United States Naval Postgraduate School

THESIS

CAI-BASIC
A Program to Teach the Programming Language "BASIC"

by

Thomas Anthony Barry

Thesis Advisor A. B. Roberts

September 1971

Approved for public release; distribution unlimited.
Best Available Copy
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer aided instruction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer assisted instruction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmed instruction</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This paper presents a computer aided instruction program that fulfills the objectives of teaching a simple programming language, interpreting student responses, and executing and editing student programs. The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM-2741 terminals while running under the CP-67/CMS time sharing system on the U.S. Naval Postgraduate School's IBM-360/67 computer system. The instructional phase of CAI-BASIC presents the fundamentals of "BASIC," a simple user oriented language, in seven lessons. During the instructional sessions the student is presented material and, based on his response to questions, he is routed to the next sequence of instructions. The execution phase of CAI-BASIC allows execution of "BASIC" programs, and has an optional debug feature that provides a trace of program variables to aid the student in finding programming errors. In the event of programming errors the user may enter an edit mode to correct mistakes in his program.
CAI-BASIC

A Program to Teach the

Programming Language "BASIC"

by

Thomas Anthony Barry
Lieutenant, United States Navy
B.S., United States Naval Academy, 1965

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1971

Author

Thomas A. Barry

Approved by:

Thesis Advisor

Chairman, Department of Mathematics

Academic Dean
ABSTRACT

This paper presents a computer aided instruction program that fulfills the objectives of teaching a simple programming language, interpreting student responses, and executing and editing student programs. The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM-2741 terminals while running under the CP-67/CMS time sharing system on the U.S. Naval Postgraduate School's IBM-360/67 computer system. The instructional phase of CAI-BASIC presents the fundamentals of "BASIC," a simple user oriented language, in seven lessons. During the instructional sessions the student is presented material and, based on his response to questions, he is routed to the next sequence of instructions. The execution phase of CAI-BASIC allows execution of "BASIC" programs, and has an optional debug feature that provides a trace of program variables to aid the student in finding programming errors. In the event of programming errors the user may enter an edit mode to correct mistakes in his program.
TABLE OF CONTENTS

I. INTRODUCTION .................................................. 4
   A. DEFINITION OF CAI ........................................ 4
   B. CAI DEVELOPMENTS ....................................... 4
   C. OBJECTIVES .............................................. 6
II. DESCRIPTION OF CAI-BASIC ................................. 7
   A. INSTRUCTION PHASE ...................................... 7
   B. EXECUTION PHASE ........................................ 8
III. LIMITATIONS AND EXTENSIONS ............................ 10
IV. CONCLUSION .................................................. 12
APPENDIX A USING CAI-BASIC ............................... 13
APPENDIX B CAI-BASIC PROGRAM ............................ 16
APPENDIX C TERMINAL SESSION WITH CAI-BASIC ............ 70
BIBLIOGRAPHY .................................................. 119
INITIAL DISTRIBUTION LIST ................................. 120
FORM DD 1473 .................................................. 121
I. INTRODUCTION

During the last few years many sophisticated projects in Computer Assisted Instruction (CAI) have evolved. The potential of this area has captured the imagination of researchers and the public at large. Yet the U.S. COMMISSION ON INSTRUCTIONAL TECHNOLOGY (ref. 9) found that the status of instructional technology in American Education was low in both quantity and quality. There are many reasons that CAI has not been accepted with any marked degree of enthusiasm by educators in general. The major reasons are the exorbitant costs of individualized instruction and illusions over just what CAI is.

A. DEFINITION OF CAI

Computer Assisted Instruction is machine augmented instruction that differs from Programmed Instruction or simple page turning machines in that the computer system has logi. and memory capabilities to assist in the instructional process. The main advantage in CAI is that it can individualize education. A student can proceed at his own pace, spending more time on difficult material, and quickly covering material which comes easy to him.

B. CAI DEVELOPMENTS

A. G. OETTINGER, in his book RUN, COMPUTER, RUN (ref. 6) states that educators and the computer industry are equally to blame for the failure of CAI to realize its potential. Instructional technology has been force fed,
oversold, and prematurely applied; and as a result the educators are wary of false promises. OETTINGER feels that colleges and universities will be an effective proving ground for future educational technology. However, at the present time, of the thousands of colleges and universities in the nation only twenty-five are considered major CAI centers.¹

A common problem confronting the nation's colleges and universities is that of providing instruction in elementary computer programming.² The popularity of the computer science field, and the range of application of computers in everyday life means that the demand for people who know how to communicate with computers is growing rapidly. The solution to the problem is the computer itself. "Whatever the state of CAI with respect to other subjects, the computer is the ideal instrument for teaching its own use."³

W. R. SMITH and J. L. YOUNG, graduate students at the Naval Postgraduate School, presented a proposal for the use of computer Assisted Instruction at the Naval Postgraduate School (ref. 10). As this introduction has done, they created an awareness for the need of CAI, and they made specific recommendations to provide a balanced program for the entire community to emulate. In particular they recommended commencing student projects to probe the potentials of CAI


3. IBID., p. 141-142.
It was with these ideas about using the computer to teach computer programming and providing a CAI base at the Naval Postgraduate School that CAI-BASIC was undertaken.

C. OBJECTIVES

The objective of CAI-BASIC was to develop a CAI program to teach a computer programming language on the Naval Postgraduate School's time-sharing terminals. The programming language chosen was "BASIC" (Beginners All-purpose Symbolic Instruction Code), a computer language developed at Dartmouth College in the mid-1960's. "BASIC" was selected because of its simplicity yet fairly large range of application for the user who wants to take advantage of computer processing. It was felt that "BASIC" would provide the average non-computer oriented graduate student with satisfactory results in a minimal amount of study time.

With this objective in mind three sub-goals were determined to be essential for the CAI-BASIC project: to be able to interpret student responses, to be able to execute "BASIC" programs in order to give the student sufficient programming practice, and to be able to allow editing of "BASIC" programs.
II. DESCRIPTION OF CAI-BASIC

The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM-2741 terminals while running under CP-67/CMS Time Sharing System on the Naval Postgraduate School's IBM-360/67 Computer System. The "BASIC" language implemented in CAI-BASIC is standard non-extended "BASIC" (refs. 3, 5, 10).

The CAI-BASIC program consists of two phases; an instruction phase, and an execution and editing phase. A detailed discussion of the use of CAI-BASIC is presented in APPENDIX A.

A. INSTRUCTION PHASE

The instruction phase presents fundamental concepts common to most programming languages: identifiers, variables, iteration, branching, subroutines, built in functions, and recursion. The instruction set consists of seven lesson modules which, for reasons of coherency, are dependent upon each other. The underlying philosophy in preparing the lesson modules was that an important idea should be presented only after a need for it has clearly been established. 4

Each lesson consists of instruction sequences followed by questions, evaluation of student responses, routing to the next sequence of instruction,

4. FENICHEL, OP. CIT., p. 142.
and a summary of the lesson followed by questions and/or problems to program. The student is allowed to progress through the lessons at his own speed, and allowed to review lessons or execute programs at the completion of each lesson.

Student responses are interpreted by one of two methods depending upon the type of question asked. When the question is of the true-false, multiple choice or give the answer type then the response is compared with a pre-stored result. When the student is asked to reply with a "BASIC" statement or to write a "BASIC" program, the CAI-BASIC compiler interprets the "BASIC" statements for syntactic correctness. When the student writes a program he can compare his answer with a pre-stored result, and if the answer is wrong the student is shown a solution program.

B. EXECUTION PHASE

The heart of the execution phase is the "BASIC" compiler that is used at the Naval Postgraduate School's Computer Center to execute "BASIC" programs in a batch mode. This compiler, written in FORTRAN IV, level G, was modified from a batch processing compiler to a line-by-line interpreter and incorporated into the CAI-BASIC program, and called the CAI-BASIC COMPILER.

The CAI-BASIC COMPILER has an added feature to aid students in debugging programs on the terminal. The added feature is called a "DEBUG" function and it produces a list of numeric and alpha-numeric data that were put into the program; and, in addition it produces a trace of all simple variables as they are assigned values during program execution.
At the completion of program execution or when an execution error occurs, the student has an option to enter the CAI-BASIC Edit mode to correct his program one line at a time. When all editing is completed the program is executed again.
III. LIMITATIONS AND EXTENSIONS

The major limitation of CAI-BASIC is the absence of any supervision over the student's progress as he proceeds through the instruction phase. A possible extension to this project would be to add a supervisory routine to maintain the student's progress and to keep a record of his errors. Thus, having the student's progress level and error record, CAI-BASIC could be tailored to instruct the student at his own learning level. In other words, CAI-BASIC could be made into a more completely interactive teaching program (refs. 1, 4).

An additional feature that would aid student interaction is a communication link between the student and a professor so that student questions can be answered. R. R. FENICHEL (refs. 1, 2) described how students enter a special mode to type questions during the instruction session on the terminal. Then at the beginning of the student's next instruction session, all of his previous questions are answered on the terminal. FENICHEL refers to this as the "mailbox" system. Students type questions for the "mailbox" and the professor replies with answers or pertinent information for the "mailbox."

Another limitation to the existing CAI-BASIC program is the inability of CAI-BASIC to monitor the student when he enters the execution phase to write programs. The CAI-BASIC system does not oversee the student, beyond checking for syntax errors in his program. There is no method of inspecting his
programs from a tutorial point of view in order to help with semantic programming errors.

A foreseeable development to bring CAI-BASIC into the Artificial Intelligence field would be to make CAI-BASIC an "intelligent" tutor. This would remove the present inflexibility of interpreting student responses with pre-stored answers and open the possibility of CAI-BASIC understanding the logic of student responses and student programs.
IV. CONCLUSION

As stated in the objectives, the goal of CAI-BASIC was to develop a CAI program to teach a computer programming language on the Naval Postgraduate School's terminal system. The technical aim of completing the project and bringing it to an operating level has been met. However, an evaluation of the practicality of the project is still in the speculative stage.

Several computer and non-computer oriented students have tried CAI-BASIC and they found it to be both understandable and beneficial, but a full scale evaluation of the effectiveness of CAI-BASIC as a teaching tool is not possible from this limited sampling. It is sufficient to say that initial indications are that CAI-BASIC can provide a satisfactory means of learning a simple but capable programming language in a minimal amount of time. The student's participation in the learning experience encourages him to teach himself and to practice new programming skills. It is hoped that future use of CAI-BASIC by students will demonstrate its practicality.

Although the scope of CAI-BASIC was not particularly broad, the results obtained provide a basis for future CAI projects. The foundation has been laid, and the limitations and extensions of the previous section suggest a direction for future CAI efforts. Many previous CAI projects have suffered from too many or too large a scope of objectives, and as a result they never realize any practical results. This project has met its objectives and, hopefully, future CAI projects will benefit from its results.
APPENDIX A

In its present configuration the CAI-BASIC program occupies approximately ten cylinders of private disk space and requires an overlaying routine to execute under the control of CP-67/CMS which limits users to a 175K virtual machine. The overlaying routine, written by a Naval Postgraduate School Computer Center system programmer, initially loads only lesson 1 when CAI-BASIC is loaded into core. When any lesson other than lesson 1 is used, it is overlaid onto lesson 1.

