUND(J)	20027040
C-----	KEY COLUMNS	20027050
300	CONTINUE	20027060
	IF(L.EQ.0) GOTO 360	20027070
	DO 350 I=1,L	20027080
	J=KEYS(I)/100	20027090
	IF(J.LT.LAST) GOTO 350	20027100
	IF(J.GT.NEXT) GOTO 350	20027110
	NEXT=J	20027120
	JTYPE=4	20027130
	X=BETA(M+I)	20027140
350	CONTINUE	20027150
360	CONTINUE	20027160
C		20027170
C-----	GET NEXT COLUMN TO CORE IF REAL (JTAG=2 OR 6)	20027180
	IF(NEXT.EQ.999999999) GOTO 510	20027190
	IF(NEXT.GT.NT) GOTO 400	20027200
	CALL IN(NEXT,AJ(JFND+1),JNCOPE+1)	20027210
C-----	ADD GUB ELEMENTS	20027220

	MP1=M+1	20027230
	IF(L.EQ.0) GOTO 385	20027240
	DO 390 I=MP1,MPL	20027250
390	AJ(JFND+T)=0.	20027260
	IGUR=400(NAME(NEXT),100000)/10	20027270
	IF(IGUR.NE.0) AJ(JFND+M+IGUR)=1.	20027280
385	IF(JTYPE.NE.2) GOTO 450	20027290
	IF(JTAG.NE.5) GOTO 450	20027300
C-----	NEGATIVE STRUCTURALS	20027310
	DO 390 I=1,MPL	20027320
390	AJ(JFND+T)=-AJ(JFND+T)	20027330
	AJ(JFND+M)=1.	20027340
	GOTO 450	20027350
C-----	ARTIFICIAL VECTOR (JTAG=5 OR 7)	20027360
400	DO 410 I=1,MPL	20027370
410	AJ(JFND+T)=0.	20027380
	AJ(JFND+IRAS)=1.	20027390
	AJ(JFND+M)=1.	20027400
	IF(JTAG.EQ.7) AJ(JFND+IRAS)=-1.	20027410
C		20027420
C-----	SUM X*AJ TO GAMMA-- THE ERROR	20027430
450	DO 460 I=1,MPL	20027440
460	GAMMA(I)=GAMMA(I)+X*AJ(JFND+T)	20027450
500	LAST=NEXT+1	20027460
501	FORMAT(4I4,F12.4,10F10.4/(28X,10F10.4))	20027470
510	CONTINUE	20027480
C		20027490
C-----	CHECK ERROR AGAINST TOLERANCE	20027500
	PRMLER=0.	20027510
	K=0	20027520
	DO 550 I=1,MPL	20027530
	ARSGAM=ABS(GAMMA(I))	20027540
	IF(ARSGAM.LE.PERTOL) GOTO 550	20027550
	K=K+1	20027560
	DELTA(K)=I	20027570
	GAMMA(K)=GAMMA(I)	20027580
	IF(ARSGAM.LE.PRMLER) GOTO 550	20027590
	PRMLER=ARSGAM	20027600
550	CONTINUE	20027610
	IF(PRMLER.LE.PERTOL) WRITE(6,552)	20027620
	IF(PRMLER.LE.PERTOL) GOTO 555	20027630
	WRITE(6,551) PRMLER,PERTOL,(DELTA(I),GAMMA(I),I=1,K)	20027640
	IF(MOD(K,19).NE.0) GOTO 555	20027650
	ITNINV=ITRN	20027660
551	FORMAT(*PRIMAL ERRORS EXCEED TOLERANCE---*/	20027670
	+ * ERROR--*F12.4* TOLERANCE--*F12.4/(4(I10,E20.8)))	20027680
552	FORMAT(* ERRORS WITHIN TOLERANCE*)	20027690
555	WRITE(6,7)	20027700
7	FORMAT(1H ,40H----- //)	20027710
	RETURN	20027720
	END	20027730

	PROGRAM REGEN(INPUT,OUTPUT,TAPEA,TAPES=INPUT,	30000010
1	TAPE6=OUTPUT,TAPE9=TAPEA)	30000020
	COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),	30000030
*	VCOST(10,5),NAMEN(10),COSTS(30,3)	30000040
	COMMON /BASICS/ CHAP(5000,4),CODE(20),PER(10),IYR(10),LYR(10)	30000050
	COMMON /OUTS/ QANDM(20),SALE(20),SAVE(20),EXIST(10,20),	30000060
*	PURCH(10,20),STOR(10,20),SALV(10,20),PROC(20),PROT(20)	30000070
	COMMON /PARAMS/ PDTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST	30000080
	DATA CODE / 2H01,2H02,2H03,2H04,2H05,2H06,2H07,2H08,2H09,2H10,	30000090
*	2H11,2H12,2H13,2H14,2H15,2H16,2H17,2H18,2H19,2H20/	30000100
	CALL SETUP	30000110
50	READ(9,100) (TITLE(I),I=1,4)	30000120
100	FORMAT(4A10)	30000130
	IF(EOF,9)7777,200	30000140
200	PDTOT=0.0	30000150
	DO 300 I=1,20	30000160
	SALE(I)=0.0	30000170
	SAVE(I)=0.0	30000180
	PROC(I)=PROT(I)	30000190
	QANDM(I)=0.0	30000200
	DO 250 N=1,10	30000210
	PURCH(N,I)=0.0	30000220
	EXIST(N,I)=0.0	30000230
	STOR(N,I)=0.0	30000240
	SALV(N,I)=0.0	30000250
250	CONTINUE	30000260
300	CONTINUE	30000270
	CALL INSOLN	30000280
	CALL CINFO	30000290
	CALL PINFO	30000300
	GO TO 50	30000310
7777	STOP	30000320
	END	30000330

	SUBROUTINE CINFO	30000340
	COMMON /VECTG/ VNAME(10),C,LENP,VLTFE(10),INH(10,16),	30000350
	* VCONST(10,5),NAMEN(10),COSTC(30,3)	30000360
	COMMON /BASICS/ CHAR(5000,4),CODE(20),PER(10),LYR(10),LYR(10)	30000370
	COMMON /OUTS/ OANDM(20),SALE(20),SAVE(20),EXIST(10,20),	30000380
	* PURCH(10,20),STOP(10,20),SALV(10,20),PPROC(20),PROT(20)	30000390
	COMMON /PARAMS/ PDTOT,INYSR,LAST,NV,NP,TOT,TITLE(4),COST	30000400
	DATA ONE,TOTAL,PERIOD / 2H01,6HTOTAL ,6HPERIOD /	30000410
	WRITE(6,100)(TITLE(I),I=1,4)	30000420
	SUMT=0.0	30000430
	TPROC=0.0	30000440
	TCOST=0.0	30000450
	DO 500 I=1,20	30000460
500	TCOST=TCOST+OANDM(I)+SAVE(I)-SALE(I)	30000470
	TCOST=COST-TCOST-PDTOT	30000480
	DO 600 I=1,NP	30000490
600	TPROC=TPROC+PROC(I)	30000500
	TPROC=TPROC/TCOST	30000510
	DO 1000 I=1,NP	30000520
	IF(PPROC(I).EQ.ONE) N=I	30000530
1000	CONTINUE	30000540
	DO 2000 I=N,NP	30000550
	K1=IYR(I)-INYSR+1	30000560
	K2=LYR(I)-INYSR+1	30000570
	DO 1500 K=K1,K2	30000580
	IF(K.EQ.K1) GO TO 1500	30000590
	OANDM(K1)=OANDM(K1)+OANDM(K)	30000600
	SALE(K1)=SALE(K1)+SALE(K)	30000610
1500	CONTINUE	30000620
	J=I-N+1	30000630
	PROC(I)=PROC(I)/TPROC	30000640
	OANDM(K1)=OANDM(K1)+SAVE(I)	30000650
	SUM=PROC(I)+OANDM(K1)-SALE(K1)	30000660
	WRITE(6,200) PERIOD,CODE(J),PROC(I),OANDM(K1),SALE(K1),SUM	30000670
	IF (J.EQ.1) GO TO 2000	30000680
	OANDM(1)=OANDM(1)+OANDM(K1)	30000690
	SALE(1)=SALE(1)+SALE(K1)	30000700
	PROC(N)=PROC(N)+PPROC(I)	30000710
2000	SUMT=SUMT+SUM	30000720
	I=LYR(NP)-INYSR+2	30000730
	WRITE(6,300) TOTAL,PDTOT,PROC(N),OANDM(1),SALE(1),SUMT	30000740
	WRITE(6,400) SALE(I)	30000750
100	FORMAT(1H1,15X,4A10 / 1H-,*COST INFORMATION* /	30000760
	1H-,12X,5(1H,12X) / 13X,1H*,* R AND D *, 1H*,	30000770
	** PROCUREMENT*,1H*,* OPERATING *,1H*,* SALVAGE *,1H*,	30000780
	** TOTAL * / 1H ,77(1H*) / 13X,5(1H*,12X))	30000790
200	FORMAT(1H ,A6,2X,A2,2X,1H*,12X,4(1H*,1X,F9.3,2X) / 13X,5(1H*,12X))	30000800
300	FORMAT(13X,5(1H*,12X) / 1H ,A6,6X,5(1H*,1X,F9.3,2X) / 1H ,77(1H*))	30000810
400	FORMAT(1H- / 1H-,*TRUNCATION VALUE FOR RESOURCES = *,F9.3)	30000820
	RETURN	30000830
	END	30000840

	SURROUTINE INSOLN	30000850
	COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),	30000860
	* VCOST(10,5),NAMFN(10),COSTS(30,3)	30000870
	COMMON /BASICS/ CHAR(5000,4),CODE(20),PER(10),IYR(10),LYR(10)	30000880
	COMMON /OUTS/ OANOM(20),SALE(20),SAVE(20),EXIST(10,20),	30000890
	* PURCH(10,20),STOP(10,20),SALV(10,20),PROC(20),PROT(20)	30000900
	COMMON /PARAMS/ PDTOT,INYR,LAST,NV,NP,TOJ,TITLE(4),COST	30000910
	INTEGER TOT	30000920
	TOT=NP+1	30000930
	DATA X/1HX/,W/1HW/,S/1HS/,BLANK/1H /,BLAN2/2H /	30000940
7777	READ(9,130)IND,VAL	30000950
	IF(IND.GE.0) GO TO 101	30000960
	COST=VAL	30000970
	GO TO 700	30000980
101	IF(IND.EQ.0) GO TO 7777	30000990
	IF(CHAR(IND,1).EQ.X) GO TO 500	30001000
	IF(CHAR(IND,1).EQ.W) GO TO 400	30001010
	IF(CHAR(IND,1).EQ.S) GO TO 300	30001020
	IF(CHAR(IND,3).NE.BLAN2) GO TO 7777	30001030
C	INTERPRET PNM VARIABLES FOR INVESTMENT CONSTRAINTS	30001040
	DO 200 I=1,20	30001050
	IF (PER(I).EQ.CHAP(IND,2)) GO TO 210	30001060
200	CONTINUE	30001070
	GO TO 1000	30001080
210	PROC(I)=PROC(I)-VAL	30001090
	PROC(I+1)=PROC(I+1)+VAL	30001100
	GO TO 7777	30001110
C	INTERPRET INHERITED FLEET AND PURCHASE FLEET VARIABLES	30001120
400	DO 405 J=1,20	30001130
	IF(CODE(J).EQ.CHAP(IND,2)) GO TO 410	30001140
405	CONTINUE	30001150
	GO TO 1000	30001160
410	DO 420 I=1,10	30001170
	IF(PER(I).EQ.CHAP(IND,3)) GO TO 430	30001180
420	CONTINUE	30001190
	GO TO 1000	30001200
430	ISTART=IYR(I)	30001210
	IF(CHAR(IND,1).EQ.X) PURCH(J,I)=VAL	30001220
	DO 450 I=1,10	30001230
	IF(PER(I).EQ.CHAP(IND,4)) GO TO 460	30001240
450	CONTINUE	30001250
	GO TO 1000	30001260
460	IEND=LYR(I)	30001270
	CALL VALUFS(J,ISTART,IEND,VAL)	30001280
	GO TO 7777	30001290
C	INTERPRET MOTHBALL VARIABLES	30001300
300	DO 350 I=1,20	30001310
	IF(CODE(I).EQ.CHAP(IND,2)) GO TO 360	30001320
350	CONTINUE	30001330
	GO TO 1000	30001340
360	DO 370 J=1,10	30001350
	IF(PER(J).EQ.CHAP(IND,3)) GO TO 380	30001360
370	CONTINUE	30001370
	GO TO 1000	30001380
380	STOP(I,J)=VAL	30001390
	LENP=LYR(J)-IYR(J)+1	30001400
	CALL MOTH(I)	30001410

SAVE(J)=SAVE(J)+C*VAL	30001420
GO TO 7777	30001430
C INTERPRET MASTER VARIABLES	30001440
500 IF(CHAP(IND,3).NE.PLAN2) GO TO 400	30001450
DO 550 I=1,20	30001460
IF(CODE(T).EQ.CHAP(IND,2)) GO TO 560	30001470
550 CONTINUE	30001480
GO TO 1000	30001490
560 PURCH(I,TOT)=VAL	30001500
IF(VAL.GT.C.C)	30001510
*PDTOT=PDTOT+VCOST(T,3)	30001520
GO TO 7777	30001530
C ERROR MESSAGE	30001540
1000 WRITE(6,600) (CHAP(IND,I),I=1,4)	30001550
STOP	30001560
700 RETURN	30001570
100 FORMAT(I4,4X,F12.4)	30001580
600 FORMAT(1H-,*ERROR IN INTERPRETATION OF *,A1,3A2)	30001590
END	30001600