The overlaying routine allows students to log on to CP-67/CMS as general users and to link into CAI-BASIC. This allows CAI-BASIC to be available to all of the terminal stations. The user gains access to CAI-BASIC by logging on the terminal as follows:

```
login yyyyxx
ENTER PASSWORD
np
ENTER 4-DIGIT PROJECT NUMBER.... ETC.
0623xx04
READY AT (TIME) ON (DATE)
CP
link 0909p 191 192
ENTER PASSWORD
teach
SET TO READ ONLY
1 cms
CMS., VERSION 01/21/71
login 192 t,p
**T (192) READ ONLY**
teach
```
Small print is typed in by the user, and the capitalized print is the computer terminal response.

The CAI-BASIC program consists of two phases, an instruction phase and an execution and editing phase. After logging on to the system, the user enters the main routine, CAIBAS, which directs the general flow of CAI-BASIC.

If the user has not used CAI-BASIC before he enters the instruction phase, he is presented with an introduction to CAI-BASIC, and then is sequentially guided through the lessons. After each lesson the user is given the opportunity to terminate his session, to review a lesson, to enter the execution phase, or to go on to the next lesson.

If the user has used CAI-BASIC before, he is given the opportunity to review any lesson or to go to the next lesson in his learning sequence.

When the execution and editing routine, TEST1, is entered the user is allowed to execute standard "BASIC" programs (refs. 3, 5, 10). The Test1 routine incorporates the "BASIC" compiler, COMPLR, which interprets each "BASIC" input statement. Each "BASIC" statement is input one line at a time; and, if a syntax error occurs on the input, COMPLR prints an error message. The user can then input the correct "BASIC" statement. When the "end" statement is input, the program is checked for global errors; and, if none occur, the program is executed.

When global errors or execution errors occur, the user is given the choice of entering the edit mode of TEST1 to correct his program, or using the "DEBUG" feature to find his programming errors.

The edit mode allows the user to delete, add to, or correct "BASIC"
statements in his program one line at a time. The user is given a listing of his program with reference numbers for editing purposes. To edit a program the user is first asked for the statement reference number, and then asked for the specific edit command.

A statement is deleted by typing in the reference number of the statement to be deleted, hitting the carriage return, and then typing in the command "DEL."

A statement is added 'after' the reference number typed in by typing in the command "ADD1" followed by the "BASIC" statement to be added. To add a statement before the first statement in the program, the reference number zero (0) is used.

A statement is corrected by typing in the reference number of the statement to be corrected, hitting carriage return, and then typing in the 'entire' correct "BASIC" statement.

The "DEBUG" feature is designed to provide useful information to the user who has encountered an execution error in his program. It is used by adding the key word "DEBUG" as a statement in the user's program.

The "DEBUG" feature gives the user a list of all numeric and alpha-numeric data that were used in the program; and, in addition, it produces a trace of all simple variables with their values as they are assigned values during program execution.
APPENDIX B

CAI-BASIC: A PROGRAM TO TEACH THE PROGRAMMING LANGUAGE "BASIC"
T.A. BARRY
U.S. NPS. 8/17/71

THE FILES USED IN CAI-BASIC UNDER CP-67/CMS ARE AS FOLLOWS:

MAIN PROGRAM---- CAIBAS

SUBROUTINES----INITIAL
CRUNCH
TEST
LESON1-> LESCN7

OVERLAYING Routines FOR CP-67/CMS:
SEARCH ALOAD EXIT

THE FOLLOWING SUBROUTINES ARE PART OF THE CAIBASIC COMPILER,
AS MODIFIED FROM THE NPS "BASIC" COMPILER USED FOR BATCH PROGRAMS,
BUT ARE NOT INCLUDED AS PART OF THE CAI-BASIC PROGRAM:

ACON COPLR ERR INSN0
APRINT CVNV DUMMY LTL
BUFFIL CPRINT EVAL RANF
CEND DIM EXEC TEST

SYMBOL TABLE: (GLOBAL SYMBOLS AND VARIABLES)

CARD VECTOR HOLDING STUDENT INPUT
CARDP VECTOR HOLDING STUDENT INPUT WITH BLANKS REMOVED
LENGTH LENGTH OF INPUT IN VECTOR CARDP
ASTRISK WHEN CARDP(1)=ASTRISK THE USER HAS MADE A TYING ERROR
ALPHA VECTOR HOLDING THE LETTERS OF THE ALPHABET
DIGIT VECTOR HOLDING THE DIGITS 0-->9
ALPHA(25) THE LETTER 'Y'
ALPHA(14) THE LETTER 'N'
CAIBAS IS THE MAIN PROGRAM. IT IS A COMPUTER AIDED INSTRUCTION PROGRAM WHICH TEACHES FUNDAMENTAL PROGRAMMING CONCEPTS USING THE LANGUAGE BASIC. THE HEART OF THE PROGRAM PROVIDES INSTRUCTION AND INTERPRETATES STUDENT RESPONSES. SOME ACED FEATURES ARE EXECUTION AND EDITING OF "BASIC" PROGRAMS IN "TEST1" SUBROUTINES CALLED FROM CAIBAS ARE:

INITIAL
CRUNCH
TEST1
LES01 --> LES07
SEARCH
LOAD

SYMBOL TABLE : (LOCAL SYMBOLS AND VARIABLES)
LSNUM CURRENT LESSON NUMBER
OLD FLAG • SET TO 1 IF USER HAS USED CAI-BASIC BEFORE
LES1 --> LES7 VARIABLES FOR OVERLAYING ROUTINES

COMMON STACK(100), PROG(250), CARD(80), CARDP(80), ALP0HA(48),
IAPTR, INPTR, IADATA(500), XDATA(500), STRING(5),
DIGIT(I0), IPRTB(I0), LIST(100), LISTT(100),
PRTR(2500), NERAS, INST, NSTLT, DEBUG, DOLGEN, QUOTE,
eQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLEFT, ASTRSK, BLANK

CCMGN INTERP, EXERR
REAL*8 LES1/LES07

99 FORMAT(4,99) 'HI • WELCOME TO CAI-BASIC • THERE ARE ONLY A FEW •'
*SIMPLE RULES TO REMEMBER IN ORDER TO HAVE A SUCCESSFUL SESSION •'
*ON THE TERMINAL WITH CAI-BASIC :  • '/ 5X,'
*1. WHEN ASKED FOR A RESPONSE • TYPE IN THE CORRECT REPLY AND '/ 5X,'
*2. IF YOU MAKE A TYPING ERROR WHILE MAKING ANY RESPONSE •'
OR INPUT 1 TYPE IN FOUR DOLLAR SIGNS ($$$$) AFTER THE ERROR ORˈ " CA100260
ANYWHERE ON THAT INPUT LINE AND HIT CARRIAGE RETURN THE ˈ " CA100270
ENTIRE LINE WILL THEN BE IGNORED AND YOU CAN TYPE IN THE CORRECT ˈ " CA100280
INPUT OR RESPONSE ˈ " 5X,ˈ "
3. IF AT ANY POINT IN THE SESSION YOU WANT TO STOP THE SESSION ˈ " CA100300
4. TYPE IN THE WORD "QUIT" AS SOON AS YOU ARE ASKED FOR THE NEXT ˈ " CA100310
RESPONSE , HIT CARRIAGE RETURN, THEN HIT ATTN KEY AND TYPE LOGOUT ˈ " 5X,ˈ "
4. DURING YOUR TERMINAL SESSION CAI-BASIC WILL HALT OCCASIONALLY ˈ " CA100330
5. TO LET YOU READ A SEQUENCE OF INFORMATION - WHEN YOU ARE ˈ " CA100340
READY TO CONTINUE TYPE IN "GO", AND HIT CARRIAGE RETURN ˈ " 5X,ˈ "
5. DURING YOUR TERMINAL SESSION YOU MAY NOTICE THAT ˈ " CA100360
THE TYPING IS NOT ALWAYS PERFECT - SOME DAYS THE COMPUTER IS ˈ " CA100370
NOT UP TO PAR AND YOU WILL HAVE TO ADJUST TO THE MINOR IRRITANT ˈ "
IF YOU ONLY WANT TO EXECUTE PROGRAMS AT THIS TIME THEN ˈ " CA100390
*REPLY : YES : OTHERWISE REPLY : NO ˈ " "//
1 READ(5,101,END=301)CARD
CALL CRUNCH(L)ENGTH
IF (CARDP(1).EQ.ASTRSK) GO TO 1
IF (CARDP(1).NE.ALPHA(25)) GO TO 2
EXECUTION ONLY PHASE
CALL TEST1
WRITE(6,109)
400 READ(5,101,END=401)CARD
CALL CRUNCH(L)ENGTH
IF (CARDP(1).EQ.ALPHA(25)) RETURN
IF (CARDP(1).EQ.ALPHA(14)) GO TO 402
401 WRITE(6,104)
GO TO 400
2 WRITE(6,104)
GO TO 400
3 WRITE(6,100)
100 FORMAT(ˈ "" IF THIS IS YOUR FIRST SESSION WITH CAIBASIC, THEN ˈ " "//
*REPLY : YES : OTHERWISE REPLY : NO ˈ " "//
4 READ(5,101,END=302)CARD
101 FORMAT(83AI)
CALL CRUNCH(L)ENGTH
IF (CARDP(1).NE.ALPHA(14)) GO TO 20
OLD USER- DETERMINE PROGRESS LEVEL AND ROUTE TO DESIRED LESSON
402 OLD=1
WRITE(6,102)
102 FORMAT(ˈ "" ; NOW YOU MAY CHOOSE TO REVIEW ANY LESSONS ˈ " "//
*OR TO BEGIN WHERE YOU ENDED DURING YOUR LAST SESSION.*

WRITE(6,105)

105 FORMAT('10X','THE LESSONS ARE AS FOLLOWS:  '/10X','LESSON 1 CA100680
*PROGRAM FORMAT AND BASIC DEFINITIONS
*   '/10X','LESSON 2 REMARKS, INPUT/OUTPUT
* AND DATA'/10X,' LESSON 3 ASSIGNMENT STATEMENTS AND BUILT IN FUNCTIONAL
*TIONS'/10X,' LESSON 4 BRANCING '/10X,' LESSON 5 LOOPING AND SUBSCRIP
*TED VARIABLES'/10X,' LESSON 6 SUBROUTINES AND RECURSION '/10X,'
*LESSON 7 SUMMARY OF BASIC STATEMENTS
*   '/*** WHEN YOU ARE READY TO CONTINUE, TYPE IN GO AND H11T '/'
*THE CARRIAGE RETURN ***/**)

READ(5,101,END=5)CARD

5 WRITE(6,103)

103 FORMAT('10X',5X,'REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO C
*OVER**/**)

6 READ(5,1C1,END=3C1)CARD

CALL CRUNCH(ILNGTH)

DO 7 I=2,8
    IF (CARDP(1).EQ.DIGIT(I)) GO TO 9
7 CONTINUE

9 LNUM=1-1
10 GO TO (11,12,13,14,15,16,17),LNUM

11 CALL LESSN1
GO TO 25

12 CALL ALOAD(LES2,N1)
GO TO 25

13 CALL ALOAD(LES3,N1)
GO TO 25

14 CALL ALOAD(LES4,N1)
GO TO 25

15 CALL ALOAD(LES5,N1)
GO TO 25

16 CALL ALOAD(LES6,N1)
GO TO 25

17 CALL ALOAD(LES7,N1)
GO TO 25

GO TO 6

104 FORMAT('10X','*** YOUR REPLY WAS TYPED INCORRECTLY; CHECK THE */*
*QUESTION AND REPLY AGAIN ***/**)

*INSTRUCTION PHASE ** NEW USER

20 IF(CARDP(1).EQ.ALPHA(25))GO TO 21

302 WRITE(6,104)

21 WRITE(6,106)

106 FORMAT('0',10X,'CAIBASIC IS A PROGRAM TO TEACH YOU THE */*

CA10G1110
*FUNDAMENTALS OF A PROGRAMMING LANGUAGE THE LANGUAGE TO BE LEARNED* CAI001120
*/ IS BASIC: A SIMPLE LANGUAGE FOR THE USER WHO HAS LITTLE KNOWLEDGE CAI001130
* OF COMPUTERS AND WHOSE PRIMARY INTEREST IS IN OBTAINING */
* RESULTS. * /lx, THE SIMPLICITY OF THE BASIC LANGUAGE AND ITS RANGE*CAI001150
* OF CAPABILITIES SHOULD ALLOW YOU TO LEARN THE LANGUAGE AND */
* WRITE PROGRAMS IN A MINIMAL AMOUNT OF TIME.* /lx, THE REFERENCE T
* EXTS RECOMMENDED FOR CAIBASIC ARE */ /lx, 1. BASIC LANGUAGE MANUA
* L, TN # C211-12 APRIL 1971/** ( FREE UPON REQUEST IN 1-147 ) */ /lx
* . 2. INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE R-L
* . NOLAN/** ( BUCK STORE / MAIN LIBRARY ) */ /lx, 3. BASIC PROGRAMMING
* . VC HARE/** ( COMPUTER CENTER LIBRARY ) **/// ** AFTER YOU HAVE CAI001170
* E FINISHED READING AN INPUT, TYPE IN GO AND HIT THE RETURN KEY**CAI001180
* AND THE PROGRAM WILL CONTINUE **///

READ5(101, END=3G7) CARD
CAI01200

307 WRITE(16, 107) CAI01210
107 FORMAT(* /lx, * DURING YOUR TERMINAL SESSION YOU WILL BE LEARNING*CAI01220
* / THE STRUCTURE OF THE LANGUAGE BASIC: THE INSTRUCTION SET */ CAI01230
* CONTAINS 7 LESSONS AND YOU MAY PROCEED THROUGH THE LESSONS AT */ CAI01240
* YOUR OWN SPEED. **///)
WRITE(6, 105) CAI01250
READ5(101, END=308) CARD
CAI01260

308 WRITE(16, 106) CAI01270
108 FORMAT(* /lx, * IN EACH LESSON YOU WILL BE GIVEN INSTRUCTION */ CAI01280
* SEQUENCES AND THEN YOU WILL BE ASKED QUESTIONS TO SEE IF YOU */ CAI01290
* UNDERSTOOD THE INSTRUCTIONS. THE QUESTIONS WILL BE OF VARIOUS */ CAI01310
* TYPES: MULTIPLE CHOICE, TRUE FALSE, ACTUAL PROGRAM STATEMENTS ETC.*CAI01320
* YOU WILL BE PROMPTED FOR YOUR ANSWER AND WHEN READY TYPE IN */
* YOUR RESPONSE AND HIT THE RETURN KEY** /lx, * IF YOU KEEP THE TELETAI01340
* TYPE OUTPUT FROM YOUR TERMINAL SESSION** YOU WILL HAVE A READY REFCAI01350
* REFERENCE FOR FUTURE USE**///
READ5(1,101,END=3G6) CARD
CAI01370

309 LSNUM=1
24 IF (LSNUM GT 7) GO TO 40
GO TO (11, 12, 13, 14, 15, 16, 17), LNUM
WRITE(6, 109) CAI01380

25 WRITE(16, 109) CAI01410
109 FORMAT(* /lx, * DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSIONCAI01420
* ??/ (REPLY: YES OR NO )///)
WRITE(1,101,END=3G1) CARD
CAI01430
CALL CRUNCH(LENGTH)
CAI01440
IF(CARDP(1), EQ, ALPHA(251)) GO TO 40
CAI01450
IF (CARDP(1), EQ, ALPHA(14)) GO TO 36
CAI01460
WRITE(6, 104) CAI01470
GO TO 28
CAI01480

310 WRITE(6, 104) CAI01490
GO TO 28
CAI01500

C OLD USERS PICK NEXT LESSON; NEW USERS GET NEXT LESSON IN SEQUENCE CAI01510
30 WRITE(16, 209) CAI01520
209 FORMAT(* /lx, * IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME */ CAI01520
* THEN REPLY: YES; OTHERWISE REPLY: NO AND THE INSTRUCTION */ CAI01500
* WILL CONTINUE **///
C}
32 READ(5,101,END=33)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ALPHA(14)) GO TO 34
IF(CARDP(1).NE.ALPHA(25)) GO TO 33

ENTER EXECUTION PHASE

CALL TEST1
GO TO 25

33 WRITE(6,104)
GO TO 32

34 IF(OLD.EQ.1) GO TO 5
WRITE(6,110)

110 FORMAT('DO YOU WANT TO REVIEW A LESSON BEFORE GOING ON ?',/REPLY :YES OR NO*/?)

27 READ(5,101,END=311)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ALPHA(25)) GO TO 5
IF(CARDP(1).EQ.ALPHA(14)) GO TO 31

311 WRITE(6,104)
GO TO 27

31 LSNUM=LSNUM+1
GO TO 24

C USER INSTRUCTION SESSION COMPLETED.

40 CONTINUE
STOP
END

SUBROUTINE TEST1

THIS ROUTINE ALLOWS EXECUTION AND EDITING OF BASIC PROGRAMS

SUBRoutines CALLED FROM TEST1 ARE:
INITIAL
CRUNCH
CGMPLR

SYMBOL TABLE : (LOCAL SYMBOLS AND VARIABLES)

NEW FLAG = SET TO 1 WHEN USER HAS EXECUTED FIRST PROGRAM TES00120
END FLAG = SET TO 1 WHEN 'END' STATEMENT READ TES00130
IEXERR FLAG = SET TO 1 WHEN AN EXECUTION/GLOBAL ERROR OCCURS TES00120
NERK FLAG = SET TO 1 WHEN A SYNTAX ERROR OCCURS IN INPUT TES00150
INTERP FLAG = SET TO 1 WHEN ONLY INTERPRETING TES00160
NStmt NUMBER OF PROGRAM STATEMENTS INPUT TES00170
SFLE SOURCE FILE VECTOR HOLDING PROGRAM STATEMENTS TES00180
NREF PROGRAM STATEMENT REFERENCE NUMBER TES00190
KTK COUNTER TO DETERMINE IF REFERENCE NUMBER IS 1 OR 2
DIGITS
IARRAY VARIABLE USED IN CAI-BASIC COMPILER
DEBUG
OLD FLAG - SET TO 1 WHEN USER HAS EDITED FIRST PROGRAM
INBIG
NFOR

COMMON
- STACK(100), IROG(200), CAKO(80), CARPO(80), ALPHA(48),
- IAPTR, INP, TINFO(50), XINFO(50), STRING(5),
- DIGIT(10), RIBLIC, LISTST(100), ISTLST(100),
- PRT(200), ERRORE, INST, NSLIST, DEBUG, DOLSN, QUOTE,
- EQUALS, PARLT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
- PARLIT, ASTRLNK, BLANK
COMMON INTERP, EXEERR
DIMENSION SPFILE(100);80)
EXECUTION PHASE

IF NEW SET DON'T PRINT INTRO AGAIN
IF (NEW. EQ. 1) GO TO 3

99 FORMAT(*15X,'**** CAIBASIC COMPILER ****',/ /* TESCO0450 */ 00460
*THE CAIBASIC COMPILER IS A LINE BY LINE INTERPRETER *THE BASIC*
*INTERPRETER ACCEPTS STANDARD BASIC STATEMENTS * AND IT ANALYZES* /* TESCO0470 */ 00460
* EACH BASIC STATEMENT AS IT IS INPUT * AN ADDITIONAL FEATURE OF* /* TESCO0480 */ 00460
*THE CAIBASIC COMPILER IS A LEXICAL AND A DEBUGGING ROUTINE * /* TESCO0490 */ 00460
*THAT ALLOWS THE USER TO ADD, DELETE AND CORRECT STATEMENTS * /* TESCO0500 */ 00460
*AND TO GET A LISTING OF ALPHA-NUMERIC AND NUMERIC DATA USED * /* TESCO0510 */ 00460
*AND A TRACE OF ALL SIMPLE VARIABLES AS THEY ARE ASSIGNED VALUES* /* TESCO0520 */ 00460
*IN THE PROGRAM*
*THE DEBUG FEATURE IS USED BY ADDING THE KEY WORD DEBUG * /* TESCO0530 */ 00460
*AS A STATEMENT TO YOUR PROGRAM */ /* TESCO0540 */ 00460
*THE EDITING MODE IS AVAILABLE TO THE USER WHEN AN EXECUTION */ /* TESCO0550 */ 00460
*ERROR OCCURS AND AFTER SUCCESSFUL EXECUTION */ /* TESCO0560 */ 00460
* IN THE EVENT OF AN INPUT ERROR THE INTERPRETER WILL ANALYZE * /* TESCO0570 */ 00460
* THE ERROR AND PRINT AN ERROR MESSAGE. IF THIS OCCURS FIND THE* /* TESCO0580 */ 00460
*ERROR AND CORRECT THE CORRECT BASIC STATEMENT FOR THE CURRENT */ /* TESCO0590 */ 00460
*LINE OF INPUT */ /* TESCO0600 */ 00460
**NOTE:** TYPING ERRORS CAN BE DELETED BY TYPING FOUR DOLLAR * /* TESCO0610 */ 00460
**SIGNS ($$$) ON THE SAME LINE AS THE ERROR * HITTING CARRIAGE */ /* TESCO0620 */ 00460
**RETURN * AND INPUTTING THE LINE AGAIN */ */ /* TESCO0630 */ 00460
**OLD** */ /* TESCO0640 */ 00460
3 WRITE(*15X,'**** CAIBASIC EXECUTION *****') /* TESCO0670 */ 00460
CALL INITIAL
NEW=1
DEBUG=C=0
NSTMT=0
IEND=0
INBIC=1
ARRAY=808
NFOR=0
INTERP=0
IXERR=0

5 WRITE(6,100)
100 FORMAT(10,5X,'INPUT BASIC PROGRAM NOW (ONE LINE AT A TIME)'///)
10 READ(5,101)END=5;CARD
101 FORMAT(8CA1)
CALL CRUNCH(ILENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 10

5 FIND "END" STATEMENT
5 DO 699 I=1,80
699 CONTINUE
5 CALL CCPLR(NFOR,ARRAY,ILNGTH,INBIC)
5 IF(NO ERRORS, FILE INPUT STATEMENT
5 IF(NERRS.EQ.0) GO TO 9
5 IF(IEXERR.EQ.1) GO TO 9
5 NERRS=0
5 GO TO 10

5 PUT IN END STATEMENT
5 NSTMT=NSTMT+1
5 IEND=1
5 DO 702 I=1,80
702 SFILK(NSTMT,I)=CARD(I)
5 GO TO 700
5 IF(NSTMT.GT.99) GO TO 511
5 IF(IEND.EQ.1) GO TO 12
5 NSTMT=NSTMT+1
5 DO 11 J=1,80
11 SFILK(NSTMT,J)=CARD(J)
5 GO TO 10
5 WRITE(6,512)
512 FORMAT(10,5X,'YOU HAVE REACHED THE MAXIMUM PROGRAM SIZE ALLOWED'///)
5 *UNDER CAI-BASIC.* YOU WILL HAVE TO MODIFY YOUR PROGRAM TO ///}
5 *" TES01140
5 TES01150
*STAY BELOW 100 PROGRAM STATEMENTS /*//)
GO TO 513