SUBROUTINE PINFO	30001610
COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),	30001620
* VCOST(10,5),NAMFN(10),COSTS(30,3)	30001630
COMMON /BASICS/ CHAR(5000,4),CODE(20),PEP(10),IYR(10),LYR(10)	30001640
COMMON /OUTS/ OANDM(20),SALE(20),SAVE(20),EXIST(10,20),	30001650
* PURCH(10,20),STOP(10,20),SALV(10,20),PPOC(20),PROT(20)	30001660
COMMON /PARAMS/ PRTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST	30001670
INTEGER TOT	30001680
DATA TOTAL,PERIOD,BLANK / 6HTOTAL ,6HPERIOD,2H /	30001690
WRITE(6,1000) (TITLE(I),I=1,4)	30001700
M=1	30001710
5 GO TO (10,20,30,40,50,60,70,80,90),NV	30001720
10 WRITE(6,1010) (VNAME(I),I=1,NV)	30001730
GO TO 100	30001740
20 WRITE(6,1020) (VNAME(I),I=1,NV)	30001750
GO TO 100	30001760
30 WRITE(6,1030) (VNAME(I),I=1,NV)	30001770
GO TO 100	30001780
40 WRITE(6,1040) (VNAME(I),I=1,NV)	30001790
GO TO 100	30001800
50 WRITE(6,1050) (VNAME(I),I=1,NV)	30001810
GO TO 100	30001820
60 WRITE(6,1060) (VNAME(I),I=1,NV)	30001830
GO TO 100	30001840
70 WRITE(6,1070) (VNAME(I),I=1,NV)	30001850
GO TO 100	30001860
80 WRITE(6,1080) (VNAME(I),I=1,NV)	30001870
GO TO 100	30001880
90 WRITE(6,1090) (VNAME(I),I=1,NV)	30001890
1000 FORMAT(1H1,15X,4A10 / 1H-,*PURCHASED RESOURCES*)	30001900
1010 FORMAT(1H-,12X, 1H*,12X /13X, 1H*,2X,A9,2X /1H ,25(1H*)/	30001910
* 13X, 1H*,12X)	30001920
1020 FORMAT(1H-,12X,2(1H*,12X) /13X,2(1H*,2X,A8,2X)/1H ,38(1H*)/	30001930
* 13X,2(1H*,12X))	30001940
1030 FORMAT(1H-,12X,3(1H*,12X) /13X,3(1H*,2X,A8,2X)/14 ,51(1H*)/	30001950
* 13X,3(1H*,12X))	30001960
1040 FORMAT(1H-,12X,4(1H*,12X) /13X,4(1H*,2X,A8,2X)/1H ,64(1H*)/	30001970
* 13X,4(1H*,12X))	30001980
1050 FORMAT(1H-,12X,5(1H*,12X) /13X,5(1H*,2X,A8,2X)/1H ,77(1H*)/	30001990
* 13X,5(1H*,12X))	30002000
1060 FORMAT(1H-,12X,6(1H*,12X) /13X,6(1H*,2X,A8,2X)/1H ,90(1H*)/	30002010
* 13X,6(1H*,12X))	30002020
1070 FORMAT(1H-,12X,7(1H*,12X) /13X,7(1H*,2X,A8,2X)/1H ,103(1H*)/	30002030
* 13X,7(1H*,12X))	30002040
1080 FORMAT(1H-,12X,8(1H*,12X) /13X,8(1H*,2X,A8,2X)/1H ,116(1H*)/	30002050
* 13X,8(1H*,12X))	30002060
1090 FORMAT(1H-,12X,9(1H*,12X) /13X,9(1H*,2X,A8,2X)/1H ,129(1H*)/	30002070
* 13X,9(1H*,12X))	30002080
100 IF(M.GE.2) GO TO 305	30002090
K=0	30002100
DO 200 I=1,TOT	30002110
IF(PER(I).EQ.CODE(1)) K=1	30002120
IF(K.NE.1) GO TO 200	30002130
TEMP1=PERIOD	30002140
TEMP2=PER(I)	30002150
IF(I.NE.TOT) GO TO 105	30002160
TEMP1=TOTAL	30002170

TEMP2=BLANK	30002180
105 GO TO (110,120,130,140,150,160,170,180,190),NV	30002190
110 WRITE(6,1110) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002200
GO TO 200	30002210
120 WRITE(6,1120) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002220
GO TO 200	30002230
130 WRITE(6,1130) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002240
GO TO 200	30002250
140 WRITE(6,1140) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002260
GO TO 200	30002270
150 WRITE(6,1150) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002280
GO TO 200	30002290
160 WRITE(6,1160) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002300
GO TO 200	30002310
170 WRITE(6,1170) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002320
GO TO 200	30002330
180 WRITE(6,1180) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002340
GO TO 200	30002350
190 WRITE(6,1190) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)	30002360
200 CONTINUE	30002370
1110 FORMAT(1H ,A6,2X,A2,2X, 1H*,2X,F8.3,2X / 13X, 1H*,12X)	30002380
1120 FORMAT(1H ,A6,2X,A2,2X,2(1H*,2X,F8.3,2X) / 13X,2(1H*,12X))	30002390
1130 FORMAT(1H ,A6,2X,A2,2X,3(1H*,2X,F8.3,2X) / 13X,3(1H*,12X))	30002400
1140 FORMAT(1H ,A6,2X,A2,2X,4(1H*,2X,F8.3,2X) / 13X,4(1H*,12X))	30002410
1150 FORMAT(1H ,A6,2X,A2,2X,5(1H*,2X,F8.3,2X) / 13X,5(1H*,12X))	30002420
1160 FORMAT(1H ,A6,2X,A2,2X,6(1H*,2X,F8.3,2X) / 13X,6(1H*,12X))	30002430
1170 FORMAT(1H ,A6,2X,A2,2X,7(1H*,2X,F8.3,2X) / 13X,7(1H*,12X))	30002440
1180 FORMAT(1H ,A6,2X,A2,2X,8(1H*,2X,F8.3,2X) / 13X,8(1H*,12X))	30002450
1190 FORMAT(1H ,A6,2X,A2,2X,9(1H*,2X,F8.3,2X) / 13X,9(1H*,12X))	30002460
C	30002470
C FIRST PART OF THIS SUBROUTINE OUTPUT INFORMATION CONCERNING	30002480
C EQUIPMENT PURCHASES DURING EACH PERIOD	30002490
C NEXT SECTION OUTPUTS RESOURCES STORED	30002500
C	30002510
WRITE(6,3000) (TITLE(I),I=1,4)	30002520
M=3	30002530
GO TO 5	30002540
305 IF(M.EQ.2) GO TO 205	30002550
N=0	30002560
DO 400 I=1,NP	30002570
K=IYP(I)-INYP+1	30002580
IF(K.LE.0) GO TO 400	30002590
N=N+1	30002600
GO TO (310,320,330,340,350,360,370,380,390),NV	30002610
310 WRITE(6,1110) PERIOD, CODE(N), (STOP(J,I),J=1,NV)	30002620
GO TO 400	30002630
320 WRITE(6,1120) PERIOD, CODE(N), (STOP(J,I),J=1,NV)	30002640
GO TO 400	30002650
330 WRITE(6,1130) PERIOD, CODE(N), (STOP(J,I),J=1,NV)	30002660
GO TO 400	30002670
340 WRITE(6,1140) PERIOD, CODE(N), (STOP(J,I),J=1,NV)	30002680
GO TO 400	30002690
350 WRITE(6,1150) PERIOD, CODE(N), (STOP(J,I),J=1,NV)	30002700
GO TO 400	30002710
360 WRITE(6,1160) PERIOD, CODE(N), (STOP(J,I),J=1,NV)	30002720
GO TO 400	30002730
370 WRITE(6,1170) PERIOD, CODE(N), (STOP(J,I),J=1,NV)	30002740
GO TO 400	30002750

380	WRITE(6,1180) PERIOD, CODE(N), (STOP(J,I), J=1, NV)	30002760
	GO TO 400	30002770
390	WRITE(6,1190) PERIOD, CODE(N), (STOP(J,I), J=1, NV)	30002780
400	CONTINUE	30002790
3000	FORMAT(1H1,15X,4A10 / 1H-, *STORED RESOURCES*)	30002800
C		30002810
C	REMAINING PART WILL OUTPUT THE TOTAL AMOUNT USED	30002820
C	DURING EACH PERIOD	30002830
C		30002840
	WRITE(6,2000) (TITLE(I), I=1,4)	30002850
	M=2	30002860
	GO TO 5	30002870
205	N=0	30002880
	DO 300 I=1, NP	30002890
	K=IYR(I) - INYR+1	30002900
	IF(K.LE.0) GO TO 300	30002910
	DO 206 J=1, NV	30002920
206	EXIST(J,K)=EXIST(J,K)-STOP(J,I)	30002930
	N=N+1	30002940
	GO TO (210,220,230,240,250,260,270,280,290), NV	30002950
210	WRITE(6,1110) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30002960
	GO TO 300	30002970
220	WRITE(6,1120) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30002980
	GO TO 300	30002990
230	WRITE(6,1130) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30003000
	GO TO 300	30003010
240	WRITE(6,1140) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30003020
	GO TO 300	30003030
250	WRITE(6,1150) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30003040
	GO TO 300	30003050
260	WRITE(6,1160) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30003060
	GO TO 300	30003070
270	WRITE(6,1170) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30003080
	GO TO 300	30003090
280	WRITE(6,1180) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30003100
	GO TO 300	30003110
290	WRITE(6,1190) PERIOD, CODE(N), (EXIST(J,K), J=1, NV)	30003120
300	CONTINUE	30003130
	RETURN	30003140
2000	FORMAT(1H1,15X,4A10 / 1H-, *TOTAL RESOURCES USED*)	30003150
	END	30003160

	SUBROUTINE SETUP	30003170
	COMMON /VECSIG/ VNAME(10),C,LENP,VLIFE(1.),INH(10,15),	30003180
	* VCOST(10,5),NAME(10),COSTS(30,3)	30003190
	COMMON /BASICS/ CHAP(5000,4),CODE(20),PER(10),IYR(10),LYP(10)	30003200
	COMMON /OUTS/ OANPM(20),SALE(20),SAVE(20),EXIST(10,20),	30003210
	* PUPCH(10,20),STOR(10,20),SALV(10,20),PPCC(20),PROT(20)	30003220
	COMMON /PARAMS/ PDTOT,INYP,LAST,NV,NP,TOT,TITLE(4),COST	30003230
	DIMENSION TEMP(4)	30003240
	DATA IVT,IPT,IED/SHVCHICLE ,8HPERIOD ,8HFNDTABLE /	30003250
	NVR=0	30003260
	NPT=0	30003270
	READ(5,1000)FNAME,INYP,LAST,NV,NT,NP	30003280
1000	FORMAT(A8,2X,6I5)	30003290
10	READ(5,1000)ITARLF	30003300
	IF(ITARLF.EQ.IVT) GO TO 20	30003310
	IF(ITARLF.EQ.IPT) GO TO 60	30003320
	IF(ITARLF.EQ.IED) GO TO 100	30003330
	WRITE(6,2000)ITARLF	30003340
2000	FORMAT(1H-,A8,* IS NOT RECOGNIZED BY SFTUP*)	30003350
	STOP	30003360
20	NVR=NVR+1	30003370
	READ(5,4000)VNAME(NVR),VLIFE(NVR)	30003380
	READ(5,1070)(VCOST(NVR,I),I=1,5)	30003390
1070	FORMAT(5F10.2)	30003400
	GO TO 10	30003410
60	NPT=NPT+1	30003420
	READ(5,1140) IYP(NPT),LYP(NPT),PTR(NPT),PROT(NPT)	30003430
1140	FORMAT(I4,I5,1X,A2,F8.2)	30003440
	GO TO 10	30003450
4000	FORMAT(A8,11X,I2)	30003460
100	CONTINUE	30003470
	PROT(NPT+1) = 0.	30003480
110	CONTINUE	30003490
	READ(9,3000) T,(TEMP(J),J=1,4)	30003500
3000	FORMAT(I5,4X,A1,3A2)	30003510
	IF(EOF,9)200,150	30003520
150	DO 150 J=1,4	30003530
160	CHAP(I,J)=TEMP(J)	30003540
	GO TO 110	30003550
200	RETURN	30003560
	END	30003570

SUBROUTINE VALUES(N, ISTART, IEND, VAL)	30003580
COMMON /VECSTG/ VNAME(10), C, LEND, VLIFE(10), INH(10, 16),	30003590
* VGCOST(10, 5), NAME(10), COSTS(30, 3)	30003600
COMMON /BASICS/ CHAR(5000, 4), CODE(20), PER(10), IYR(10), LYR(10)	30003610
COMMON /OUTS/ OANOM(20), SALE(20), SAVE(20), EXIST(10, 20),	30003620
* PURCH(10, 20), STOR(10, 20), SALV(10, 20), PROC(20), PROT(20)	30003630
COMMON /PARAMS/ PDTOT, INYR, LAST, NV, NP, TOT, TITLE(4), COST	30003640
CALL YRCOST(N)	30003650
I=ISTART-INYR	30003660
II=TEND-ISTART+1	30003670
K=1	30003680
DO 10 J=1, II	30003690
I=I+1	30003700
IF(I.LE.0) GO TO 10	30003710
OANOM(I)=OANOM(I)+COSTS(J, K)*VAL	30003720
EXIST(N, I)=EXIST(N, I)+VAL	30003730
10 CONTINUE	30003740
I=I+1	30003750
K=K+1	30003760
IF(IEND.EQ.LAST) K=K+1	30003770
SALE(I)=COSTS(J, K)*VAL+SALE(I)	30003780
SALV(N, I)=VAL+SALV(N, I)	30003790
RETURN	30003800
END	30003810

```

SUBROUTINE VRCOST(J)
C A SUBROUTINE TO COMPUTE THE OPERATING, SALVAGE, AND TRUNCATION
C COSTS YEAR BY YEAR. ALSO THE YEARLY MOTHRALLING SAVING IS COMPUTED.
COMMON /VEGSTG/ VNAME(10), C,LENP, VLIFE(10), TNH(10,16),
* VCOST(10,5), NAMFN(10), COSTS(30,3)
INTEGER VNAME,VLIFE
C ASSUME THE OPERATING AND MAINTANCE COST INCEPEACES AT R*100 PER-CENT
C A YEAR (NOT A COMPOUND PATE INCEASE)
P=0.0
C
C LET X= THE 1ST YEAR O. AND M. COST. THEN
C  $X + (1+R)*X + (1+2R)*X + \dots + (1+9R)*X = VCOST(J,2)$ 
X= VCOST(J,2)/(10.0 + 45.0*R)
C ASSUME NO PERIOD IS LONGER THAN 6 YEARS.
IB=VLIFE(J) +10
DO 10 I=1,IB
COSTS(I,1)=(1.0 + FLOAT(I-1)*R)*X*(VCOST(J,4)**(I-1))
10 CONTINUE
C
C ASSUME THE SALVAGE VALUE OF A VEHICLE AFTER I YEARS OF SERVICE IS
C (ALPHA)**I *PURCHASE COST.
ALPHA=0.5
Y=VCOST(J,1)
DO 20 I=1,IB
Y= ALPHA*Y
COSTS(I,2)=Y
20 CONTINUE
C
C ASSUME TRUNCATION AFTER IYEAPS OF SERVICE IS
C (VLIFE-T)*(PURCHASE COST)/VLIFE
Y=VCOST(J,1)/VLIFE(J)
DO 30 I=1,IB
IX=VLIFE(J)-I
IF (IX.LT.0) IX=0
COSTS(I,3)=IX*Y
30 CONTINUE
RETURN
ENTRY MOTH
C ASSUME THE MOTHRALLING SAVING IS P1*100 PER CENT OF THE FIRST YEAR COST-X
P1=0.90
C
C C=0
C DO 546 IL=1,LENP
C 546 C=C-C.1*P1*VCOST(J,2)*VCOST(J,4)**(IL-1)
C C =-X * P1
C=-VCOST(J,2)/(10.0 + 45.0*R) * P1
RETURN
END

```

```

10008670
10008640
10008650
10008660
10008670
10008680
10008690
10008700
10008710
10008720
10008730
10008740
10008750
10008760
10008770
10008780
10008790
10008800
10008810
10008820
10008830
10008850
10008860
10008870
10008880
10008890
10008900
10008910
10008920
10008930
10008940
10008950
10008960
10008970
10008980
10008990
10009000
10009010
10009080
10009090

```

APPENDIX E
ERROR MESSAGES

ERROR MESSAGES FROM MATRIX GENERATOR

". . . is not a table name."