EXECUTION ERROR GO TO EDIT ROUTINE

12 IF(IEXERR.EQ.1)GO TO 31
513 WRITE(6,205)
205 FORMAT('10*1,1X,* DO YOU WANT TO EDIT YOUR PROGRAM */* REPLY : YES ; NO */')
*5 OR NU */")
312 READ(5,131)END=315)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ALPHA(25)) GO TO 33
IF(CARDP(1).EQ.ALPHA(14)) GO TO 313
315 WRITE(6, 106)
GC TO 312
313 WRITE(6,105)
105 FORMAT('10*1,1X,* IF YOU DESIRE TO HAVE A SMOOTH COPY OF YOUR */* PROGRAM WITH ITS EXECUTION THEN REPLY : YES ; OTHERWISE */* REPLY : NO */")
20 READ(5,101)END=326)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).NE.ALPHA(25))GO TO 25
PRINT SMOOTH COPY OF PROGRAM AND EXECUTE IT

314 WRITE(6,104)
104 FORMAT('10*1///////////')
DO 13 I=1,NSTMT
13 WRITE(6,103)((FILE1(J),J=1,80)
103 FORMAT('10*5X,8GAI')
135 CALL INITIAL
DEBUG=2*0
136 INBIG=1
IARRAY=808
NFOR=0
INTERP=0
IEXERR=0
DO 15 K=1,NSTMT
15 DO 14 L=1,80
14 CARD(L)=FILE(K,L)
CALL CRUNCH(ILNGTH)
15 CALL COMPIL(FOR,IARRAY,ILNGTH,INBIG)
EXECUTION ERROR GO TO EDIT ROUTINE

IF(IEXERR.EQ.1)GO TO 31
WRITE(6,104)
GO TO 26
25 IF(CARDP(1).EQ.ALPHA(1))GO TO 26  
326 WRITE(6,106)  
106 FORMAT('0',*"*** YOUR REPLY IS INCORRECTLY TYPED REPLY AGAIN***"/ )  
* TES01660  
* TES01650  
* TES01670  
* TES01680  
* TES01690  
GO TO 20  
26 WRITE(6,107)  
107 FORMAT('110X: *DO YOU WANT TO EXECUTE ANOTHER PROGRAM*/')  
* YES OR NO */  
29 READ(5,101,END=330) CARD  
CALL CRUNCH(11NGTH)  
IF(CARDP(1).EQ.ALPHA(14))GO TO 30  
IF(CARDP(1).EQ.ALPHA(25))GO To 3  
330 WRITE(6,106)  
GO TO 29  
30 RETURN  
C  
C EDIT ROUTINE.  
C  
31 WRITE(6,108)  
108 FORMAT('0*,16X:* *** EXECUTION ERROR **** 
* IF YOU WANT TO CORRECT YOUR PROGRAM NOW THEN REPLY : YES AND */  
* OTHERWISE REPLY : NO */  
208 READ(5,101,END=343)CARD  
CALL CRUNCH(11NGTH)  
IF(CARDP(1).NE.ALPHA(25))GO TO 43  
33 IF(OLD.EQ.1)GO TO 321  
WRITE(6,115)  
110 FORMAT('0*,16X:* *** CAIBASIC EDIT MODE **** */  
* BY USING THE REFERENCE NUMBERS LISTED TO THE LEFT OF YOUR BASIC */  
* PROGRAM STATEMENTS YOU MAY ADD, DELETE, OR CORRECT ONE LINE */  
* OF THE PROGRAM AT A TIME. IN ALL EDITING THE FIRST STEP IS TO*/  
* INPUT THE PROGRAM STATEMENT REFERENCE NUMBER AND HIT THE CARRIAGE*/  
* RETURN THE SECOND STEP DEPENDS ON WHAT EDITING YOU Do : */  
* 1. DELETE */  
* THE BASIC STATEMENT REFERENCED IS DELETED BY TYPING THE */  
* LETTERS OR */  
* 2. CORRECT */  
* TO CORRECT THE BASIC STATEMENT REFERENCED TYPE IN THE COMPLETE*/  
* CORRECT BASIC STATEMENT */  
* ADD */  
* A BASIC STATEMENT IS ADDED **AFTER** THE BASIC STATEMENT */  
* REFERENCED BY TYPING THE LETTERS ADD1 FOLLOWED BY THE BASIC */  
* STATEMENT. ALL BLANKS FOLLOWING THE LETTERS ADD1 WILL BE */  
* INCLUDED IN THE BASIC STATEMENT. TO PLACE A STATEMENT **BEFORE** */  
* THE FIRST STATEMENT IN THE PROGRAM, USE THE REFERENCE NUMBER GES02090 */  
* */  
OLD=1  
TES02110
WRITE PROGRAM WITH REFERENCE NUMBERS

DO 32 I=1,NSTMT
32 WRITE(6,109) I,(SFILE(I,J),J=1,80)

109 FORMAT(' ',14,I0X,80A1)

WRITE(6,111)
111 FORMAT('0', 'INPUT REFERENCE NUMBER NOW')
KNT=0
READ(5,101,END=331)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(I).EQ.ASTRSK) GO TO 35
DO 237 I=1,2
DO 235 J=1,16

CHECK FOR DIGIT
IF(CARDP(I).EQ.DIGIT(J)) GO TO 236
CONTINUE
IF(J.NE.10.AND.CARDP(I).EQ.BLANK) GO TO 237
WRITE(6,108)
GO TO 34

CONVERT ALPHA CHARACTER TO NUMBER
KNT=KNT+1
IF(KNT.EQ.1) NREF=J-1
IF(KNT.EQ.2) NREF=10*NREF+(J-1)
CONTINUE

CHECK FOR LEGAL REFERENCE NUMBER
IF(NREF.LE.NSTMT.AND.NREF.GE.0) GO TO 36
WRITE(6,106)
GO TO 35

WRITE(6,112)
112 FORMAT('0', 'INPUT EDITING NOW (DEL,BASIC STATEMENT,ADD1..../) //')
37 READ(5,101,END=36)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(I).EQ.ASTRSK) GO TO 36

DELETE CURRENT LINE
IF(CARDP(I).EQ.CMINUS) GO TO 50

ADD A STATEMENT AFTER CURRENT LINE
IF(CARDP(I).EQ.PLUS) GO TO 55
DELET STATEMENT

50 NSTMT=NSTMT-1
51 DO 52 I=NREF,NSTMT
52 DO 52 J=1,8C
53 SFIE(I,J)=SFIE(I+1,J)
54 GO TO 47

ADD A BASIC STATEMENT AFTER CURRENT LINE

55 IF(NSTMT.GT.99) GO TO 65
56 NSTMT=NSTMT +1
57 ITEMP=NSTMT
58 DEBUG=0,0
59 INTP=1
60 IEXRR=0

FIND BEGINNING OF STATEMENT

61 DO 56 K=1,80
62 IF(CARD(K).EQ.BLANK) GO TO 56
63 IF(CARD(K).NE.ALPHA(1).OR.CARD(K+2).NE.ALPHA(4)) GO TO 56
64 IF(CARD(K+3).EQ.DIGIT(2)) GO TO 60

CONTINUE

65 CONTINUE
66 WRITE(6,106)
67 GG TO 37
68 IBEG=K+3
69 DO 600 L=1,80
70 IBEG=IBEG+1
71 IF(IBEG._GT.80) GO TO 601
72 IF(CARD(L).EQ.CARD(IBEG))
73 CONTINUE

CALL CRUNCH(ILNTH)

74 CALL CRUNCH(ILNTH,AVRSK) GO TO 36
75 DO 603 J=1,86
76 IF(CARD(B+3).EQ.ALPHA(14).AND.CARD(B+3).EQ.ALPHA(20)) GO TO 602

CONTINUE

77 CONTINUE
78 CALL COMPLR(NFOR,IARRAY,ILNTH,INBIG)
79 IF NO ERRORS THEN FILE STATEMENT

80 IF(NERRS.EQ.0) GO TO 602
81 NERRS=0
82 GO TO 36

ADD STATEMENT TO PROGRAM FILE. MOVE STATEMENTS UP AND DOWN

83 IADD=NREF+1
DO 62 I=1,80
   SF=FILE(ITEMP,I)=SF=FILE(ITEMP-1,I)
   ITEMP=ITEMP-1
   IF(ITEMP.GT.IADD) GO TO 61
   DO 63 J=1,80
   SF=FILE(IADD,J)=CARD(J)
   GO TO 47
65 WRITE(6,512)
GO TO 47
END

SUBROUTINE CRUNCH(ILNGTH)
GIVEN THE VECTOR CARD; THIS SUBROUTINE REMOVES ALL BLANKS IN CARDCR000030
AND RETURNS THE BLANK-LESS VERSION IN CARDP*.CR000040
THIS ROUTINE ALSO CHECKS FOR TYPING ERRORS AND CHECKS IF THE USER CR000030
IS COMPLETED WITH HIS SESSION OR IN AN EDIT MODE CR000060
COMMON
- STACK(100), PRG(2000), CARD(80), CARDP(80), ALPHA(48), CR000090
- IAPTR, INPTR, IDATA(500), XDATA(500), STRING(5), CR000120
- DIGIT(10), IPRITB(10), LIST(100), ISTAT(100), CR000110
- PTR(2500), NERRS, INST, NSNIST, DEBUG, DOLGNS, QUOTE, CR000120
- EQUALS, PARL, DECIMAL, PLUS, CMINUS, SLASH, COMMA, CR000130
- COMMON, INTERP, IERRCR000140
DO 19 I=1,80
CARDP(I)=BLANK
REMOBE BLANKS
ILNGTH=0
DO 20 I=1,80
IF [CARD(I).EQ.BLANK) GO TO 20
ILNGTH=ILNGTH+1
CARDP(ILNGTH)=CARD(I)
CONTINUE
CHECK FOR BLANK INPUT
IF(ILNGTH.EQ.0) CARDP(1)=ASTRKR CR000190
DO 25 I=1,80
CHECK FOR TYPING ERROR($$$$)
IF(CARDP(I).NE.DOLGNS).OR.CARDP(I+1).NE.DOLGNS) GO TO 26
IF(CARDP(I+2).EQ.DOLGNS.AND.CARDP(I+3).EQ.DOLGNS) CARDP(1)=ASTRKR CR000300
CHECK FOR END OF SESSION (QUIT)

26 IF(CARDP(I).NE.ALPHA(17).OR.CARDP(I+1).NE.ALPHA(21)) GO TO 27
IF(CARDP(I+2).EQ.ALPHA(9).AND.CARDP(I+3).EQ.ALPHA(20)) STOP

CHECK FOR EDIT COMMAND (ADD)

27 IF(CARDP(I).NE.ALPHA(1).OR.CARDP(I+1).NE.ALPHA(4)) GO TO 28
IF(CARDP(I+2).EQ.ALPHA(4).AND.CARDP(I+3).EQ.DIGIT(2).AND.CARDP(1)=PLUS) STOP

CHECK FOR EDIT COMMAND (DEL)

28 IF(CARDP(I).NE.ALPHA(1).OR.CARDP(I+1).NE.ALPHA(5)) GO TO 25
IF(CARDP(I+2).EQ.ALPHA(12).AND.CARDP(1)=CMINUS) RETURN

SUBROUTINE INITIAL

THIS ROUTINE Initializes THE SYSTEM. IT LOADS ALL VECTORS
WITH APPROPRIATE INFORMATION TO BE USED BY THE SYSTEM. IT ALSO
SETS CERTAIN POINTERS TO THEIR INITIAL VALUES.