This message indicates that the input deck is not properly constructed since the program has read a card which should have been a header card but was not. The location of the error can be narrowed down by checking the output listing to see which tables have been correctly read. This error terminates execution.

"Vehicle name . . . not defined in a vehicle."

This message is output when a task table is being read. It indicates either that the vehicle name is misspelled or located improperly on the card, or that the task table has preceded the vehicle table, if the vehicle table exists. This error also terminates execution.

"The period tables are out of order."

This message indicates that the first year of the period just read was not equal to one plus the last year of the last period read. This can be caused by improper sequencing or improper definition of the periods. This error will cause execution to terminate.

"Warning--the number of tables input was not the expected number."

This message does not terminate execution, but does indicate that there was a difference between the number of tables indicated on the title card and the number actually read by the program.

"Incorrectly read file . . . columns read as . . ."

"The M-1 column was . . ., unable to find RHS mark."

"Reached EOF while writing column . . . and row . . ."

These three error messages all refer to errors encountered when trying to convert the MPS360 file to the file for BBCAV2. If one of

these errors occurs, a major problem exists within the program. As a result these prevent the creation of the BBCAV2 file, but allow the program to execute to completion to give the analyst the most information possible about the problem.

ERROR MESSAGES FROM MAIN PROGRAM

"BLIST size exceeded"

BLIST is the array used for storing nodes of the branching tree. It is presently dimensioned to handle 25 nodes. When there exist more than 25 nodes which have been defined but have not been evaluated, this message is generated. It indicates that this particular problem is converging very slowly, and if one desires an accurate answer then the arrays EKBL, PSIGL, NXBL, XNXBL, and BLIST should be enlarged. This error causes the system to print out the best solution found and proceed to the next problem.

"Time is up . . . cycling to next problem."

"Have solved max. no. of LP probs."

These two messages inform one that the solution which is output is not necessarily optimal but was caused by one of the input parameters. The first message indicates a violation of the time indicated by the second field of the real parameter card. The second message is a result of reaching the limit on the number of nodes (LP problems) which is set in the last field of the integer parameter card.

"Premature EOF on a matrix tape at column . . ."

This message indicates that the size of the tape file does not correspond with the size indicated on the integer parameter card. It also gives an indication of the size of the tape file for comparison against that which was input. This error terminates execution of the program.

"IO--column not located in NT reads"
"Row--key not in core"
"Insert cannot find rejected column"
"PIVOT--PIVOT less than PIVTOL"
"KEYCH--essential packet no basic column"

These five error messages are used exclusively for debugging, and should not occur in normal operation. The general cause for this is that some section of core has been overwritten accidentally.

"PIVOT dropped column . . ."

This message indicates that a column was removed from the basis during the inversion process. This occurs when the input basis is not feasible, and when numerical errors have caused the current basis to "drift" out of the feasible region.

ERROR MESSAGES FROM REPORT GENERATOR

". . . is not recognized by SETUP."

The routine SETUP has encountered an error in the input deck while attempting to read a table name. This error terminates execution.

"Error in interpretation of . . ."

The program has been unable to determine the meaning of the seven-character code indicated in the message. If the code is a valid one (one of the forms shown in Fig. 11), then there is probably an error in the period descriptions of the input deck. There is also possibility of other errors in the input deck or, as a last resort, of errors in the reference list file. If the code is not a valid one, then the error must be in the reference list. This error also terminates execution.

GLOSSARY

This section contains the mnemonic definitions for all three programs—GENLCP, BBCAV2, and REPGEN. It is arranged into two major sections. The first section lists the mnemonics in labeled common— then the local variables contained within each subroutine, for each of the three programs, respectively. The second section is an overall alphabetical listing for handy reference. Note that in this listing, the same mnemonic may have two or more meanings. Each entry is identified here as a local or global variable, and is cross-referenced to the first section. Use of the two sections, in conjunction, should eliminate any ambiguity.

SECTION 1.....G-2

SECTION 2.....G-25

Section I

GENLCP CODING DEFINITIONS

COMMON/VECSTG/

VNAME (10) - vehicle names
C - temporary storage for cost data
LENP - length of period
VLIFE (10) - maximum life of resource (vehicle)
INH (10, 16) - number of each type resource inherited from each year
VCOST (10,5) - cost data for each resource
NAMEN (10) - pointers for numbering resources
COST (30, 3) - yearly operating, salvage and truncation costs

COMMON/ALTSTG/

ALTER (288,9) - array used for eliminating infeasible alternatives from tasks
YAVL (10) - year resource first available

COMMON/TSKSTG/

U (7, 288, 9) - array of task alternatives
NTSK (9) - number of alternatives in task

COMMON/PRDSTG/

NPERYR (10, 3) - first and last year of period and number of tasks in period
NPTASK (10, 9) - ID number of each task in period
PTASK (10, 9) - multiplicative factor for all values in associated task for each period

LOCAL VARIABLES

GENLCP

ALPHA - temporary storage for attrition
AU (16) - temporary storage for alternatives

BUDG (10)	- limit on procurement expenditures in each period
CMAX	- temporary cost storage for ordered resources
FNAME	- file name
IHVN (10)	- pointers for inherited vehicles
INHYRS	- number of years from which vehicles are inherited
ITABLE	- temporary storage for table name
LIFER	- temporary storage for remaining useful life of a vehicle
LY	- last year of problem
MAXL	- temporary storage for vehicle life
MCOL	- number of columns in matrix
NAMES (10)	- temporary pointers
NINHP	- number of inherited periods
NIV	- number of inherited vehicle types
NL (10)	} temporary storage used in formatting output
NN (10)	
NPP	- number of periods
NPT	- number of period tables read
NRD	- number of vehicles having R&D
NROW	- number of rows in matrix
NT	- number of tasks
NTR	- number of task tables read
NV	- number of vehicle types
NVEHU (10)	- indicates if vehicle used in period
NVR	- number of vehicle tables read
NYR	- temporary storage for last year of period
ONE	- "1.0"
ONEM	- "-1.0"

SY - start year of problem

UB (10) - calculated upper bounds on resources

UMAX - temporary storage for greatest quantity of a specific vehicle which might be used in a task

YEARS (21) - stores inherited years

YRINT (20) - scale factor for all tasks in period

YRCOST

ALPHA - rate of decrease in salvage value

R - rate of increase in operating cost

RL - portion of operating cost refunded for mothballing resource

YINTERP

JSUB (10) - pointers for vehicle subscripts

VMIN - temporary storage for minimum quantity of vehicles which can be used for a task

MATFILL

C - "COLUMNS"

CNAME - column name for which RVAL is being derived

CTEMP - temporary storage for column name

IROWTP (100) - indicates row type; all set to zero except generalized upper bound rows which are set to 4

ITEMP - temporary storage for first letter of RNAME

R - "RHS"

RNAME (120) - row names

RTEMP - temporary storage for row names

RVAL (100) - vector of values in each row for a specific column

VAL - temporary storage for value of specific row and column

BBCAV2 CODING DEFINITIONS

COMMON/CV1/

- IP (12) - storage for input parameters on integer parameter card
- RP (12) - storage for real parameters, first four locations are for input from real parameter card, rest are temporary storage
- TMP (10) - temporary storage

COMMON/CV2/

- T (100, 10) - storage for columns of matrix associated with nonlinear variables
- BO (100) - right-hand-side vector
- BLO (10) - set of lower bounds on nonlinear variables
- ULO (10) - set of upper bounds on nonlinear variables
- CO (10) - vector for linear approximation for nonlinear cost functions

COMMON/CV3/

- M - number of rows in matrix
- N - number of columns in matrix
- NCF - number of nonlinear variables
- PHIT - cost of a nonlinear solution
- UZ - cost of best nonlinear solution
- USP - $UZ (1 + \epsilon)^{-1}$
- USM - $UZ (1 - \epsilon)^{-1}$
- EKO - cost associated with the lower bounds of the node
- MPLUS - number of rows in the matrix including the cost row (M + 1)

COMMON/CV4/

- IX (110) - columns in basic solution
- X (110) - values associated with columns in IX

- IXZ (110) - columns in best solution
- XZ (110) - values associated with columns in IXZ
- XCON (10) - stores values found in X which are associated with the nonlinear variables
- COST - cost of the solution returned from the LP

COMMON/CV5/

- SIGMA (100, 4) - stores information which defines the current node
- TSIG - temporary storage associated with EKO
- LSTMAX - maximum length which the branching list has achieved

COMMON/CV7/

- NPHASE - stores LP phase code
- NF1 - signifies feasible solution when set equal to 1
- CFX - no longer used
- IOPT - used to flag unbounded solution
- NOP - node number
- NOPS - nodes solved
- NEWXZ - flags when new best solution found and should be output

COMMON/CV8/

- NXBK - index of branching variable
- XK - value of branching variable
- NOBOL - number of nodes on list
- EKBL (25) - EKO value associated with each node on the list

COMMON/CV9/

- PSIGL (25) - lower bound associated with each node on list
- NXBL (25) - index of branching variable for each node
- XNXBL (25) - value of branching variable for each node
- BLIST (25, 131) - branching list; contains right-hand-side vector, plus upper bounds, lower bounds, and linear cost approximations for nonlinear variables

COMMON/TMX/

TMO - time SET was called
EXT - time when time limit on problem will expire

COMMON/CORE/

AJ (5000) - columns in core plus basis inverse
JA (101) - in-core column disc indices
JAK (101) - dummy storage area
JAREJ (101) - set to 1 when corresponding in-core column rejected

COMMON/PARAMS/

TMAX - maximum time before MAPOUT
ITNINV - iteration of next invert
INVF - invert frequency
K1 - not used
K2 - not used
K3 - output control parameter
K4 - XCHECK control parameter
K5 - maximum LP iterations before MAPOUT

COMMON/INPUT/

INPUT - file containing input matrix
INPUM - number of rows in matrix
INPUTN - number of columns in matrix

COMMON/FILES/

IA1 - disc file for matrix less GUB rows
IA2 - disc file for packed matrix less GUB rows
IMAP - file for starting and terminating basis

COMMON/STATE/

IROW - current selected row

ITRN - iteration count
 JCOL - current selected column
 JOUT - rejected column index
 JPOS - selected column index
 NDJS - number of negative DJ's
 NPIF - number of primal infeasibilities
 NREJ - number of rejected in-core columns

COMMON/LIMS/

JNCORE - number of columns in core
 MAXTRY - maximum number of in-core iterations
 NCRMAX - maximum number of columns which fit in core
 NSCAN - number of disc reads
 NTRY - number of in-core iterations

COMMON/IXX/

IX (100) - indices of solution columns

COMMON/XX/

X (100) - values of solution columns

COMMON/TOLS/

CTOL - cost tolerance for infeasibility
 DERTOL - dual error tolerance; not used
 DJTOL - DJ tolerance
 PERTOL - primal error tolerance
 PIVTOL - pivot tolerance
 ZERO - smallest recognized number

COMMON/BASIS/

IBASIS (101) - basic columns for non-GUB rows
 KEYS (101) - storage of GUB key columns

COMMON/DJS/

DJ (100) - values of current in-core DJ's

COMMON/MOVES/

BNDJ - value of current column bound

DMAX - largest DJ value stored

DUALER - dual error; unused

PRMLER - primal error; unused

THETA - step chosen by ROW, adjusted in PRIMAL

COMMON/I/

IC - current cost row

ICOST - user's cost row

IPHASE - current LP phase

IPI - current location of PI vector in basis

JRHS - user's input RHS

L - number of GUB rows

M - number of active interval rows

MC - last logical column

MPL - M plus L

NT - total number of columns (MC + INPUTN)

COMMON/A/

ALPHA (101) - work space, usually current column inverse

COMMON/B/

BETA (101) - work space usually values of basic and key variables

COMMON/C/

GAMMA (101) - not used

COMMON/D/

DELTA (101) - not used

COMMON/ROWTYP/

IROWTYP (101) - user's input row types

COMMON/NAMES/

NAME (600) - state of each variable or column

COMMON/BOUNDS/

BOUNDS (100) - values of upper bounds

IBDS (100) - column indices of bound columns

NBDS - number of bounds

COMMON/RHS/

RHS (100) - stores user's current right-hand-side

LOCAL VARIABLES

BBCAV2 and BOX1

BLO (10) - temporary storage for BLO

BBK - lower bound on branching variable

BBK2 - value of branching variable

COST1 - solution cost for lower branch

COST2 - solution cost for upper branch

CO (10) - temporary storage for CO

EPSI - epsilon value from real parameter card

ESIG - temporary storage for EKO

ICOL - temporary storage for column index

INDIC - indicates which branch (upper or lower) is being solved

LSTFRE (25) - gives locations of storage areas on the branching list which are vacant

MNC - the negative of NCF

MIN - the negative of N

NCF1 - NCF

NCF4 - $NORA + 3 * NCF$

NFREE - number of gaps (empty location between two filled locations) in the BLIST

NMIN - index of the lowest bound on the BLIST, or N-1, depending on where it is used

NOL - index for storage on BLIST

NORA - M

NXB - temporary storage for next branching variable

PH1 and PH2 - temporary storage of values from GETPHI

PMIN - value of lowest bound on BLIST

TSTO (130) - temporary storage

TITLE (4) - alphanumeric title of problem

UBK - upper bound on branching variable

UBK2 - difference between upper bound and value for branching variable

ULT (10) - temporary storage for ULO

US - temporary storage for USP

INITA

AJ (100) - temporary storage for column of matrix

DUM1 and DUM2 - temporary storage for reading unused sections of tape

READIN

NC - number of basis cards to be read from input

TIMEC

SECS - actual CPU clock time

XX - elapsed time on problem

GETASQ

TEMP - location used while swapping contents of two locations in an array

NEM1 - number of elements in an array minus one

GETC

- DIF - difference between upper and lower bound for a variable
- FX1 (10) - cost function values for lower bounds
- FX2 (10) - cost function values for upper bounds
- ICX - number of variables for which cost slopes are to be derived

NXBRN

- BLT (10) - temporary storage for BLO
- CT (10) - temporary storage for CO
- YT (10) - differences between solution point and lower bounds
- FX1 (10) - cost function values for lower bounds
- FX2 (10) - cost function values for solution point
- DIF (10) - differences between cost functions and linear approximations
- NDX (10) - indices of nonlinear variables
- NFX - the negative of NCF
- XT (10) - solution values for nonlinear variables

PRESET AND PARAMS

- IBMAX - maximum number of nodes which may be stored on BLIST
- JBMAX - maximum number of words of information which may be stored for each node in BLIST
- NORA - number of rows in the matrix including the objective function

LP

- B (100) - basis inverse stored by rows
- IORG - origin of basis inverse
- MROWS - user's number of rows
- NCHGS - user's number of bound columns

NCOLS - user's number of columns
NWAJ - storage dimension of the array AJ

SETUP

ID - local row type being processed
IKOST - temporary storage of user's cost row
INON - temporary number of non-GUB rows found
PKT1 - temporary count of GUB row packet columns
PKT - actual GUB row column being processed

IO

ALPHA - column to be written or read
B - address of origin of basis inverse
JCOL - core position of column being read
JNT - index of columns read
KEY - index of key column to be located
KOL1 - last column read on file IA1
KOL2 - last column read on file IA2
KOL - column to be located on either file or packet number of desired key
NAAM - not used
NAME - column name, or position in core to which column is read
PACK (100) - temporary storage of packed column
ZS - parameter used to pack coefficients
Z - parameter used to pack index of coefficient

MAPIN

ATBND - "ATBND"
BASIC - "BASIC"
BNDJ - value of bound
B - origin of basis inverse

CARD (8) - image of map card
 ENDER - "END"
 ID - column number from map card
 INVERSE - "INVERS"
 KEE - "KEY"
 MM - number of elements in basis inverse
 NAMES (5) - column indices from map card
 NULL - "NULL"
 PKT - storage of column packet
 ROWS - "ROWS"
 TYPE1 - first word on map card
 TYPE2 - second word on map card

MAPOUT

IBAS - count of basis variables
 IBND - count of bound variables
 IKEY - count of key variables
 INLL - count of null variables
 JCOL - user's column index of column processed
 JNCORE - number of columns in core
 MAPBAS (100) - basic column indices
 MAPBND (100) - bound column indices
 MAPKEY (1000) - key column indices
 MAPNLL (10) - null column indices
 MM - number of elements in basis inverse
 MP1 - M plus 1

INVERT

ATBND - column type
 BASIC - column type

BNDJ - bound on current column
 B - basis inverse
 IORG - origin of first element in B
 ITYPE - row type
 JNT - current column index
 JORG - origin in AJ to which column is read
 JTYPE - variable type
 KORG - origin in AJ to which key column is read
 PKTO - GUB packet number of column in AJ (KORG)
 PKT - GUB packet number of column being processed

FEASCH

BNDJ - bound on current column
 B - basis inverse
 IORG - origin of any row in B
 JPKT - GUB packet of current column
 KEY - switch to return key processing to key loop
 NB - number of basic variables in a packet
 SUMIE - sum of infeasibilities
 SUM - value of variable before feasibility adjustment

PRIMAL

BASIC - column type
 B - basis inverse
 EPSI - value of new basic variable
 ITYPE - type of step to be used
 JOUTPK - GUB packet of column rejected
 JPOSPK - GUB packet of column entering
 NBVPKT - number of basis variable in selected GUB row

NPEGLM - maximum rejection due to degeneracy
 NDEG - number of degeneracy rejections
 NEWROW - row for column changing from key to basic

STATUS

ATBND - state of a column
 BASIC - state of a column
 B - basis inverse
 COST - value of current objective function
 JNSCAN - columns in core + 1000 times number of rewinds
 of file IAl
 JNTO - index of last column read from disc
 JNT - last column read from disc (if MNTRY = 0)
 MNTRY - number of in-core iterations
 NCOLS - number of columns read from disc file IAl
 NJOUT - name code of column to be rejected
 NOTE (4) - 40 character comment
 X - elapsed CPU seconds

ROW

BASIC - state of a column
 B - basis inverse
 IB - basic column index
 IORG - origin of basis inverse
 IROW - row calling parameter, row of zero
 ITYPE - type of step; 1-unbounded, 2-column to zero,
 3-column to bound
 JCOL - core index of selected column
 JORG - core origin of selected column
 JOUT - column to be rejected
 JPKT - GUB packet of column selected

JPOS - disc index of column selected
 KORG - origin of KEY column for packet JPKT
 STEP - step to current row
 THETA - best feasible step

COLUMN

ATBND - logical column state
 BASIC - logical column state
 B - basis inverse
 JCOL - core position of selected column
 JKEY - core position of key for JCOL (if in GUB row)
 JORG - origin of a row in B
 JPKTO - current stored GUB key packet
 JPKT - GUB packet of new column
 JTYPE - type of column selected
 KORG - origin of KEY in AJ
 NCORE - number of columns in core
 NDJST - number of negative DJ's from disk read
 NULL - column state
 PIKEY - PJ value for current KEY JPKT

CHECK

ATBND - state of column
 BASIC - state of column
 B - basis inverse
 DJ - current column sensitivity
 JCOUNT - count of columns processed
 JFBCH - number of columns, checked in current batch
 JNT - index of current column

JORG - origin in AJ to which columns are read
 JTYPE - type of column being processed
 KORG - origin of key column in AJ
 NBCH - number of columns in batch
 NFBCH - number of columns retained from batch
 PIKEY - DJ for current key at KORG
 PKTO - packet of current key
 PKT - packet of new column, JNT

INSERT

B - basis inverse
 DJ - DJ for column to be stored
 DMAX - largest DJ of stored columns
 D (15) - DJ's of stored columns
 ID (15) - indices of stored columns
 JORG - origin of vacancy for column in AJ
 JPOSR - disc index of column to be rejected
 JPOS - disc index of column to be stored
 JREJ - origin of rejected column in AJ
 NPBCH - number of columns to be saved from batch
 N - number of columns currently saved

KEYCH

B - basis inverse
 IB - disc index of basic column for current row
 IORG - origin of a row in B
 IROW - row to which key column is shifted when made basic
 JCOLPK - GUB packet of column being moved from KEY
 JCOL - column to be moved

JKEY - candidate key column
JORG - origin of a row in B
MPK - row of column which was KEY
SUM - temporary storage

PIVOT

ALPHA - column to be pivoted into basis
B - basis inverse
DIVOT - candidate pivot while searching for best
IORG - origin of pivot row in B
IROW - pivot row
JORG - origin of a row in B
JP - basic column for a row
PIV - pivot used

SETBND

I - input disk column index
J - absolute value of I
K - new state

DOT

DOT - double precision inner product of X and Y
DOTS - single precision inner product of X and Y
M - vector dimension
SUM - double precision accumulator
X - input vector
Y - input vector

BOUND

BOUND - value of column bound (or 10^{**70})
IB - bound index in IBDS
J - input disc column index

KEYFIND

I - dummy variable
JAJ - potential column's in-core position
JPKT - GUB packet number for column
JTYPE - column type
KEYFIND - position of key found
KEY - column number of key to be located
PKT - GUB packet of desired key

ESCAPE

AALPHA - "ALPHA"
ABASIS - "BASIS"
ABETA - "BETA"
ADELTA - "DELTA"
ADJ - "DJ"
AGAMMA - "GAMMA"
AJAREJ - "JAREJ"
AJA - "JA"
AKEY - "KEY"
ANAME - "NAME"
B - basis inverse

XCHECK

ATBND - logical column state
AT - dummy
BASIC - logical column state
B - basis inverse
CALLER - calling name
IORG - origin of a row in basis inverse

JAJ - disk index of an in-core column
JEND - origin of vacant work space in AJ
JORG - origin of a column in AJ
J1 - first column in column printout
J2 - last column in column printout

REPGEN CODING DEFINITIONS

COMMON/VECSTG/

- VNAME (10) - stores resource names
- C - temporary storage location used in calculating savings from resource storage
- LENP - length of period under consideration
- VLIFE (10) - expected resource life
- INH (10, 16) - not used
- VCOST (10, 5) - the five costs associated with each resource are stored in this array; in order, they are salvage and truncation, operating, R&D, retention rate, and procurement. (Explained in detail in matrix generator description.)
- NAMEN (10) - not used
- COSTS (30, 3) - cost of operating (1), selling (2), or truncating (3), a resource in the 1st thru 30th year of its life

COMMON/BASICS/

- CHAR (5000, 4) - storage of column names which have been broken down into their four meaningful parts
- CODE (20) - storage of the numbers 1 - 20 in two digit alphanumeric form
- PER (10) - pointers for two digit, alphanumeric code for periods
- IYR (10) - initial year of each period
- LYR (10) - last year of each period

COMMON/OUTS/

- OANDM (20) - operating cost for each year
- SALE (20) - salvage or truncation value for each year
- SAVE (20) - savings from resource storage for each year
- EXIST (10, 20) - number of each type resource available in each year

- PURCH (10, 20) - number of each type resource purchased in each year
- STOR (10, 20) - number of each type resource stored in each year
- SALV (10, 20) - number of each type resource disposed of at end of each year
- PROC (20) - procurement funds spent during each period
- PROT (20) - procurement funds available during each period

COMMON/PARAMS/

- RDTOT - total R&D expenditures
- INYR - initial year of problem
- LAST - last year of problem
- NV - number of resource types
- NP - number of subperiods
- TOT - number of subperiods plus 1
- TITLE (4) - name of specific solution
- COST - total cost of solution

LOCAL VARIABLES

SETUP

- FNAME - problem title (not used)
- IED - "ENDTABLE"
- IPT - "PERIOD"
- ITABLE - table name
- IVT - "VEHICLE"
- NPT - period tables read in
- NT - number of tasks (not used)
- NVR - number of resources read in
- TEMP (4) - temporary storage for column names

INSOLN

- BLANK - " "

IEND - last year of resource existence
 IND - column number temporary storage
 ISTART - first year of resource existence
 S - "S"
 VAL - column value temporary storage
 W - "W"
 X - "X"

CINFD

PERIOD - "PERIOD"
 ONE - "01"
 SUM - total cost for a period
 SUMT - total cost for all periods
 TCOST - temporary storage for total procurement
 TOTAL - "TOTAL"
 TPROC - correction factor for procurement

PIINFO

TEMP1 - }
 TEMP2 - } temporary storage locations for alphanumeric output
 BLANK - " "

PERIOD - "PERIOD"
 TOTAL - "TOTAL"

AALPHA - 'ALPHA'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 ABASIS - 'BASIS'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 ABETA - 'BETA'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 ADELTA - 'DELTA'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 ADJ - 'DJ'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 AGAMMA - 'GAMMA'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 AJAREJ - 'JAREJ'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 AJA - 'JA'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 AJ(5000) - COLUMNS IN CORE PLUS BASIS INVERSE
 (GLOBAL - MAIN PROGRAM'S COMMON / CORE /)
 AJ(100) - TEMPORARY STORAGE FOR COLUMN OF MATRIX
 (LOCAL - MAIN PROGRAM'S INITA)
 AKEY - 'KEY'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 ALPHA(101) - WORK SPACE, USUALLY CURRENT COLUMN INVERSE
 (GLOBAL - MAIN PROGRAM'S COMMON / A /)
 ALPHA - COLUMN TO BE WRITTEN OR READ
 (LOCAL - MAIN PROGRAM'S IO)
 ALPHA - TEMP STORAGE FOR ATTRITION
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 ALPHA - RATE OF DECREASE IN SALVAGE VALUE
 (LOCAL - MATRIX GENERATOR'S YRCOST)
 ALPHA - COLUMN TO BE PIVOTED INTO BASIS
 (LOCAL - MAIN PROGRAM'S PIVOT)
 ALTER (288, 9) - ARRAY USED FOR ELIMINATING INFEASIBLE ALTERNATIVES FROM TASK
 (GLOBAL - MATRIX GENERATOR'S COMMON / ALTSTG /)
 ANAME - 'NAME'
 (LOCAL - MAIN PROGRAM'S ESCAPE)
 AT - DUMMY
 (LOCAL - MAIN PROGRAM'S XCHECK)
 ATBND - 'ATBND'
 (LOCAL - MAIN PROGRAM'S MAPIN)
 ATBND - COLUMN TYPE
 (LOCAL - MAIN PROGRAM'S INVERT)
 ATBND - LOGICAL COLUMN STATE
 (LOCAL - MAIN PROGRAM'S XCHECK)
 (LOCAL - MAIN PROGRAM'S CHECK)
 (LOCAL - MAIN PROGRAM'S COLUMN)
 (LOCAL - MAIN PROGRAM'S STATUS)
 AU (16) - TEMP. STORAGE FOR ALTERNATIVES
 (LOCAL - MATRIX GENERATOR'S GENLCP)