COMMON

STACK(100), PROG(200C), CARD(80), CARDP(80), ALPHA(48), L
IAPTR, INPTR, IADATA(500), XADATA(500), STRING(5),
DIGIT(1C), IPRT(10), LIST(T100), IRLIST(100),
PRT(250C), NERRS, INST, NILST, DEBUG, DOLSN, QUOTE,
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK

COMMON /RANDOM/IRAND
DIMENSION ATMP(26), DITMP(10), CHARTM(12)
DATA ATMP/1HA, 1H5, 1PC, 1MD, 1HE, 1HF, 1HH, 1HI, 1HJ/,
1HK, 1HL, 1HP, 1HN, 1HC, 1HR, 1HS, 1HT, 1HU, 1HM, 1H6, 1H7, 1H8, 1H9/,
DATA DITMP/1H0, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9/,
DATA CHARTM/1H*, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9/,
1H*, 1H*/

INITIALIZE

IRAND=5555
NERRS=0
INST=0
NILST=0
IAPTR=C
INPTR=0

C
INITIALIZE PRIORITY TABLE
IPRITB(1)=2
IPRITB(2)=2
IPRITB(3)=3
IPRITB(4)=3
IPRITB(5)=4
IPRITB(6)=5

C
SET UP VOCABULARY
LOAD ALPHABET

DC 10 I=1,26
10 ALPHA(I)=ATEMP(I)

C
LOAD DIGITS

DO 20 I=1,10
20 DIGIT(I)=DIGTMP(I)

C
LOAD DIGITS INTO ALPHA

LOC=26
DC 12 I=1,10
LCC=LOC+1
12 ALPHA(LOC)=DIGIT(I)

C
LOAD SPECIAL CHAR INTO ALPHA

DO 16 I=1,12
16 ALPHA(LOC)=CHARTM(I)

C
INITIALIZE ARRAY STORAGE POINTERS

DO 50 I=287,384,4
50 PRV(I)=0.0

C
LOAD SPECIAL CHARACTERS

ASTR=CHARTM(1)
BLANK =CHARTM(2)
COMMA =CHARTM(3)
DECMAL=CHARTM(4)
EQUALS=CHARTM(5)
PARRT =CHARTM(6)
PARLFCT = CHARTM(7)  
PLUS = CHARTM(8)  
QUOTE = CHARTM(9)  
CGLSSTN = CHARTM(10)  
CHINUS = CHARTM(11)  
SLASH = CHARTM(12)  
RETURN  
END  

THIS LESSON COVERS PROGRAM FORMAT AND MISC INFORMATION SUCH AS VARIABLES, NUMBERS, KEY WORDS, EXPRESSIONS, ETC.
*BY AN END STATEMENT. THE WORDS : REM, READ, LET, PRINT DATA *LES00380*
*/ AND END ARE KEY WORDS THAT MAKE UP A BASIC STATEMENT. */ LES00390*
READ(5,101), END=304, CARD /*LES00400*/
304 WRITE(6,104)
104 FORMAT('0',1GX, 'MOST OF THE KEY WORDS USED IN THE BASIC
*/ STATEMENTS ARE SELF-EXPLANATORY:
*/ REM ALLOWS REMARKS/COMMENTS
*/ READ M,G ASSIGNS NUMBERS IN THE DATA STATEMENT TO THE
*/ VARIABLES M AND G
*/ LET ASSIGNS THE RESULT OF M DIVIDED BY G INTO VARIABLE T
*/ PRINT*STRING* CAUSES THE STRING IN SINGLE QUOTES TO BE
*/ PRINTED LITERALLY
*/ END TELLS THE COMPUTER THAT THE INPUT PROGRAM IS TO BE
*/ EXECUTED*/)
READ(5,101), END=305, CARD /*LES00500*/
305 WRITE(6,105)
105 FORMAT('0',1GX, 'IF AT THIS POINT YOU WOULD LIKE TO RUN THE
*/ SAMPLE PROGRAM TO GAIN SOME CONFIDENCE IN THE COMPUTER AND ITS*
*/ ABILITY TO PROVIDE SPEEDY RESULTS, THEN REPLY : YES ; OTHER-*//
*/ WISE REPLY : NO AND THE INSTRUCTION WILL CONTINUE */
READ(5,101), END=306, CARD /*LES00600*/
10 READ(5,101), END=306, CARD /*LES00600*/
CALL CRUNCH LENGTH /*LES00580*/
IF (CARDP(1).EQ.ASTRSK) GO TO 10
IF (CARDP(1).EQ.ALPHA(14)) GO TO 25
IF (CARDP(1).EQ.ALPHA(25)) GO TO 15
306 WRITE(6,106)
106 FORMAT(0,1GX, '*** YOUR REPLY WAS TYPED INCORRECTLY ; CHECK THE
*/ QUESTION AND REPLY AGAIN *** /)
GO TO 10 /*LES00620*/
EXECUTE SAMPLE PROGRAM /*LES00623*/
15 WRITE(6,107)
107 FORMAT(0,1GX, 'TO EXECUTE THE SAMPLE PROGRAM TYPE IN THE BASIC/*LES00630*/
*/ STATEMENTS APPEAR AND DONT WORRY ABOUT SPACING THE
*/ COMPUTER ANALYZES EACH BASIC STATEMENT AS IT IS INPUT : AND */
*/ IF THERE ARE NO ERRORS , IT UNLOCKS THE KEYBOARD AND WAITS FOR */LES00720*/
*/ YOUR NEXT INPUT. WHEN THE END STATEMENT IS INPUT THE PROGRAM*/LES00730*/
*/ IS EXECUTED */)
READ(5,101), END=308, CARD /*LES00740*/
308 WRITE(6,108) /*LES00760*/
108 FORMAT(0,1GX, 'HOWEVER, IF YOU MAKE AN ERROR THE COMPUTER WILL
*/ TELL YOU THE ERROR, AND EXPECT A CORRECTION. DON'T WORRY ABOUTLES00780*/
*/ THE ERROR; CHECK YOUR INPUT AGAINST THE SAMPLE PROGRAM ANDLES00790*/
*/ INPUT THE CORRECT STATEMENT */LES00900*)
* INPUT FOUR ASTERIX(****), HIT RETURN, AND THEN INPUT THE */LES02910*/
*CORRECT STATEMENT */10X,* YOU MAY USE ANY PAIR OF INTEGER OR DECILES00820
* MAL NUMBERS*\*/ FOR INPUT DATA YOU MAY OMIT THE REM STATEMENTS AND LES00830 * THE PRINT **MID TRAVELED** ETC*/ REM STATEMENT IF YOU DONT WANT LES00840 *T TO TYPE A LOT OF STATEMENTS**// LES00850 CALL TST1 0,1 LES00860 25 WRITE(0,111) 1,LES00870 111 FORMAT('C*•5X,'B* PROGRAM FORMAT*/5X,' A BASIC PROGRAM CONSISTS * OF A SEQUENCE OF BASIC STATEMENTS , ONE*/ STATEMENT PER INPUT LILES00890 *NE \*/ FOLLOWED BY AN END STATEMENT */ BECAUSE NO BASIC STATEMENTSLES00930 * MAY BE LONGER THAN ONE INPUT LINE (80 SPACES), */ THERE IS NO PRELES00910 *OVISION FOR CONTINUING STATEMENTS FROM ONE LINE*/ TO THE NEXT ** LES00920 *ONEVER \*/ YOU MAY SPACE THE INPUT LINE */ AS DESIRED FOR READABILLES00930 *TY SINCE THE COMPUTER IGNORES BLANKS IN BASIC */LES00940 READ(5+101,END=3091)CARD LES00950 309 WRITE(0,112) 1,LES00960 112 FORMAT('C*•10X,'1 STATEMENT NUMBERS *\*/ EACH BASIC STATEMENT MAY HAVE AN OPTIONAL // LES00980 * STATEMENT NUMBER PRECEDING IT FOR IDENTIFICATION PURPOSES */ LES00990 *THIS STATEMENT NUMBER MUST BE AN INTEGER BETWEEN 1 \*/ TO 9999 */ LES01000 *5X,* FOR EXAMPLE : 12 READ M,G LES01010 *IOX,* 2 KEY WORDS */ THE KEY WORDS THAT MAKE UP A LES01020 * BASIC STATEMENT (REM ,READ ,LET ,ETC) */ ARE SPECIAL TERMINAL SYMLES01030 *BCLS THAT ARE RECOGNIZED BY THE COMPUTER*/ AND FOR THIS REASON THLES01040 *EY MUST BE SPelled CORRECTLY AND ONLY **// USED IN BASIC STATEMENTS LES01050 */ **// THE END STATEMENT INDICATES THAT THE INPUT PROGRAM IS?*/ LES01060 *COMPLETED AND THAT PROGRAM EXECUTION IS TO BEGIN \*/ THE **// LES01070 *STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM */LES01080 *STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM */ LES01090 READ(5+101,END=3131)CARD LES01100 313 WRITE(0,113) 1,LES01110 113 FORMAT('C*•10X,' YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT*/ LES01110 * WHAT YOU HAVE JUST LEARNED */LES01120 WRITE(0,114) 1,LES01130 114 FORMAT('D*•10X,' 1 ) IS THIS A LEGAL BASIC PROGRAM(REPLY : YES OR LES01140 *NO)/**//5X,* REM ONE LINE DO NOTHING PROGRAM*/5X,' END**// LES01150 30 READ(5+101,END=3151)CARD LES01150 CALL CRUNCH(LINGTH) IF(CARDP(1).EQ•ASTRKS) GO TO 30 IF(CARDP(1).EQ•ALPHA(14).OR.CARDP(1).EQ•ALPHA(251)) GO TO 31 315 WRITE(0,106) GO TO 30 31 WRITE(0,115) 1,LES01230 115 FORMAT('D*•10X,' THE SIMPLEST BASIC PROGRAM CONSISTS OF JUST AN /* END ** STATEMENT */ LES01240 WRITE(0,116) 1,LES01250 116 FORMAT('D*•10X,' 2 ) WHICH OF THE FOLLOWING BASIC STATEMENTS IS**/ LES01260 * IN THE PROPER FORMAT */15X,* A=15 LET T=M/G/15X,* B=15LET=T=LES01270 *M/G*/15X,* C=15 LET T=M/G */10X,* REPLY A , B , C OR LES01280 *ALL */LES01290 32 READ(5+101,END=3171)CARD LES01300
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 32
IF(CARDP(1).EQ.ALPHA(1).OR.CARDP(1).EQ.ALPHA(2)) GO TO 33
IF(CARDP(1).EQ.ALPHA(3)) GO TO 33
317 WRITE(6,106)
GO TO 32
33 WRITE(6,117)
117 FORMAT(*/)
* ALL OF THE ABOVE BASIC STATEMENTS ARE CORRECT BECAUSE
* * SPACES ARE DISREGARDED. NOTE THAT STATEMENT B, * ALTHOUGH
* CORRECT IS CONFUSING TO READ. BLANKS AND INDENTATIONS MAKE A*
* PROGRAM EASY TO READ. */
* WRITE(6,218)
218 FORMAT(*/)
*C. ALPHONUMERIC CHARACTERS
*ALPHONUMERIC CHARACTERS ARE THE LEGAL CHARACTERS * Digits, AND
* SPECIAL CHARACTERS THAT CAN BE USED IN BASIC. */
* 1. CHARACTERS
*CHARACTERS CONSIST OF THE LETTERS IN THE ALPHABET A->Z */
* 2. DIGITS
*DIGITS ARE THE SINGLE NUMBERS 0->9 */
* 3. SPECIAL CHARACTERS
* THE SPECIAL CHARACTERS ARE ** */ + - ( ) = , $ */
*/ X */ A STRING */ A STRING IS ANY LIST OF ALPHONUMERIC CHARACTERS */
*/ ENCLOSED IN **SINGLE** QUOTES. FOR EXAMPLE **/5X*/ THIS IS ALE*/
* STRING */
READ(5,101,END=318)CARD
88
318 WRITE(6,118)
118 FORMAT(*/)
*INPUT */ BASIC THERE ARE THREE TYPES OF VARIABLES: SIMPLE, ALPHONUMERIC
* AND SUBSCRIPTED. SUBSCRIPTED VARIABLES WILL BE COVERED IN
*LESSON 5 */
* SIMPLE VARIABLES
* SIMPLE VARIABLES ARE IDENTIFIED BY A SINGLE LETTER
* OR A SINGLE LETTER FOLLOWED BY A DIGIT BETWEEN 0-9
* 5X, FOR EXAMPLE: A, A3, Z07, AND $X ARE LEGAL VARIABLES; BUT 5LEON, 120X, 9Z
* 7 AB, AND XXY ARE ILLEGAL VARIABLES */ THEREFORE
* 2. YOU HAVE 5 SIMPLE VARIABLES FOR USE IN YOUR PROGRAMS. */
* READ(5,101,END=319)CARD
319 WRITE(6,119)
119 FORMAT(*/)
* ALPHA VARIABLES ARE USED FOR ALPHONUMERIC MANIPULATION*/
* LATIONS */ IN WHICH A GROUP OF ALPHONUMERIC CHARACTERS, CALLED */
* A STRING, ARE REPRESENTED BY AN ALPHA VARIABLE. THE ALPHA */
* VARIABLE CONSISTS OF A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN, */
* THE MAXIMUM LENGTH OF THE ALPHONUMERIC STRING ASSIGNED*/
* TO THE ALPHA VARIABLE IS 16 CHARACTERS. */
* FOR EXAMPLE: A$, X$, Z$ ARE LEGAL ALPHA VARIABLES */
* A SIMPLE SELECTION OF ALPHA VARIABLES FOLLOWS: */5X, */ READ A$, B$, */ THILEON, 70
* PRINT A$, B$, */5X, */ DATA **MONDAY** */, 21 JUNE**, */5X, */ END */
* $ PROGRAM WILL PRODUCE THE OUTPUT: */10X, */ MONDAY 21 JUNE*/
* LESO1310
* LESO1320
* LESO1330
* LESO1340
* LESO1350
* LESO1360
* LESO1370
* LESO1380
* LESO1390
* LESO1400
* LESO1410
* LESO1420
* LESO1430
* LESO1440
* LESO1450
* LESO1460
* LESO1470
* LESO1480
* LESO1490
* LESO1500
* LESO1510
* LESO1520
* LESO1530
* LESO1540
* LESO1550
* LESO1560
* LESO1570
* LESO1580
* LESO1590
* LESO1600
* LESO1610
* LESO1620
* LESO1630
* LESO1640
* LESO1650
* LESO1660
* LESO1670
* LESO1680
* LESO1690
* LESO1700
* LESO1710
* LESO1720
* LESO1730
* LESO1740
* LESO1750
* LESO1760
* LESO1770
* LESO1780
CALL CRUNCH(ILNGTH)
IF(CARP(1).EQ.ASTRSK) GO TO 34
IF(CARP(1).EQ.DIGIT(8)) GO TO 37
DO 35 I=1,10
IF(CARP(1).EQ.DIGIT(1)) GO TO 36
35 CONTINUE
324 WRITE(6,106)
GO TO 34
124 FORMAT('0',* YOUR ANSWER IS INCORRECT * THE EXPRESSION IS EVALUATE)
*AS FOLLOWS *')
* WRITE(6,124)
WRITE(6,125)
125 FORMAT(10,X,' 4+6/2 -> 4+3 -> 7')
37 WRITE(6,126)
126 FORMAT(10*,10X,' (4+6)/2 HAS THE VALUE (REPLY WITH VALUE)')
38 READ(5,101,END=397)CARD
CALL CRUNCH(ILNGTH)
IF(CARP(1).EQ.ASTRSK) GO TO 38
IF(CARP(1).EQ.DIGIT(6)) GO TO 41
DO 39 I=1,10
IF(CARP(1).EQ.DIGIT(1)) GO TO 40
39 CONTINUE
327 WRITE(6,106)
GO TO 38
40 WRITE(6,124)
WRITE(6,127)
127 FORMAT(10*,10X,' (4+6)/2 -> 10/2 -> 5')
41 WRITE(6,128)
128 FORMAT(10*,10X,' (4+6)/2**2 HAS THE VALUE (REPLY : VALUE)')
42 READ(5,101,END=46)CARD
CALL CRUNCH(ILNGTH)
IF(CARP(1).EQ.ASTRSK) GO TO 42
IF(CARP(1).EQ.DIGIT(3).AND.CARP(2).EQ.DIGIT(6)) GO TO 48
DO 43 I=1,10
IF(CARP(1).EQ.DIGIT(1)) GO TO 44
43 CONTINUE
GO TO 46
44 CONTINUE
DO 45 I=1,10
IF(CARP(2).EQ.DIGIT(1)) GO TO 47
45 CONTINUE
46 WRITE(6,106)
GO TO 42
47 WRITE(6,124)
WRITE(6,129)
129 FORMAT(10*,10X,' (4+6)/2**2 -> (10/2)**2 -> (5)**2 -> 25')
48 WRITE(6,130)
IF(CARDP(5).NE. ALPHA(1) .OR. CARDP(6).NE.DIGIT(2)) GO TO 58
IF(CARDP(7).NE. PLUS .OR.CARDP(8).NE.ALPHA(26)) GO TO 58
IF(CARDP(9).EQ.DIGIT(2) AND.CARDP(10).EQ.PARRT) GO TO 59
58 WRITE(6,135) LEO3230
135 FORMAT( 'O*,* THE EXPRESSION SHOULD BE /10X,* A1=B2*(A1+Z1),*'/
* EACH ARITHMETIC OPERATION MUST BE WRITTEN OUT; A(B) IS NOT ASSUMED/
*ED**, TO BE A*(B) **/
59 WRITE(6,136) LEO3270
136 FORMAT ('O*,10X,'**, LINE 4 IS SIMILAR TO LINE 3. ANY ERRORS?*/
* REPLY: YES OR NO) **/
60 READ(5,101) END=361) CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.ASTRK) GO TO 60
IF(CARDP(1).EQ.ALPHA(25)) GO TO 64
IF(CARDP(1).EQ.ALPHA(14)) GO TO 65
361 WRITE(6,106) LEO3300
60 WRITE(6,137) LEO3320
137 FORMAT('O*,* THE EXPRESSION IS CORRECT.**
65 WRITE(6,138) LEO3340
138 FORMAT('O*,10X,'**, LINE 5 CONTAINS A LET STATEMENT FOLLOWED*/
* BY A NUMBER. ANY ERRORS (REPLY: YES OR NO) ?*/
70 READ(5,101) END=371) CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.ASTRK) GO TO 70
IF(CARDP(1).EQ.ALPHA(14) .OR. CARDP(1).EQ.ALPHA(25)) GO TO 71
371 WRITE(6,106) LEO3350
60 WRITE(6,139) LEO3370
139 FORMAT('O*,* THE NUMBER CONTAINS 10 DIGITS AND THE MAXIMUM ALLOWED*/
* IS 9 DIGITS.**
64 WRITE(6,140) LEO3390
140 FORMAT('O*,10X,'**, LINE 6 CONTAINS A PRINT STATEMENT FOLLOWED*/
* BY A LIST OF VARIABLES. ANY ERRORS (REPLY: YES OR NO)?*/
72 READ(5,101) END=373) CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.ASTRK) GO TO 72
IF(CARDP(1).EQ.ALPHA(14)) GO TO 74
IF(CARDP(1).EQ.ALPHA(25)) GO TO 73
373 WRITE(6,106) LEO3400
72 WRITE(6,132) LEO3420
73 WRITE(6,140) LEO3440
141 FORMAT('O*,* THE ILLEGAL VARIABLE IS 1A. SIMPLE VARIABLES ARE A*/
**' LETTER OR A LETTER FOLLOWED BY A SINGLE DIGIT ,IE. A1 **'  