B - BASIS INVERSE

(LOCAL - MAIN PROGRAMS LP)
(LOCAL - MAIN PROGRAMS XCHECK)
(LOCAL - MAIN PROGRAMS ESCAPE)
(LOCAL - MAIN PROGRAMS PIVOT)
(LOCAL - MAIN PROGRAMS KEYCH)
(LOCAL - MAIN PROGRAMS INSERT)
(LOCAL - MAIN PROGRAMS CHECK)
(LOCAL - MAIN PROGRAMS COLUMN)
(LOCAL - MAIN PROGRAMS ROW)
(LOCAL - MAIN PROGRAMS STATUS)
(LOCAL - MAIN PROGRAMS PRIMAL)
(LOCAL - MAIN PROGRAMS FEASCH)
(LOCAL - MAIN PROGRAMS INVERT)
(LOCAL - MAIN PROGRAMS MAPIN)
(LOCAL - MAIN PROGRAMS IO)

BASIC - BASIC
(LOCAL - MAIN PROGRAMS MAPIN)

BASIC - COLUMN TYPE
(LOCAL - MAIN PROGRAMS INVERT)
(LOCAL - MAIN PROGRAMS PRIMAL)

BASIC - LOGICAL COLUMN STATE
(LOCAL - MAIN PROGRAMS XCHECK)
(LOCAL - MAIN PROGRAMS CHECK)
(LOCAL - MAIN PROGRAMS COLUMN)
(LOCAL - MAIN PROGRAMS STATUS)
(LOCAL - MAIN PROGRAMS ROW)

BRK - LOWER BOUND ON BRANCHING VARIABLE
(LOCAL - MAIN PROGRAMS BRCAV2)

BRK2 - VALUE OF BRANCHING VARIABLE
(LOCAL - MAIN PROGRAMS BRCAV2)

BETA(10) - WORK SPACE, USUALLY VALUES OF BASIC AND KEY VARIABLES
(GLOBAL - MAIN PROGRAMS COMMON / B /)

BLANK - ' '
(LOCAL - REPORT GENERATOR'S INSOIN)
(LOCAL - REPORT GENERATOR'S PINFO)

BLIST(25,13) - BRANCHING LIST, CONTAINS RIGHT-HAND-SIDE VECTOR, PLUS UPPER BOUNDS, LOWER BOUNDS AND LINEAR COST APPROXIMATIONS FOR NON-LINEAR VARIABLE
(GLOBAL - MAIN PROGRAMS COMMON / CV9 /)

BLO(10) - SET OF LOWER BOUNDS ON NON-LINEAR VARIABLES
(GLOBAL - MAIN PROGRAMS COMMON / CV2 /)

BLT(10) - TEMPORARY STORAGE FOR BLO
(LOCAL - MAIN PROGRAMS BRCAV2)
(LOCAL - MAIN PROGRAMS NXBRN)

BNDJ - BOUND ON CURRENT COLUMN
(GLOBAL - MAIN PROGRAMS COMMON / MOVES /)

BOUNDS(100) - VALUES OF UPPER BOUNDS
(GLOBAL - MAIN PROGRAMS COMMON / BOUNDS /)

BOUND - VALUE OF COLUMN BOUND (OF 10 ** 70)
(LOCAL - MAIN PROGRAMS BOUND)

BO(100) - RIGHT-HAND-SIDE VECTOR
(GLOBAL - MAIN PROGRAMS COMMON / CV2 /)

BUDG (10) - LIMIT ON PROCUREMENT EXPENDITURES IN EACH PERIOD
(LOCAL - MATRIX GENERATOR'S GENLCP)

C - TEMP. STORAGE FOR COST DATA
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)

C - TEMPORARY STORAGE LOCATION USED IN CALCULATING SAVINGS FROM RESOURCE STORAGE
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)

C - 'COLUMNS'
 (LOCAL - MATRIX GENERATOR'S MATFILL)

CALLER - CALLING NAME
 (LOCAL - MAIN PROGRAM'S XCHECK)

CARD(8) - IMAGE OF MAP CARD
 (LOCAL - MAIN PROGRAM'S MAPIN)

CFX - NO LONGER USED
 (GLOBAL - MAIN PROGRAM'S COMMON / CV7 /)

CHAR(5000,4) - STORAGE OF COLUMN NAMES WHICH HAVE BEEN BROKEN DOWN INTO THEIR FOUR MEANINGFUL PARTS
 (GLOBAL - REPORT GENERATOR'S COMMON / BASICS /)

CMAX - TEMP. COST STORAGE FOR ORDERING RESOURCES
 (LOCAL - MATRIX GENERATOR'S GENLCP)

CNAME - COLUMN NAME FOR WHICH RVAL IS BEING DERIVED
 (LOCAL - MATRIX GENERATOR'S MATFILL)

CODE(20) - STORAGE OF THE NUMBERS 1 - 20 IN TWO DIGIT ALPHANUMERIC FORM
 (GLOBAL - REPORT GENERATOR'S COMMON / BASICS /)

COST - VALUE OF CURRENT OBJECTIVE FUNCTION
 (LOCAL - MAIN PROGRAM'S STATUS)

COST - TOTAL COST OF SOLUTION
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)

COST - COST OF THE SOLUTION RETURNED FROM THE LP
 (GLOBAL - MAIN PROGRAM'S COMMON / CV4 /)

COST1 - SOLUTION COST FOR LOWER BRANCH
 (LOCAL - MAIN PROGRAM'S BRCAV2)

COST2 - SOLUTION COST FOR UPPER BRANCH
 (LOCAL - MAIN PROGRAM'S BRCAV2)

COSTS (30,3) - COST OF OPERATING (1), SELLING (2) OR TRUNCATING (3) A RESOURCE IN THE 1ST THRU 30TH YEAR OF ITS LIFE
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)

CO(10) - VECTOR FOR LINEAR APPROXIMATION FOR NON-LINEAR COST FUNCTIONS
 (GLOBAL - MAIN PROGRAM'S COMMON / CV2 /)

CTEMP - TEMP. STORAGE FOR COLUMN NAME
 (LOCAL - MATRIX GENERATOR'S MATFILL)

CTOL - COST TOLFRANCE FOR INFEASIBILITY
 (GLOBAL - MAIN PROGRAM'S COMMON / TOLS /)

CT(10) - TEMPORARY STORAGE FOR CO
 (LOCAL - MAIN PROGRAM'S BRCAV2)
 (LOCAL - MAIN PROGRAM'S NXPRN)

DELTA(101) - NOT USED
 (GLOBAL - MAIN PROGRAM'S COMMON / D /)

DERTOL - DUAL ERROR TOLERANCE, NOT USED
 (GLOBAL - MAIN PROGRAM'S COMMON / TOLS /)

DIF - DIFFERENCE BETWEEN UPPER AND LOWER BOUND FOR A VARIABLE
 (LOCAL - MAIN PROGRAM'S GFTC)

DIF(10) - DIFFERENCES BETWEEN COST FUNCTIONS AND LINEAR APPROXIMATIONS
 (LOCAL - MAIN PROGRAM'S NXPRN)

DIVOT - CANDIDATE PIVOT WHILE SEARCHING FOR BEST
 (LOCAL - MAIN PROGRAM'S PIVOT)

DJ(100) - VALUES OF CURRENT IN-CORE DJ'S
 (GLOBAL - MAIN PROGRAM'S COMMON / DJ'S /)

DJ - DJ FOR COLUMN TO BE STORED
 (LOCAL - MAIN PROGRAM'S INSERT)

DJ - CURRENT COLUMN SENSITIVITY
 (LOCAL - MAIN PROGRAM'S CHECK)

DJTOL - DJ TOLERANCE
 (GLOBAL - MAIN PROGRAM'S COMMON / TOLS /)

DMAX - LARGEST DJ OF STORED COLUMNS
 (GLOBAL - MAIN PROGRAM'S COMMON / MOVES /)

DOT - DOUBLE PRECISION INNER PRODUCT OF X AND Y
 (LOCAL - MAIN PROGRAM'S DOT)

DOT5 - SINGLE PRECISION INNER PRODUCT OF X AND Y
 (LOCAL - MAIN PROGRAM'S DOT)

D(15) - DJ'S OF STORED COLUMNS
 (LOCAL - MAIN PROGRAM'S INSERT)

DUALER - DUAL ERROR, UNUSED
 (GLOBAL - MAIN PROGRAM'S COMMON / MOVES /)

DUM1 AND DUM2 - TEMPORARY STORAGE FOR READING UNUSED SECTIONS OF TAPE
 (LOCAL - MAIN PROGRAM'S INITA)

EKBL(25) - EKO VALUE ASSOCIATED WITH EACH NODE ON THE LIST
 (GLOBAL - MAIN PROGRAM'S COMMON / CVR /)

EKO - COST ASSOCIATED WITH THE LOWER BOUNDS OF THE NODE
 (GLOBAL - MAIN PROGRAM'S COMMON / CVR /)

ENDR - IEND
 (LOCAL - MAIN PROGRAM'S MAIN)

EPSI - EPSILON VALUE FROM REAL PARAMETER CARD
 (LOCAL - MAIN PROGRAM'S BRCAV2)

EPSI - VALUE OF NEW BASIC VARIABLE
 (LOCAL - MAIN PROGRAM'S PRIMAL)

ESIG - TEMPORARY STORAGE FOR EKO
 (LOCAL - MAIN PROGRAM'S BRCAV2)

EXIST(10*20) - NUMBER OF EACH TYPE RESOURCE AVAILABLE IN EACH YEAR
 (GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

EXT - TIME WHEN TIME LIMIT ON PROBLEM WILL EXPIRE
 (GLOBAL - MAIN PROGRAM'S COMMON / TIX /)

FNAME - PROBLEM TITLE (NOT USED)
 (LOCAL - REPORT GENERATOR'S SETUP)

FNAME - FILE NAME
 (LOCAL - MATRIX GENERATOR'S GENLCP)

FX1(10) - COST FUNCTION VALUES FOR LOWER BOUNDS
 (LOCAL - MAIN PROGRAM'S GETC)
 (LOCAL - MAIN PROGRAM'S NXRPN)

FX2(10) - COST FUNCTION VALUES FOR UPPER BOUND
 (LOCAL - MAIN PROGRAM'S GETC)

FX2(10) - COST FUNCTION VALUES FOR SOLUTION POINT
 (LOCAL - MAIN PROGRAM'S NXRPN)

GAMMA(101) - NOT USED
 (GLOBAL - MAIN PROGRAM'S COMMON / C /)

I - DUMMY VARIABLE
 (LOCAL - MAIN PROGRAM'S KEYEND)

I - INPUT DISK COLUMN INDEX
 (LOCAL - MAIN PROGRAM'S SETEND)
 IA1 - DISC FILE FOR MATRIX LESS GUB ROWS
 (GLOBAL - MAIN PROGRAM'S COMMON / FILES /)
 IA2 - DISC FILE FOR PACKED MATRIX LESS GUB ROWS
 (GLOBAL - MAIN PROGRAM'S COMMON / FILES /)
 IB - BASIC COLUMN INDEX
 (LOCAL - MAIN PROGRAM'S ROW)
 IB - DISC INDEX OF BASIC COLUMN FOR CURRENT ROW
 (LOCAL - MAIN PROGRAM'S KEYCH)
 IB - BOUND INDEX IN IRDS
 (LOCAL - MAIN PROGRAM'S BOUND)
 IBAS - COUNT OF BASIS VARIABLES
 (LOCAL - MAIN PROGRAM'S MAPOUT)
 IBASIS(101) - BASIC COLUMNS FOR NON-GUB ROWS
 (GLOBAL - MAIN PROGRAM'S COMMON / BASIS /)
 IRDS(100) - COLUMN INDICES OF BOUND COLUMNS
 (GLOBAL - MAIN PROGRAM'S COMMON / BOUNDS /)
 IRMAX - MAXIMUM NUMBER OF NODES WHICH MAY BE STORED ON BLIST
 (LOCAL - MAIN PROGRAM'S PRESET)
 IRND - COUNT OF BOUND VARIABLES
 (LOCAL - MAIN PROGRAM'S MAPOUT)
 IC - CURRENT COST ROW
 (GLOBAL - MAIN PROGRAM'S COMMON / I /)
 ICOST - USER'S COST ROW
 (GLOBAL - MAIN PROGRAM'S COMMON / I /)
 ICOL - TEMPORARY STORAGE FOR COLUMN INDEX
 (LOCAL - MAIN PROGRAM'S BRCAV2)
 ICX - NUMBER OF VARIABLES FOR WHICH COST SLOPES ARE TO BE DERIVED
 (LOCAL - MAIN PROGRAM'S GETC)
 ID - LOCAL ROW TYPE BEING PROCESSED
 (LOCAL - MAIN PROGRAM'S SETUP)
 ID - COLUMN NUMBER FROM MAP CARD
 (LOCAL - MAIN PROGRAM'S MAPIN)
 ID(15) - INDICES OF STORED COLUMNS
 (LOCAL - MAIN PROGRAM'S INSERT)
 IED - 'ENDTABLE'
 (LOCAL - REPORT GENERATOR'S SETUP)
 IEND - LAST YEAR OF RESOURCE EXISTANCE
 (LOCAL - REPORT GENERATOR'S INSOLN)
 IHVN (10) - POINTERS FOR INHERITED VEHICLES
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 IKEY - COUNT OF KEY VARIABLES
 (LOCAL - MAIN PROGRAM'S MAPOUT)
 IKOST - TEMPORARY STORAGE OF USER'S COST ROW
 (LOCAL - MAIN PROGRAM'S SETUP)
 IMAP - FILE FOR STARTING AND TERMINATING BASIS
 (GLOBAL - MAIN PROGRAM'S COMMON / FILES /)
 INDIC - INDICATES WHICH BRANCH (UPPER OR LOWER) IS BEING SOLVED
 (LOCAL - MAIN PROGRAM'S BRCAV2)
 IND - COLUMN NUMBER TEMPORARY STORAGE
 (LOCAL - REPORT GENERATOR'S INSOLN)

INH (10, 16) - NUMBER OF EACH TYPE RESOURCE INHERITED FROM EACH YEAR
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)
 INH(10,16) - NOT USED
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)
 INHYRS - NUMBER OF YEARS FROM WHICH VEHICLES ARE INHERITED
 (LOCAL - MATRIX GENERATOR'S GENLOC)
 INLL - COUNT OF NULL VARIABLES
 (LOCAL - MAIN PROGRAM'S MAPOUT)
 INON - TEMPORARY NUMBER OF NON GUR ROWS FOUND
 (LOCAL - MAIN PROGRAM'S SETUP)
 INPUT - FILE CONTAINING INPUT MATRIX
 (GLOBAL - MAIN PROGRAM'S COMMON / INPUT /)
 INPUTM - NUMBER OF ROWS IN MATRIX
 (GLOBAL - MAIN PROGRAM'S COMMON / INPUT /)
 INPUTN - NUMBER OF COLUMNS IN MATRIX
 (GLOBAL - MAIN PROGRAM'S COMMON / INPUT /)
 INVF - INVERT FREQUENCY
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)
 INVERS - 'INVERS'
 (LOCAL - MAIN PROGRAM'S MAPIN)
 INVR - INITIAL YEAR OF PROBLEM
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)
 IORT - USED TO FLAG UNROUNDED SOLUTION
 (GLOBAL - MAIN PROGRAM'S COMMON / CV7 /)
 IORG - ORIGIN OF BASIS INVERSE
 (LOCAL - MAIN PROGRAM'S LP)
 (LOCAL - MAIN PROGRAM'S ROW)
 (LOCAL - MAIN PROGRAM'S INVERT)
 IORG - ORIGIN OF PIVOT ROW IN B
 (LOCAL - MAIN PROGRAM'S PIVOT)
 IORG - ORIGIN OF A ROW IN BASIS INVERSE
 (LOCAL - MAIN PROGRAM'S XCHECK)
 (LOCAL - MAIN PROGRAM'S KEYCH)
 (LOCAL - MAIN PROGRAM'S FEASCH)
 IPHASE - CURRENT LP PHASE
 (GLOBAL - MAIN PROGRAM'S COMMON / I /)
 IPI - CURRENT LOCATION OF PI VECTOR IN BASIS
 (GLOBAL - MAIN PROGRAM'S COMMON / I /)
 IPT - 'PERIOD'
 (LOCAL - REPORT GENERATOR'S SETUP)
 IP(12) - STORAGE FOR INPUT PARAMETERS ON INTEGER PARAMETER CARD
 (GLOBAL - MAIN PROGRAM'S COMMON / CV1 /)
 IROWTP(101) - USER'S INPUT ROW TYPES
 (GLOBAL - MAIN PROGRAM'S COMMON / ROWTP /)
 IROWTP(100) - INDICATES ROW TYPE J ALL SET TO ZERO EXCEPT GENERALIZED UPPER
 BOUND ROWS WHICH ARE SET TO 4
 (LOCAL - MATRIX GENERATOR'S MATFILL)
 IROW - CURRENT SELECTED ROW
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)
 IROW - ROW CALLING PARAMETER, ROW OR ZERO
 (LOCAL - MAIN PROGRAM'S ROW)
 IROW - ROW TO WHICH KEY COLUMN IS SHIFTED WHEN MADE BASIC
 (LOCAL - MAIN PROGRAM'S KEYCH)
 IROW - PIVOT ROW
 (LOCAL - MAIN PROGRAM'S PIVOT)

ISTART - FIRST YEAR OF RESOURCE EXISTANCE
 (LOCAL - REPORT GENERATOR'S INSOLN)

ITABLE - TABLE NAME
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 (LOCAL - REPORT GENERATOR'S SETUP)

ITEMP - TEMP. STORAGE FOR FIRST LETTER OF RNAME
 (LOCAL - MATRIX GENERATOR'S MATFILL)

ITNINV - ITERATION OF NEXT INVERT
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)

ITRN - ITERATION COUNT
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)

ITYPE - ROW TYPE
 (LOCAL - MAIN PROGRAM'S INVERT)

ITYPE - TYPE OF STEP, 1-UNBOUNDED, 2-COLUMN TO ZERO, 3-COLUMN ROUND
 (LOCAL - MAIN PROGRAM'S ROW)
 (LOCAL - MAIN PROGRAM'S PRIMAL)

IVT - 'VEHICLE'
 (LOCAL - REPORT GENERATOR'S SETUP)

IX(100) - INDICES OF SOLUTION COLUMNS
 (GLOBAL - MAIN PROGRAM'S COMMON / IXX /)

IX(110) - COLUMNS IN BASIC SOLUTION
 (GLOBAL - MAIN PROGRAM'S COMMON / CVA /)

IXZ(110) - COLUMNS IN BEST SOLUTION
 (GLOBAL - MAIN PROGRAM'S COMMON / CVA /)

IYR(10) - INITIAL YEAR OF EACH PERIOD
 (GLOBAL - REPORT GENERATOR'S COMMON / BASICS /)

J - ABSOLUTE VALUE OF I
 (LOCAL - MAIN PROGRAM'S SETBND)

J - INPUT DISC COLUMN INDEX
 (LOCAL - MAIN PROGRAM'S BOUND)

J1 - FIRST COLUMN IN COLUMN PRINTOUT
 (LOCAL - MAIN PROGRAM'S XCHECK)

J2 - LAST COLUMN IN COLUMN PRINTOUT
 (LOCAL - MAIN PROGRAM'S XCHECK)

JAU - DISK INDEX OF AN IN-CORE COLUMN
 (LOCAL - MAIN PROGRAM'S XCHECK)

JAJ - POTENTIAL COLUMN'S IN-CORE POSITION
 (LOCAL - MAIN PROGRAM'S KEYEND)

JAK(101) - DUMMY STORAGE AREA
 (GLOBAL - MAIN PROGRAM'S COMMON / CORE /)

JAREJ(101) - SET TO L WHEN CORRESPONDING IN-CORE COLUMN REJECTED
 (GLOBAL - MAIN PROGRAM'S COMMON / CORE /)

JA(101) - IN-CORE COLUMN DISC INDICES
 (GLOBAL - MAIN PROGRAM'S COMMON / CORE /)

JBMAX - MAXIMUM NUMBER OF WORDS OF INFORMATION WHICH MAY BE STORED FOR
 EACH NODE IN BLIST
 (LOCAL - MAIN PROGRAM'S PRESET)

JCOL - CURRENT SELECTED COLUMN
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)

JCOL - USER'S COLUMN INDEX OF COLUMN PROCESSED
 (LOCAL - MAIN PROGRAM'S MAPOUT)

JCOL - CORE POSITION OF SELECTED COLUMN
 (LOCAL - MAIN PROGRAM'S COLUMN)
 (LOCAL - MAIN PROGRAM'S IO)
 (LOCAL - MAIN PROGRAM'S ROW)

JCCL - COLUMN TO BE MOVED
 (LOCAL - MAIN PROGRAM'S KEYCH)
 JCCLPK - GUR PACKET OF COLUMN BEING MOVED FROM KEY
 (LOCAL - MAIN PROGRAM'S KEYCH)
 JCCOUNT - COUNT OF COLUMNS PROCESSED
 (LOCAL - MAIN PROGRAM'S CHECK)
 JEND - ORIGIN OF VACANT WORK SPACE IN AJ
 (LOCAL - MAIN PROGRAM'S XCHECK)
 JERCH - NUMBER OF COLUMNS CHECKED IN CURRENT BATCH
 (LOCAL - MAIN PROGRAM'S CHECK)
 JKEY - CORE POSITION OF KEY FOR JCCL (IF GUR ROW)
 (LOCAL - MAIN PROGRAM'S COLUMN)
 JKEY - CANDIDATE KEY COLUMN
 (LOCAL - MAIN PROGRAM'S KEYCH)
 JNCORE - NUMBER OF COLUMNS IN CORE
 (GLOBAL - MAIN PROGRAM'S COMMON / LIMS /)
 (LOCAL - MAIN PROGRAM'S MAPOUT)
 JNSCAN - COLUMNS IN CORE + 1000 * NUMBER OF REWINDS OF FILE 1A1
 (LOCAL - MAIN PROGRAM'S STATUS)
 JNT - INDEX OF COLUMNS READ
 (LOCAL - MAIN PROGRAM'S IC)
 JNT - CURRENT COLUMN INDEX
 (LOCAL - MAIN PROGRAM'S CHECK)
 (LOCAL - MAIN PROGRAM'S INVERT)
 JNT - LAST COLUMN READ FROM DISC (IF MNTRY = 0)
 (LOCAL - MAIN PROGRAM'S STATUS)
 JNTC - INDEX OF LAST COLUMN READ FROM DISC
 (LOCAL - MAIN PROGRAM'S STATUS)
 JORG - CORE ORIGIN OF SELECTED COLUMN
 (LOCAL - MAIN PROGRAM'S ROW)
 JORG - ORIGIN OF VACANCY FOR COLUMN IN AJ
 (LOCAL - MAIN PROGRAM'S INSERT)
 JORG - ORIGIN IN AJ TO WHICH COLUMN IS READ
 (LOCAL - MAIN PROGRAM'S CHECK)
 (LOCAL - MAIN PROGRAM'S INVERT)
 JORG - ORIGIN OF A ROW IN R
 (LOCAL - MAIN PROGRAM'S KEYCH)
 (LOCAL - MAIN PROGRAM'S COLUMN)
 (LOCAL - MAIN PROGRAM'S PIVOT)
 JORG - ORIGIN OF A COLUMN IN AJ
 (LOCAL - MAIN PROGRAM'S XCHECK)
 JOUT - REJECTED COLUMN INDEX
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)
 JOUT - COLUMN TO BE REJECTED
 (LOCAL - MAIN PROGRAM'S ROW)
 JOUTPK - GUR PACKET OF COLUMN REJECTED
 (LOCAL - MAIN PROGRAM'S PRIMAL)
 JP - BASIC COLUMN FOR A ROW
 (LOCAL - MAIN PROGRAM'S PIVOT)
 JPKT - GUR PACKET OF NEW COLUMN
 (LOCAL - MAIN PROGRAM'S COLUMN)
 (LOCAL - MAIN PROGRAM'S ROW)

JPKT - GUR PACKET OF CURRENT COLUMN
 (LOCAL - MAIN PROGRAM'S FFASCH)
 (LOCAL - MAIN PROGRAM'S KEYEND)
 JPKTO - CURRENT STORED GUR KEY PACKET
 (LOCAL - MAIN PROGRAM'S COLUMN)
 JPOS - SELECTED COLUMN INDEX
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)
 JPOS - DISC INDEX OF COLUMN SELECTED
 (LOCAL - MAIN PROGRAM'S ROW)
 JPOS - DISC INDEX OF COLUMN TO BE STORED
 (LOCAL - MAIN PROGRAM'S INSERT)
 JPOSPK - GUR PACKET OF COLUMN ENTERING
 (LOCAL - MAIN PROGRAM'S PRIMAL)
 JPOSR - DISC INDEX OF COLUMN TO BE REJECTED
 (LOCAL - MAIN PROGRAM'S INSERT)
 JREFJ - ORIGIN OF REJECTED COLUMN IN AJ
 (LOCAL - MAIN PROGRAM'S INSERT)
 JRHS - USER'S INPUT RHS
 (GLOBAL - MAIN PROGRAM'S COMMON / I /)
 JSUR(10) - POINTERS FOR VEHICLE SUBSCRIPTS
 (LOCAL - MATRIX GENERATOR'S YINTERP)
 JTYPE - VARIABLE TYPE
 (LOCAL - MAIN PROGRAM'S INVERT)
 JTYPE - COLUMN TYPE
 (LOCAL - MAIN PROGRAM'S KEYEND)
 (LOCAL - MAIN PROGRAM'S CHECK)
 (LOCAL - MAIN PROGRAM'S COLUMN)
 K - NEW STATE
 (LOCAL - MAIN PROGRAM'S SETEND)
 K1 - NOT USED
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)
 K2 - NOT USED
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)
 K3 - OUTPUT CONTROL PARAMETER
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)
 K4 - XCHECK CONTROL PARAMETER
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)
 K5 - MAXIMUM LP ITERATIONS BEFORE MAPOUT
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)
 KEF - 'KEY'
 (LOCAL - MAIN PROGRAM'S MAPIN)
 KEY - INDEX OF KEY COLUMN TO BE LOCATED
 (LOCAL - MAIN PROGRAM'S IO)
 (LOCAL - MAIN PROGRAM'S KEYEND)
 KEY - SWITCH TO RETURN KEY PROCESSING TO KEY LOOP
 (LOCAL - MAIN PROGRAM'S FFASCH)
 KEYS(101)-STORAGE OF GUB KEY COLUMNS
 (GLOBAL - MAIN PROGRAM'S COMMON / BASIS /)
 KEYEND - POSITION OF KEY FOUND
 (LOCAL - MAIN PROGRAM'S KEYEND)
 KOL - COLUMN TO BE LOCATED ON EITHER FILE, OR PACKET NUMBER OF DESIRED KEY
 (LOCAL - MAIN PROGRAM'S IO)
 KOL1 - LAST COLUMN READ, ON FILE 1A1
 (LOCAL - MAIN PROGRAM'S IO)