75 WRITE(6,142)  

142 FORMAT(0,10X,' THERE ARE NO ERRORS IN LINE 7. IS THE PROGRAM'/'
* READY TO EXECUTE (ASSUMING ABOVE ERRORS CORRECTED)/' (REPLY : YE

*S OR NO')/

750 READ(5,101,END=80)CARD  

CALL CRUNCH(LENGTH)  

IF(CARDP(1).EQ.ASTRSK) GO TO 750  

IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 80  

WRITE(6,106)  

80 WRITE(6,143)  

143 FORMAT(0,10X,' THE PROGRAM FORMAT REQUIRES THAT AN END STATEMENT'LES0980  

**'/' BE THE LAST STATEMENT OF THE PROGRAM. THIS SAMPLE PROGRAMLES0980  

*M'/' WOULD NOT EXECUTE '/// THIS CONCLUDES THE REVIEW QUESTIONS'LES0980  

** FROM LESSON 1 '///

CALL EXIT

END

** LESSON 2 INTRODUCES THE REM , PRINT , READ , DATA , RESTORE ,  

AND RESTORE STATEMENTS

COMMON  

STACK(100) , PROG(200) , CARD(80) , CARDP(80) , ALPHA(48) ,  

IAPTR , INPTR , IADATA(500) , XDATA(500) , STRING(5) ,  

DIGIT(10) , IPRITB(10) , LIST(100) , IINSTLST(100) ,  

PRT(250) , NERRS , INST , NINSTLST , DEBUG , DOLSGN , DOUL ,  

EQUALS , PARRT , DECIMAL , PLUS , CMINUS , SLASH , COMMA ,  

PARLET , ASTRSK , BLANK

COMMON INTERP, IEXERR

REAL*8 LES2/*LES2N /*

WRITE(6,101)  