KOL2 - LAST COLUMN READ ON FILE IAP
 (LOCAL - MAIN PROGRAMS IO)
 KOR6 - ORIGIN IN AJ TO WHICH KEY COLUMN IS READ
 (LOCAL - MAIN PROGRAMS INVERT)
 KORC - ORIGIN OF KEY COLUMN FOR PACKET JKPT
 (LOCAL - MAIN PROGRAMS POW)
 KOR8 - ORIGIN OF KEY COLUMN IN AJ
 (LOCAL - MAIN PROGRAMS CHECK)
 (LOCAL - MAIN PROGRAMS COLUMN)
 L - NUMBER OF SUB ROWS
 (GLOBAL - MAIN PROGRAMS COMMON / I /)
 LAST - LAST YEAR OF PROBLEM
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)
 LEND - LENGTH OF PERIOD UNDER CONSIDERATION
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)
 LIFEER - TEMP. STORAGE FOR REMAINING USEFUL LIFE OF A VEHICLE
 (LOCAL - MATRIX GENERATOR'S GENLOC)
 LSTMAX - MAXIMUM LENGTH WHICH THE BRANCHING LIST HAS ACHIEVED
 (GLOBAL - MAIN PROGRAMS COMMON / CV5 /)
 LSTRE(25) - GIVES LOCATIONS OF STORAGE AREAS ON THE BRANCHING LIST WHICH
 ARE VACANT
 (LOCAL - MAIN PROGRAMS BRCAV2)
 LY - LAST YEAR OF A PROBLEM
 (LOCAL - MATRIX GENERATOR'S GENLOC)
 LYP(10) - LAST YEAR OF EACH PERIOD
 (GLOBAL - REPORT GENERATOR'S COMMON / BASICS /)
 M - NUMBER OF ACTIVE INTERNAL ROWS
 (GLOBAL - MAIN PROGRAMS COMMON / I /)
 M - NUMBER OF ROWS IN MATRIX
 (GLOBAL - MAIN PROGRAMS COMMON / CV3 /)
 M - VECTOR DIMENSION
 (LOCAL - MAIN PROGRAMS DCT)
 MAPRND(100) - ROUND COLUMN INDICES
 (LOCAL - MAIN PROGRAMS MAPOUT)
 MAPBAS(100) - BASIC COLUMN INDICES
 (LOCAL - MAIN PROGRAMS MAPOUT)
 MAPKEY(1000) - KEY COLUMN INDICES
 (LOCAL - MAIN PROGRAMS MAPOUT)
 MAPNULL(10) - NULL COLUMN INDICES
 (LOCAL - MAIN PROGRAMS MAPOUT)
 MAXL - TEMP. STORAGE FOR VEHICLE LIFE
 (LOCAL - MATRIX GENERATOR'S GENLOC)
 MAXTRY - MAXIMUM NUMBER OF IN-CORE ITERATIONS
 (GLOBAL - MAIN PROGRAMS COMMON / LIMS /)
 MC - LAST LOGICAL COLUMN
 (GLOBAL - MAIN PROGRAMS COMMON / I /)
 MCOL - NUMBER OF COLUMNS IN MATRIX
 (LOCAL - MATRIX GENERATOR'S GENLOC)
 MW - NUMBER OF ELEMENTS IN BASIS INVERSE
 (LOCAL - MAIN PROGRAMS MAPOUT)
 (LOCAL - MAIN PROGRAMS MAPIN)
 MNC - THE NEGATIVE OF NCF
 (LOCAL - MAIN PROGRAMS BRCAV2)

MNTY - NUMBER OF IN-CORE ITERATIONS
 (LOCAL - MAIN PROGRAM'S STATUS)

MNX - THE NEGATIVE OF N
 (LOCAL - MAIN PROGRAM'S BRCAV2)

MPI - M PLUS I
 (LOCAL - MAIN PROGRAM'S MAPOUT)

MPK - ROW OF COLUMN WHICH WAS KEY
 (LOCAL - MAIN PROGRAM'S KEYCH)

MPLUS - NUMBER OF ROWS IN THE MATRIX INCLUDING THE COST ROW (M+1)
 (GLOBAL - MAIN PROGRAM'S COMMON / CV3 /)

MPL - M PLUS L
 (GLOBAL - MAIN PROGRAM'S COMMON / I /)

MROWS - USER'S NUMBER OF ROWS
 (LOCAL - MAIN PROGRAM'S LP)

N - NUMBER OF COLUMNS IN MATRIX
 (GLOBAL - MAIN PROGRAM'S COMMON / CV3 /)

N - NUMBER OF COLUMNS CURRENTLY SAVED
 (LOCAL - MAIN PROGRAM'S INSERT)

NAAM - NOT USED
 (LOCAL - MAIN PROGRAM'S IO)

NAMFN (10) - POINTERS FOR NUMBERING RESOURCES
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)

NAMFN(10) - NOT USED
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)

NAMES(5) - COLUMN INDICES FROM MAP CARD
 (LOCAL - MAIN PROGRAM'S MAPIN)

NAMES (10) - TEMPORARY POINTERS
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NAME(600) - STATE OF EACH VARIABLE OR COLUMN
 (GLOBAL - MAIN PROGRAM'S COMMON / NAMES /)

NAME - COLUMN NAME, OR POSITION IN CORE TO WHICH COLUMN IS READ
 (LOCAL - MAIN PROGRAM'S IO)

NB - NUMBER OF BASIC VARIABLES IN A PACKET
 (LOCAL - MAIN PROGRAM'S FEASCH)

NRCH - NUMBER OF COLUMNS IN BATCH
 (LOCAL - MAIN PROGRAM'S CHECK)

NRDS - NUMBER OF ROUNDS
 (GLOBAL - MAIN PROGRAM'S COMMON / ROUNDS /)

NRVPKT - NUMBER OF BASIS VARIABLE IN SELECTED GUR ROW
 (LOCAL - MAIN PROGRAM'S PRIMAL)

NC - NUMBER OF BASIS CARDS TO BE READ FROM INPUT
 (LOCAL - MAIN PROGRAM'S READIN)

NCF - NUMBER OF NON-LINEAR VARIABLES
 (GLOBAL - MAIN PROGRAM'S COMMON / CV3 /)

NCF1 - NCF
 (LOCAL - MAIN PROGRAM'S BRCAV2)

NCF4 - NCRA + 3 * NCF
 (LOCAL - MAIN PROGRAM'S BRCAV2)

NCHGS - USER'S NUMBER OF BOUND COLUMNS
 (LOCAL - MAIN PROGRAM'S LP)

NCOLS - NUMBER OF COLUMNS READ FROM DISC FILE IAI
 (LOCAL - MAIN PROGRAM'S STATUS)

NCOLS - USER'S NUMBER OF COLUMNS
 (LOCAL - MAIN PROGRAM'S LP)

NCORE - NUMBER OF COLUMNS IN CORE
 (LOCAL - MAIN PROGRAM'S COLUMN)

NCRMAX - MAXIMUM NUMBER OF COLUMNS WHICH FIT IN CORE
 (GLOBAL - MAIN PROGRAM'S COMMON / LIMS /)

NDEGLM - MAXIMUM REJECTION DUE TO DEGENERACY
 (LOCAL - MAIN PROGRAM'S PRIMAL)

NDEG - NUMBER OF DEGENERACY REJECTIONS
 (LOCAL - MAIN PROGRAM'S PRIMAL)

NDUJS - NUMBER OF NEGATIVE DU'S
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)

NDUJS - NUMBER OF NEGATIVE DU'S FROM DISK READ
 (LOCAL - MAIN PROGRAM'S COLUMN)

NDV(10) - INDICES OF NON-LINEAR VARIABLES
 (LOCAL - MAIN PROGRAM'S NIXRPN)

NEM1 - NUMBER OF ELEMENTS IN AN ARRAY MINUS ONE
 (LOCAL - MAIN PROGRAM'S GETASO)

NEWXZ - FLAG WHEN NEW BEST SOLUTION FOUND AND SHOULD BE OUTPUT
 (GLOBAL - MAIN PROGRAM'S COMMON / CV7 /)

NEWROW - ROW FOR COLUMN CHANGING FROM KEY TO BASIC
 (LOCAL - MAIN PROGRAM'S PRIMAL)

NFI - SIGNIFIES FEASIBLE SOLUTION WHEN SET EQUAL TO 1
 (GLOBAL - MAIN PROGRAM'S COMMON / CV7 /)

NFRCH - NUMBER OF COLUMNS RETAINED FROM BATCH
 (LOCAL - MAIN PROGRAM'S CHECK)

NFREE - NUMBER OF GAPS (EMPTY LOCATION BETWEEN TWO FILLED LOCATIONS) IN
 THE PLIST
 (LOCAL - MAIN PROGRAM'S BRCAV2)

NFX - THE NEGATIVE OF NCF
 (LOCAL - MAIN PROGRAM'S NIXRPN)

NINHP - NUMBER OF INHERITED PERIODS
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NIV - NUMBER OF INHERITED VEHICLE TYPES
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NJOUT - NAME CODE OF COLUMN TO BE REJECTED
 (LOCAL - MAIN PROGRAM'S STATUS)

NL (10) - TEMP. STORAGE USED IN FORMATTING OUTPUT
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NMN - INDEX OF THE LOWEST ROUND ON THE PLIST, OR N-1, DEPENDING ON WHERE
 IT IS USED
 (LOCAL - MAIN PROGRAM'S BRCAV2)

NN (10) TEMP. STORAGE USED IN FORMATTING OUTPUT
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NORCL - NUMBER OF NODES ON LIST
 (GLOBAL - MAIN PROGRAM'S COMMON / CVH /)

NOL - INDEX FOR STORAGE ON PLIST
 (LOCAL - MAIN PROGRAM'S BRCAV2)

NOP - NODE NUMBER
 (GLOBAL - MAIN PROGRAM'S COMMON / CV7 /)

NORS - NODES SOLVED
 (GLOBAL - MAIN PROGRAM'S COMMON / CV7 /)

NORA - NUMBER OF ROWS IN THE MATRIX INCLUDING THE OBJECTIVE FUNCTION
 (LOCAL - MAIN PROGRAM'S PROAV2)
 (LOCAL - MAIN PROGRAM'S PRESET)

NOTE(4) - 40 CHARACTER COMMENT
 (LOCAL - MAIN PROGRAM'S STATUS)

NPI - NUMBER OF SUPPERIODS
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)

NPRCH - NUMBER OF COLUMNS TO BE SAVED FROM BATCH
 (LOCAL - MAIN PROGRAM'S INSERT)

NPERYR (10, 3) - FIRST AND LAST YEAR OF PERIOD AND NUMBER OF TASKS IN PERIOD
 (GLOBAL - MATRIX GENERATOR'S COMMON / PRUSIG /)

NPHASE - STORES LP PHASE CODE
 (GLOBAL - MAIN PROGRAM'S COMMON / CV /)

NPIC - NUMBER OF PRIMAL INFEASIBILITIES
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)

NPP - NUMBER OF PERIODS
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NPTASK (10, 9) - ID NUMBER OF EACH TASK IN PERIOD
 (GLOBAL - MATRIX GENERATOR'S COMMON / PRUSIG /)

NPT - NUMBER OF PERIOD TABLES READ
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 (LOCAL - REPORT GENERATOR'S SETUP)

NRD - NUMBER OF VEHICLES HAVING R AND D
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NREJ - NUMBER OF REJECTED IN-CORE COLUMNS
 (GLOBAL - MAIN PROGRAM'S COMMON / STATE /)

NROW - NUMBER OF ROWS IN MATRIX
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NV - NUMBER OF RESOURCE TYPES
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)

NSCAN - NUMBER OF DISC READS
 (GLOBAL - MAIN PROGRAM'S COMMON / LIMS /)

NT - NUMBER OF TASKS
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NT - TOTAL NUMBER OF COLUMNS (MC+INPUTN)
 (GLOBAL - MAIN PROGRAM'S COMMON / I /)

NT - NUMBER OF TASKS (NOT USED)
 (LOCAL - REPORT GENERATOR'S SETUP)

NTR - NUMBER OF TASK TABLES READ
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NTRY - NUMBER OF IN-CORE ITERATIONS
 (GLOBAL - MAIN PROGRAM'S COMMON / LIMS /)

NTSK (9) - NUMBER OF ALTERNATIVES IN TASK
 (GLOBAL - MATRIX GENERATOR'S COMMON / TSKSIG /)

NULL - 'NULL'
 (LOCAL - MAIN PROGRAM'S MAPIN)

NULL - COLUMN STATE
 (LOCAL - MAIN PROGRAM'S COLUMN)

NV - NUMBER OF VEHICLE TYPES
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NVEHU (10) - INDICATES IF VEHICLE USED IN PERIOD
 (LOCAL - MATRIX GENERATOR'S GENLCP)

NVR - NUMBER OF VEHICLE TABLES READ
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 (LOCAL - REPORT GENERATOR'S SETUP)

NWAJ - STORAGE DIMENSION OF THE ARRAY AJ
 (LOCAL - MAIN PROGRAM'S LP)

NYBX - INDEX OF BRANCHING VARIABLE
 (GLOBAL - MAIN PROGRAM'S COMMON / CVH /)

NXQL(25) - INDEX OF BRANCHING VARIABLE FOR EACH NODE
 (GLOBAL - MAIN PROGRAM'S COMMON / CVY /)

NYR - TEMPORARY STORAGE FOR NEXT BRANCHING VARIABLE
 (LOCAL - MAIN PROGRAM'S BRCAV2)

NYR - TEMP. STORAGE FOR LAST YEAR OF PERIOD
 (LOCAL - MATRIX GENERATOR'S GENLCP)

OANDM(20) - OPERATING COST FOR EACH YEAR
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

ONE - '01'
 (LOCAL - REPORT GENERATOR'S CINFO)

ONE - 1.0
 (LOCAL - MATRIX GENERATOR'S GENLCP)

ONEM - -1.0
 (LOCAL - MATRIX GENERATOR'S GENLCP)

PACK(100) - TEMPORARY STORAGE OF PACKED COLUMN
 (LOCAL - MAIN PROGRAM'S IO)

PER(10) - POINTERS FOR TWO DIGIT, ALPHANUMERIC CODE FOR PERIODS
 (GLOBAL - REPORT GENERATOR'S COMMON / BASICS /)

PERIOD - 'PERIOD'
 (LOCAL - REPORT GENERATOR'S CINFO)
 (LOCAL - REPORT GENERATOR'S PINFO)

PH1 AND PH2 - TEMPORARY STORAGE OF VALUES FROM GETPHI
 (LOCAL - MAIN PROGRAM'S BRCAV2)

PHIT - COST OF A NON-LINEAR SOLUTION
 (GLOBAL - MAIN PROGRAM'S COMMON / CV3 /)

PIKEY - DJ VALUE FOR CURRENT KEY JKPT
 (LOCAL - MAIN PROGRAM'S COLUMN)

PIKEY - DJ FOR CURRENT KEY AT KORG
 (LOCAL - MAIN PROGRAM'S CHECK)

PIVTOL - PIVOT TOLERANCE
 (GLOBAL - MAIN PROGRAM'S COMMON / TOLS /)

PIV - PIVOT USED
 (LOCAL - MAIN PROGRAM'S PIVOT)

PKT1 - TEMPORARY COUNT OF GUR ROW PACKET COLUMN
 (LOCAL - MAIN PROGRAM'S SETUP)

PKT - ACTUAL GUR ROW COLUMN BEING PROCESSED
 (LOCAL - MAIN PROGRAM'S SETUP)

PKT - STORAGE OF COLUMN PACKET
 (LOCAL - MAIN PROGRAM'S MARIN)

PKT - GUR PACKET NUMBER OF COLUMN BEING PROCESSED
 (LOCAL - MAIN PROGRAM'S INVERT)

PKT - PACKET OF NEW COLUMN, JNT
 (LOCAL - MAIN PROGRAM'S CHECK)

PKT - GUR PACKET OF DECIDED KEY
 (LOCAL - MAIN PROGRAM'S KEYEND)

PKT2 - GUR PACKET NUMBER OF COLUMN IN AJ(KORG)
 PKT2 - PACKET OF CURRENT KEY
 (LOCAL - MAIN PROGRAM'S CHECK)
 (LOCAL - MAIN PROGRAM'S INVERT)

PMIN - VALUE OF LOWEST BOUND ON PLIST
 (LOCAL - MAIN PROGRAM'S PRCV2)

PERTOL - PRIMAL ERROR TOLERANCE
 (GLOBAL - MAIN PROGRAM'S COMMON / IOLS /)

PRMLER - PRIMAL ERROR, UNUSED
 (GLOBAL - MAIN PROGRAM'S COMMON / MOVES /)

PROC(20) - PROCUREMENT FUNDS SPENT DURING EACH PERIOD
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

PROT(20) - PROCUREMENT FUNDS AVAILABLE DURING EACH PERIOD
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

PSIGL(25) - LOWER BOUND ASSOCIATED WITH EACH NODE ON LIST
 (GLOBAL - MAIN PROGRAM'S COMMON / CV9 /)

PTASK (10, 0) - MULTIPLICATIVE FACTOR FOR ALL VALUES IN ASSOCIATED TASK FOR
 EACH PERIOD
 (GLOBAL - MATRIX GENERATOR'S COMMON / PRODIG /)

PURCH(10,20) - NUMBER OF EACH TYPE RESOURCE PURCHASED IN EACH YEAR
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

R - RATE OF INCREASE IN OPERATING COST
 (LOCAL - MATRIX GENERATOR'S YRCOST)

R - 'RHS'
 (LOCAL - MATRIX GENERATOR'S MATFILL)

R1 - PORTION OF OPERATING COST REFUNDED FOR MONTH BALLING RESOURCE
 (LOCAL - MATRIX GENERATOR'S YRCOST)

RDTOT - TOTAL R AND D EXPENDITURES
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)

RHS(100) - STORES USER'S CURRENT RIGHT HAND SIDE
 (GLOBAL - MAIN PROGRAM'S COMMON / RHS /)

RNAME(120) - ROW NAMES
 (LOCAL - MATRIX GENERATOR'S MATFILL)

ROWS - 'ROWC'
 (LOCAL - MAIN PROGRAM'S MAPIN)

RP(12) - STORAGE FOR REAL PARAMETERS, FIRST FOUR LOCATIONS ARE FOR INPUT
 FROM REAL PARAMETER CARD, REST ARE TEMPORARY STORAGE
 (GLOBAL - MAIN PROGRAM'S COMMON / CV1 /)

RTEMP - TEMP. STORAGE FOR ROW NAMES
 (LOCAL - MATRIX GENERATOR'S MATFILL)

RVAL(100) - VECTOR OF VALUES IN EACH ROW FOR A SPECIFIC COLUMN
 (LOCAL - MATRIX GENERATOR'S MATFILL)

S - 'S'
 (LOCAL - REPORT GENERATOR'S INSOLN)

SALF(20) - SALVAGE OR TRUNCATION VALUE FOR EACH YEAR
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

SALV(10,20) - NUMBER OF EACH TYPE RESOURCE DISPOSED OF AT END OF EACH YEAR
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

SAVE(20) - SAVINGS FROM RESOURCE STORAGE FOR EACH YEAR
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

SECS - ACTUAL CPU CLOCK TIME
 (LOCAL - MAIN PROGRAM'S TIMEC)

SIGMA(100,4) - STORES INFORMATION WHICH DEFINES THE CURRENT NODE
 (GLOBAL - MAIN PROGRAM'S COMMON / CV5 /)

STEP - STEP TO CURRENT ROW
 (LOCAL - MAIN PROGRAM'S ROW)

STOR(10,20) - NUMBER OF EACH TYPE RESOURCE STORED IN EACH YEAR
 (GLOBAL - REPORT GENERATOR'S COMMON / OUIS /)

SUM - VALUE OF VARIABLE BEFORE FEASIBILITY ADJUSTMENT
 (LOCAL - MAIN PROGRAM'S FEASCH)
 SUM - TEMPORARY STORAGE
 (LOCAL - MAIN PROGRAM'S KEYCH)
 SUM - DOUBLE PRECISION ACCUMULATOR
 (LOCAL - MAIN PROGRAM'S DCI)
 SUM - TOTAL COST FOR A PERIOD
 (LOCAL - REPORT GENERATOR'S CINEQ)
 SUMIE - SUM OF INFEASIBILITIES
 (LOCAL - MAIN PROGRAM'S FEASCH)
 SUMT - TOTAL COST FOR ALL PERIODS
 (LOCAL - REPORT GENERATOR'S CINEQ)
 SY - START YEAR OF PROBLEM
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 T(100,10) - STORAGE FOR COLUMNS OF MATRIX ASSOCIATED WITH NON-LINEAR
 VARIABLES
 (GLOBAL - MAIN PROGRAM'S COMMON / CV2 /)
 TCOST - TEMP STORAGE FOR TOTAL PROCUREMENT
 (LOCAL - REPORT GENERATOR'S CINEQ)
 TEMP - LOCATION USED WHILE SWAPPING CONTENTS OF TWO LOCATIONS IN AN ARRAY
 (LOCAL - MAIN PROGRAM'S GETASQ)
 TEMP1 - TEMPORARY STORAGE LOCATIONS FOR ALPHANUMERIC OUTPUT
 (LOCAL - REPORT GENERATOR'S PINEQ)
 TEMP2 - TEMPORARY STORAGE LOCATIONS FOR ALPHANUMERIC OUTPUT
 (LOCAL - REPORT GENERATOR'S PINEQ)
 TEMP(4) - TEMP STORAGE FOR COLUMN NAMES
 (LOCAL - REPORT GENERATOR'S SETUP)
 THETA - STEP CHOSEN BY ROW, ADJUSTED IN PRIMAL
 (GLOBAL - MAIN PROGRAM'S COMMON / MOVES /)
 THETA - BEST FEASIBLE STEP
 (LOCAL - MAIN PROGRAM'S ROW)
 TITLE(4) - ALPHANUMERIC TITLE OF PROBLEM
 (LOCAL - MAIN PROGRAM'S BRCAV2)
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)
 TMAX - MAXIMUM TIME BEFORE MAPOUT
 (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS /)
 TMO - TIME SET WAS CALLED
 (GLOBAL - MAIN PROGRAM'S COMMON / IMX /)
 TMP(10) - TEMPORARY STORAGE
 (GLOBAL - MAIN PROGRAM'S COMMON / CV1 /)
 TOT - NUMBER OF SUPERIODS PLUS 1
 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)
 TOTAL - TOTAL
 (LOCAL - REPORT GENERATOR'S CINEQ)
 (LOCAL - REPORT GENERATOR'S PINEQ)
 TPROC - CORRECTION FACTOR FOR PROCUREMENT
 (LOCAL - REPORT GENERATOR'S CINEQ)
 TRIG - TEMPORARY STORAGE ASSOCIATED WITH FKO
 (GLOBAL - MAIN PROGRAM'S COMMON / CV5 /)
 TRIG(130) - TEMPORARY STORAGE
 (LOCAL - MAIN PROGRAM'S BRCAV2)
 TYPE1 - FIRST WORD ON MAP CARD
 (LOCAL - MAIN PROGRAM'S MAPIN)

TYPE2 - SECOND WORD ON MAP CARD
 (LOCAL - MAIN PROGRAM'S MAPIN)

U (7, 288, 0) - ARRAY OF TASK ALTERNATIVES
 (GLOBAL - MATRIX GENERATOR'S COMMON / TASKSIG /)

UP (10) - CALCULATED UPPER BOUNDS ON RESOURCES
 (LOCAL - MATRIX GENERATOR'S GENLCP)

URK - UPPER BOUND ON BRANCHING VARIABLE
 (LOCAL - MAIN PROGRAM'S BRCAV2)

URK2 - DIFFERENCE BETWEEN UPPER BOUND AND VALUE FOR BRANCHING VARIABLE
 (LOCAL - MAIN PROGRAM'S BRCAV2)

ULO(10) - SET OF UPPER BOUNDS ON NON-LINEAR VARIABLES
 (GLOBAL - MAIN PROGRAM'S COMMON / CV2 /)

ULT(10) - TEMPORARY STORAGE FOR ULO
 (LOCAL - MAIN PROGRAM'S BRCAV2)

UMAX - TEMP. STORAGE FOR GREATEST QUANTITY OF A SPECIFIC VEHICLE WHICH MIGHT
 BE USED IN A TASK
 (LOCAL - MATRIX GENERATOR'S GENLCP)

US - TEMPORARY STORAGE FOR USP
 (LOCAL - MAIN PROGRAM'S BRCAV2)

USM = UZ/(1-F)
 (GLOBAL - MAIN PROGRAM'S COMMON / CV3 /)

USP = UZ/(1+F)
 (GLOBAL - MAIN PROGRAM'S COMMON / CV3 /)

UZ - COST OF BEST NON-LINEAR SOLUTION
 (GLOBAL - MAIN PROGRAM'S COMMON / CV3 /)

VAL - COLUMN VALUE TEMPORARY STORAGE
 (LOCAL - REPORT GENERATOR'S INSOLN)

VAL - TEMP. STORAGE FOR VALUE OF SPECIFIC ROW AND COLUMN
 (LOCAL - MATRIX GENERATOR'S MATFILL)

VCOST(10,5) - THE FIVE COSTS ASSOCIATED WITH EACH RESOURCE ARE STORED IN
 THIS ARRAY - IN ORDER, THEY ARE SALVAGE AND TRUNCATION, OPERATING, R AND D,
 RETENTION RATE, AND PROCUREMENT.
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)

VLIFE (10) - MAXIMUM LIFE OF RESOURCE (VEHICLE)
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)

VMIN - TEMP. STORAGE FOR MINIMUM QUANTITY OF VEHICLES WHICH CAN BE USED FOR
 TASK
 (LOCAL - MATRIX GENERATOR'S YINIERP)

VNAME(10) - STORES RESOURCE NAMES
 (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG /)
 (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG /)

W - 'W'
 (LOCAL - REPORT GENERATOR'S INSOLN)

X - INPUT VECTOR
 (LOCAL - MAIN PROGRAM'S DOT)

X - 'X'
 (LOCAL - REPORT GENERATOR'S INSOLN)

X - ELAPSED CPU SECONDS
 (LOCAL - MAIN PROGRAM'S STATUS)

XCON(10) - STORES VALUES FOUND IN X WHICH ARE ASSOCIATED WITH THE
 NON-LINEAR VARIABLES
 (GLOBAL - MAIN PROGRAM'S COMMON / CV4 /)

XK - VALUE OF BRANCHING VARIABLE
 (GLOBAL - MAIN PROGRAM'S COMMON / CV8 /)
 XNXPL(25) - VALUE OF BRANCHING VARIABLE FOR EACH NODE
 (GLOBAL - MAIN PROGRAM'S COMMON / CV9 /)
 XT(10) - SOLUTION VALUES FOR NON-LINEAR VARIABLE
 (LOCAL - MAIN PROGRAM'S NABRN)
 XX - ELAPSED TIME ON PROBLEM
 (LOCAL - MAIN PROGRAM'S TIMEC)
 X(100) - VALUES OF SOLUTION COLUMNS
 (GLOBAL - MAIN PROGRAM'S COMMON / AA /)
 X(110) - VALUES ASSOCIATED WITH COLUMNS IN IX
 (GLOBAL - MAIN PROGRAM'S COMMON / CVA /)
 XZ(110) - VALUES ASSOCIATED WITH COLUMNS IN IXZ
 (GLOBAL - MAIN PROGRAM'S COMMON / CVA /)
 Y - INPUT VECTOR
 (LOCAL - MAIN PROGRAM'S DOT)
 YAVL (10) - YEAR RESOURCE FIRST AVAILABLE
 (GLOBAL - MATRIX GENERATOR'S COMMON / ALTSTG /)
 YEARS (21) - STORES INHERITED YEARS
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 YRINT (20) - SCALE FACTOR FOR ALL TASKS IN PERIOD
 (LOCAL - MATRIX GENERATOR'S GENLCP)
 YT(10) - DIFFERENCES BETWEEN SOLUTION POINT AND LOWER BOUNDS
 (LOCAL - MAIN PROGRAM'S NXBRN)
 Z - PARAMETER USED TO PACK INDEX OF COEFFICIENTS
 (LOCAL - MAIN PROGRAM'S IC)
 69H8 8 7==1977 H91879IG9= 97=19H
 (GLOBAL - MAIN PROGRAM'S COMMON / TOLS /)
 ZS - PARAMETER USED TO PACK COEFFICIENTS
 (LOCAL - MAIN PROGRAM'S IC)

REFERENCES FOR VOLUME I

1. J. C. Hetrick, "Mathematical Models in Capital Budgeting," Chapter 7. New Decision Making Tools for Managers; edited by E. C. Bursk and J. F. Chapman, (Cambridge, Harvard University Press, 1963) p. 186.
2. J. E. Falk and R. M. Soland, "An Algorithm for Separable Non-Convex Programming Problems," Management Science, Vol. 15, No. 9, May 1969.
3. E. M. L., Beale, "Advanced Algorithmic Features for General Mathematical Programming Systems," Integer and Nonlinear Programming, edited by J. Abadie, North-Holland Publishing Company, 1970.

REFERENCES FOR VOLUME II

4. R. New, "Optimal Planning Over Time — OPT," Journal of Systems Management, March 1972.
5. R. M. Soland, "An Algorithm For Separable Non-Convex Programming Problems II: Non-Convex Constraints," Management Science, Volume 17, No. 11, July 1971.