100 FORMAT(0,5X,' *** LESSON 2. ***'  

* THIS INSTRUCTION SEQUENCE WILL INTRODUCE YOU TO THE BASIC LANGUAGELES0/0  

* STATEMENTS , REM , PRINT , READ , DATA , USING THESE STATEMENTSLES0/0  

*ENTS'/' YOU WILL BE ABLE TO CONSTRUCT AND EXECUTE ELEMENTARY PROGLES0/0  

*AMS '/// 1X THE FORM FOR EACH BASIC STATEMENT WILL INCLUDE '/// 10  

* X , '/** KEY WORD ' /// ELEMENTS OR LIST OF ELEMENTS SEPARATED BY COMMALES0/0  

* 'S' /// '/// LES0/0  

* THE CARRY SYMBOLS '<< >>' DELINATE THE LEGAL ITEMS THAT MAY'/' LES0/0  

*FOLLOW THE KEYWORD AND MAKE UP THE BASIC STATEMENT '///

READ(5,101,END=3C2)CARD

101 FORMAT(80A1)  

302 WRITE(6,102)  

102 FORMAT(0,10X,' A. REM  

* '///5X,' THE BASIC STATEMENT WHICH ALLOWS YOU TO INSERT '///LES0/0  

* REMARKS INTO YOUR PROGRAM IS IDENTIFIED BY THE KEY WORD '/**REM'/'LES0/0  

* '/// REM IS A NON-EXECUTING STATEMENT WHICH MAY BE USED OPTIONALLY'LES0/0
"" AT ANY PLACE IN YOUR PROGRAM TO INTRODUCE A PROGRAM NAME. TO "" LE000310
"" EXPLAIN VARIABLES, TO DOCUMENT YOUR PROGRAM, ETC. "" LE000320
READ(5,101,END=3031CARD
303 WRITE(6,103)
103 FORMAT(*'= 5X, 'THE FORM FOR THE REM STATEMENT IS: ' '15X,' 5X, 'THE REM STATEMENT IS IGNORED BY THE CAIBASIC COMPILER AND IS'' LE00370
*CNLY FOR YOUR INFORMATION: ' '15X.' FOR EXAMPLE : ' '10X, ' REM PROGRO LE00380
*A TO COMPUTE INCOME TAX.'' LE00390
READ(5,101,END=3041CARD
304 WRITE(6,104)
104 FORMAT(*'= 10X,' B. PRINT ' '15X,' THE PRINT STATEMENT IS THE METHOD OF WRITING OUT THE RESULTS OF.' LE00430
* THE BASIC PROGRAM, TO DISPLAY VALUES OF VARIABLES, TO LABEL." '' LE00440
*THE RESULTS, AND TO SKIP A LINE. THE FORM OF THE PRINT " " LE00450
*STATEMENT IS: ' '1X,' PRINT '< EXPRESSION OR ' 'STRING' ' >' 「EXPRES " LE00460
*SESSION OR ' 'STRING' ' >'/ ' '' MULTIPLE ELEMENTS IN THE PRINT LIST ARE LE00470
* SEPARATED BY COMMAS.'' LE00480
READ(5,101,END=3051CARD
305 WRITE(6,105)
105 FORMAT(*'= 5X,' 1X, 'PRINT << EXPRESSION >> WILL WRITE OUT THE' ' '' LE00520
*CURRENT VALUE OF THE EXPRESSION, WHERE AN EXPRESSION AS DEFINED.' '' LE00530
*IN LESSON 1 WAS A NUMBER, A VARIABLE, OR AN ARITHMETIC ' ' '' LE00550
*EXPRESSION THAT IS TO BE EVALUATED.' '10X,' FOR EXAMPLE : ' '10X,' LE00540
*PRINT J ' A,B,5**2,' ' ASSUMING THAT A=10.6, B=13.3 WOULD PRINT ' ' '' LE00550
* ' '10X,' ' 10.6 13.3 25' ' '' LE00560
READ(5,101,END=3061CARD
306 WRITE(6,106)
106 FORMAT(*'= 5X,' 2X, 'PRINT ' ' '<STRING' ' > WILL PRINT OUT ALL THE' ' '' LE00580
* '' ALPHA NUMERIC CHARACTERS OF THE STRING WITHIN ' 'SINGLE' ' QUOTES' '' LE00600
*S. ' ' THIS FORM IS USED FOR LABELING THE COMPUTER OUTPUT. ' ' '' LE00610
*FOR EXAMPLE: ' 'PRINT ' ' ' ' THE ANSWER IS ' ' '' LE00620
*PRODUCES THE RESULT: ' '5X,' ' THE ANSWER IS ' '5X,' '3,' 'PRINT BY ' ' '' LE00630
* ITSELF IS USED TO SKIP A LINE ON THE COMPUTER OUTPUT.' ' '' LE00640
READ(5,101,END=3071CARD
307 WRITE(6,107)
107 FORMAT(*'= 5X,' THE COMPUTER OUTPUT SHEET IS DIVIDED INTO 8 ZONES.' ' '' LE00660
* EACH 15 COLUMNS WIDE.' 'PRINT ZONES CAN BE SKIPPED BY PUTTING A' ' '' LE00680
* BLANK IN THE PRINT LIST. FOR EXAMPLE: ' '15X,' ' PRINT A, B, X' ' '' LE00690
*OUTPUTS THE VALUE OF A IN THE SECOND ZONE, AND PUTS X IN THE THIRD ZONE.' ' '' LE00700
*SKIPS THE THIRD ZONE, AND PUTS X IN THE FOURTH ZONE.' ' '' LE00710
*ALPHA VARIABLES AND ' 'STRINGS' ' MAY EXTEND OVER SEVERAL.' ' '' LE00720
*ZONES, BUT NUMERIC RESULTS ARE LEFT ADJUSTED IN THE SPECIFIED.' ' '' LE00730
*ZONE.' ' '' IF MORE THAN EIGHT ITEMS OCCUR IN THE PRINT LIST, THE' ' '' LE00750
*ITEMS WILL OVERFLOW AND BE PRINTED ON THE NEXT LINE.' ' '' LE00770
READ(5,101,END=3081CARD
308 WRITE(6,108)
108 FORMAT(*'= 5X,' USING THE PRINT AND END STATEMENT YOU NOW HAVE THE' ' '' LE00780
* FACILITY TO EXECUTE YOUR FIRST PROGRAMS * FOR EXAMPLE : */10X;* LES00790
*REM PROGRAM TO COMPUTE THE SQUARE OF A NUMBER */10X;* LES00800
*PRINT"5 SQUARE":"5**2"/10X, END = LES0810
*PRODUCES SQUARE RESULT :"/5X, 5 SQUARED = 25
*YOU WILL NOW BE GIVEN 10C PROBLEMS TO SOLVE * YOU WILL ENTER */LES0830
*THE EXECUTION PHASE OF CABSITIC WHERE YOU CAN RUN YOUR PROBLEMS */LES0840
* AND THEN YOU WILL RETURN TO FINISH THE LESSON. */5X, 1) FIND THE LES0850
* SQUARE ROOT OF 5 SQUARED MINUS 6 TIMES 2 TIMES 2 */5X. 2) FIND LES0860
* THE VALUE OF 3.1416(8 CUBED)/12 /* WHERE B=2.50, H=3.03 */ LES0880

309 WRITE(6,109)
109 FORMAT(0); IF YOU WISH TO SKIP THESE PROBLEMS REPLY : YES */
*AND THE LESSON WILL CONTINUE */
5 READI5,101,END=3091CARD
CALL CRUNCH(LENGTH)
 IF(CARDP(1).EQ.*ASTRASK1GO TO 5
 IF(CARDP1).EQ.*ALPHA(25)1GO TO 10
 CALL TEST
310 WRITE(6,110)
110 FORMAT(0);REPLY WITH THE ANSWER TO THE FIRST PROBLEM */
6 READS,101,END=3101CARD
 CALL CRUNCH(LENGTH)
 IF(CARDP1).EQ.*ASTRASK1GO TO 6
 IF(CARDP1).EQ.*DIGIT(4)1GO TO 7
 WRITE(6,111)
111 FORMAT(0); YOUR ANSWER IS WRONG * YOUR PRINT STATEMENT SHOULD */
* HAVE BEEN PRINT ((5**2)-((4**2)*2)**5) /*
7 WRITE(6,112)
112 FORMAT(0); REPLY WITH THE ANSWER TO THE SECOND PROBLEM */
8 READS,101,END=71CARD
 CALL CRUNCH(LENGTH)
 IF(CARDP1).NE.*DIGIT(2)AND.CARDP2.NE.*DIGIT(3)1GO TO 9
 IF(CARDP3).EQ.*DECIMAL AND.CARDP4.EQ.*DIGIT(4)1GO TO 10
9 WRITE(6,113)
113 FORMAT(0); YOUR ANSWER IS WRONG * YOUR PRINT STATEMENT SHOULD */
* HAVE BEEN PRINT (3.1416*2.50**3)*3.03/12 /*
10 WRITE(6,114)
114 FORMAT(*.LES01150
*THE READ STATEMENT IS THE METHOD WHICH PROVIDES INPUT TO THE /* LES01160
*PROGRAM * THE FORM OF THE READ STATEMENT IS : */LES01170
*READ <**VARIABLE, **VARIABLE >> */LES01180
*FOR EVERY VARIABLE IN THE READ LIST THERE MUST BE A CORRESPONDING
*/ ELEMENT IN A DATA STATEMENT * THE READ AND DATA STATEMENTS ARE LES01200
*/ USED TOGETHER TO ASSIGN INPUT VALUES TO PROGRAM VARIABLES **X
*/ WHEN THE READ STATEMENT IS EXECUTED * EACH VARIABLE IS ASSIGN LES01220
*/ SUCCESSIVE NUMBERS FROM A STACK OF NUMERIC DATA OR SUCCESSIVELES01230
*/ STRINGS** FROM A STACK OF ALPHA-NUMERIC DATA * AS EACH VARIABLELES01240
*/ IS READ , IT TAKES THE TOP ELEMENT OF THE APPROPRIATE DATA */LES01250
* STACK */LES01260
READ(5,101,END=315)CARD

315 WRITE(6,115)LES01270
115 FORMAT(1X,'D DATA
*/' 'THE DATA STATEMENT IS A LIST OF INPUT NUMBERS OR "STRINGS" THAT
* / WILL BE ASSIGNED TO VARIABLES IN A READ STATEMENT. THE FORM
** OF THE DATA STATEMENT IS:
** DATA <= NUMBER OR "STRING":, NUMBER OR "STRING" >> */
** THE "STRING" OF ALPHA-NUMERIC CHARACTERS MUST BE ENCLOSED IN
** DOUBLE QUOTES "". 
** DATA STATEMENTS MAY BE PLACED ANYWHERE IN A PROGRAM, BUT THERE
** IS AN UPPER LIMIT OF 50C NUMERIC AND 50C ALPHA-NUMERIC DATA */
** ELEMENTS FOR EACH PROGRAM */
READ(5,101,END=316)CARD

316 WRITE(6,116)LES01430
116 FORMAT(1X,'WHEN THE FIRST DATA STATEMENT IS INTERPRETED */
* / BY THE CASE BIAS COMPILER A FIRST IN, FIRST OUT STACK IS FORMED/*
* / FOR NUMERIC AND ALPHA-NUMERIC DATA AS EACH NUMBER OR STRING
*/
* / IN A DATA LIST IS INTERPRETED, IT IS PLACED ON THE BOTTOM
*/
* / OF ITS RESPECTIVE DATA STACK AS OTHER DATA STATEMENTS ARE LOCATED*/
* / IN THE PROGRAM. ITS ELEMENTS ARE PLACED ON THE BOTTOM OF THE
* / PROPER STACK. */10X,' Example : */' '5X,' DATA 10,0,13, 33,' J.E.S
*/
* / "MITH', 18'/5X,' DATA '2', X,' DDE'/5X,' */
* / PRODUCES THE FOLLOWING DATA STACKS : */' '5X,' "NUMERIC',10X,' ALPH
* / "ERIC'/5X,' 10,0,10X,' J.E.S.MITH'/5X,' 13,33',10X,' Z',4, DDE'/5X,' 
* / 18'/5X,' 19'/
READ(5,101,END=317)CARD

317 WRITE(6,117)LES01530
117 FORMAT(1X,'DURING EXECUTION OF READ STATEMENTS, AS THE */
* / VARIABLES ARE TO BE TREATED AS THE TOP OF THE
*/
* / THE APPROPRIATE DATA STACK THE DATA STACK IS DECREMENTED AND
* / THE NEXT ELEMENT POPS UP */10X,' Example : */' '5X,' READ A,B1,LES01570
* / ASSUMING THAT THE DATA FROM EXAMPLE 1.1 IS AVAILABLE, THE
*/
*/ VARIABLES IN THE READ LIST ARE ASIGNED VALUES AS FOLLOWS */' '5X,' L
* / 'A <-- 10,0'/5X,' B1 <-- 13,33'/5X,' Z$ <-- J.E.S.MITH */
*/
* / THE RESULTING DATA STACKS ARE AS FOLLOWS : */' '5X,' "NUMERIC',10X,' AL
* / "PHAS' ) '/5X,' 10,0,15X,' Z',4, DDE'/5X,' 19'/
READ(5,101,END=318)CARD

318 WRITE(6,118)LES01630
118 FORMAT(1X,'E. RESTORE, RESTORES, STATEMENTS ARE USED TO RETURN THE
*/
* / NUMERIC ALPHANUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO*/
*/
* / THAT THE DATA MAY BE USED AGAIN. THE FORM OF THE RESTORE STATEMENTS
*/
* / NT" IS : */10X,' RESTORE ( RESTORES NUMERIC DATA ) */10X,' RESTORE
* / AND ALPHANUMERIC DATA ) */
*/
* / THE READ STATEMENTS, */ / CONSIDER THE FOLLOWING STATEMENTS AS */
*/
* / PART OF A BASIC PROGRAM : */' '5X,' DATA 2,5,7,16,'ANS'/5X,' DATA LES01730
** / "CORRECT", 25,' "WRONG", 0'/5X,' READ A,B3,1$,C1,D'/5X,' READ J
LES01740
WHAT IS THE VALUE OF C1? **REPLY WITH VALUE 1**


NOW THAT YOU HAVE SEEN HOW READ AND DATA STATEMENTS WORK TOGETHER, TO INPUT VALUES INTO YOUR PROGRAM, AND HOW THE PRINT STATEMENT IS USED TO OUTPUT RESULTS YOU HAVE THE FACILITY TO WRITE SIMPLE PROGRAMS USING INPUT DATA, FOR EXAMPLE:

**A=B**2-4*A*C, THEN BY SAYING PRINT A, Z, THE RESULTS OF THE EXPRESSIONS WOULD BE DISPLAYED. THIS WILL GIVE YOU GREATER FLEXIBILITY IN WRITING EXPRESSIONS, AND WILL ALLOW YOU TO DO ASSIGNMENTS SUCH AS:**

YOU WILL NOW BE GIVEN SOME REPRESENTATIVE PROBLEMS TO GIVE YOU A CHANCE TO EXERCISE YOUR NEW PROGRAMMING TOOLS.
*10X,'1.6* WRITE A PROGRAM TO SOLVE THE EQUATION: X**2+10Y-24 /* LES02240
*WHERE THE INPUT DATA IS X=16 ,Y=3 /*10X,'1.6* WRITE A PROGRAM LES02250
*5 TO SOLVE THE QUADRATIC EQUATION/*-B+(B**2-4*A*C)**1.5/2A /*10X,'1.6* WRITE A PROGRAM LES02270
*E THESE PROGRAMS NOW ONE REPLY: YES/* AND YOU WILL ENTER THE CAIBALESES02280
*5C COMPILER ; OTHERWISE REPLY: NO/* AND YES WILL GO ON TO THE NLE502290
*END LESSON */ /*10X,'1.6* WRITE (5,101,END=325)CARD
35 READ(5,101)END=325)
36 CALL CRUNCH(LENGTH)
37 IF(CARDP(1).EQ.ASTRSK) GO TO 35
38 IF(CARDP(1).EQ.ALPHA(25))GO TO 45
39 IF(CARDP(1).EQ.ALPHA(14))GO TO 40
40 WRITE(6,126)
41 FORMAT(*'*** YOUR REPLY IS INCORRECTLY TYPED ,REPLY 00:00:00***'
*')
42 WRITE(6,127)
43 FORMAT(*'*** IF YOU DECIDE TO RUN THESE PROBLEMS LATER THE ANSWERSLES02400
*'/ WILL BE GIVEN SO YOU MAY CHECK YOUR RESULTS */'/*5X,'5X****1 longevity
44 CALL EXIT
45 READ(6,128)
46 FORMAT(*'*** THE CORRECT ANSWER IS 106 AND THE PROGRAM SHOULD */
47 *HAPPEN SIMILAR TO */'/*5X,'READ X,Y'/5X,'PRINT**ANSWER=**'/*5X,'106'/*5X,'END**/*5X,'106
48 CALL EXIT
49 WRITE(6,129)
50 FORMAT(*'*** THE CORRECT ANSWER IS 50 AND THE PROGRAM SHOULD **/)
51 *LOOK SIMILAR TO */'/*5X,'READ A,B,C'/5X,'PRINT**ANSWER=-,-BLES02630
52 ** (8**2-4*A*C)**1.5/2*A'/5X,'DATA 2,5,2'/5X,'END**/LES02640
53 CALL EXIT
54 LESSON 3 PRESENTS THE 'LET' STATEMENT AND BUILT-IN FUNCTIONS
55 COMMON STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
56 LESS04010
57 LESS04020
IAPTR, INPTR, IADATA(500), XDATA(500), STRING(5),
DIGIT(10), IPRIT(10), LISTS(1000), LISTS(1000),
PRIT(2500), NERRS, INST, NSLIST, DEBUG, DOLSN, QUOTE,
EQUALS, PARR, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARL, ASTRSK, BLANK
COMMON INTERP,EXERR
REAL*B LE53/LESON3 */
CALL ALOAD(LE53,N1)
WRITE(6,100)
100 FORMAT('0',5X,'*** LESSON 3, ***
*THIS INSTRUCTION SET WILL INTRODUCE YOU TO ASSIGNMENT STATEMENTS*/
* AND BUILT-IN FUNCTIONS THE LET STATEMENT AND THE 10 BUILT IN*/
*FUNCTION WILL ENABLE YOU TO EVALUATE AND ASSIGN VARIABLES TO */
*COMPLEX ARITHMETIC EXPRESSIONS. */
READ(5,101)END=302)CARD
101 FORMAT(80A1)
302 WRITE(6,102)
102 FORMAT('0',1X,'A, LET
*THE LET STATEMENT IS AN ASSIGNMENT OR SUBSTITUTION COMMAND. IT*/
*CAUSES THE EVALUATION OF AN EXPRESSION TO BE SUBSTITUTED FOR THE*/
*CURRENT VALUE OF A VARIABLE THE FORM OF THE LET STATEMENT IS :*/
*10X, LET <VARIABLE >> = < EXPRESSION >>
*OR
*LET <VARIABLE >> = <VARIABLE >> = < EXPRESSION >> */
READ(5,101)END=303)CARD
303 WRITE(6,103)
103 FORMAT('0',5X,'THERE MAY BE ANY NUMBER OF VARIABLE = VARIABLE*/
*IN THE FORM OF THE LET STATEMENT AN EXPRESSION IS A NUMBER */
*VARIABLE OR AN ARITHMETIC EXPRESSION. */
*WHEN THE LET STATEMENT IS EXECUTED THE EXPRESSION ON THE RIGHT*/
*SIDE OF THE EQUAL SIGN IS EVALUATED AND THE RESULTING VALUE */
*ASSIGNED TO THE VARIABLE OR VARIABLES ON THE LEFT SIDE OF THE EQUAL*/
*SIGN. THE PREVIOUS VALUE ASSIGNED TO THE VARIABLE OR VARIABLE*/
*IS LOST FOR EXAMPLE : */10X, LET A=12,3*/10X, LET A=16,4*/10X, LET A=B=25 */
*THE VALUE OF THE VARIABLES A AND B IS */
*NOW A=B=25 */
READ(5,101)END=304)CARD
304 WRITE(6,104)
104 FORMAT('0',1X,'THERE IS A ONLY RESTRICTION ON THE USE OF VARIABLES */
*IS THAT SIMPLE AND SUBSCRIPTED VARIABLES CAN ONLY BE ASSIGNED*/
*NUMERIC VALUES AND ALPHA VARIABLES CAN ONLY BE ASSIGNED ALPHA*/
*NUMERIC "STRINGS", FOR EXAMPLE : */10X, LET A=D4*X(5)*/3**2*/
*6/1*/10X, LET AS=XS=HELP*/10X, LET Y=(B - 4*A*C)**5*/10X*/
*LET OS=ANSWER */
*THE FOLLOWING ASSIGNMENT IS ILLEGAL : */10X*/
*LET A=AS=(3**4) */
READ(5,101)END=305)CARD
305 WRITE(6,105)
105 FORMAT('0',1X,'YOU WILL NOW BE ASKED QUESTIONS CONCERNING */
*WHAT YOU HAVE JUST LEARNED */
*/
106 FORMAT(10,'10X,1.) REPLY WITH VALUE OF X IN BELOW PROGRAM */
*#1X* Let A=**1/10X,* LET X=A**2 - 4*B
* + 3*C /10X,* PRINT "ANSWER = "X'/10X,* END /*
5 READ(5,101,END=306) CARD
IF(CARDP(1).EQ.ASTRSK) GO TO 5
IF(CARDP(1).EQ.DIGIT(2)) GO TO 10
IF(CARDP(1).EQ.PLUS.AND.CARDP(2).EQ.DIGIT(2)) GO TO 10
306 WRITE(6,107)
107 FORMAT('""" YOUR RESPONSE WAS INCORRECT. THE EXPRESSION TO */
* EVALUATED IS: X=4**2 - 4*6 + 3*3 WHICH EVALUATES AS :15X,* X < */
* -- 16 - 24 + 9 ; X <- + 1/*)
10 WRITE(6,108)
108 FORMAT('"""10X,* 2.) REPLY WITH VALUE OF Z IN BELOW PROGRAM */
*#1X,* DATA 1.2,3.4,6.7/10X,* DATA 'RIGHT ON',7.8/10X,* READ A,B,X */
*#1X,* READ G*,Y,C,D,'10X,* LET Z=X+Y/10X,* END /*
15 READ(5,101,END=309) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 15
IF(CARDP(1).EQ.DIGIT(8)) GO TO 20
309 WRITE(6,109)
109 FORMAT('"""10X,* YOUR RESPONSE WAS INCORRECT. THE VARIABLES ARE */
*ASSIGNED VALUES AS FOLLOWS :/'5X,* "NUMERIC"/10X,* "ALPHA"/5X,* AK */
*-1,'1CX,* D<=-7/'5X,* B<=-2/'5X,* X<=-3/'5X,* Y<=-4 */
*#5X,* C<=-6/'5X,* D<=-7/'5X,* X<=-4/'5X,* Y<=-5 */
20 WRITE(6,110)
110 FORMAT('"""10X,* ICX,* 3,) LET R= A+B/C-D/* WHICH FORMULA DOES THIS */
* STATEMENT REPRESENT ?*/10X,* REPLY WITH CORRECT LETTER/10X,* A. */
* R=(A+B/(C-D))/10X,* B.) R= A+B/C-D/10X)
25 READ(5,121,END=311) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 25
IF(CARDP(1).EQ.ALPHA(21)) GO TO 30
311 WRITE(6,111)
111 FORMAT('"""10X,* YOUR RESPONSE WAS INCORRECT. WHEN THERE ARE NO */
* PARENTHESES IN AN EXPRESSION, THE HIERARCHY OF OPERATORS APPLIES */
* THIS IN THIS EXPRESSION THE DIVIDE OPERATION IS DONE FIRST; */
* THEN ADDITION AND SUBTRACTION. IF YOU ARE STILL HAVING PROBLEMS, */
* WITH EXPRESSIONS YOU HAD BETTER REVIEW YOUR SESSION ON LESSON 1.
*"""
30 WRITE(6,112)
112 FORMAT('"""10X,* B. BUILT-IN FUNCTIONS */
* BUILT-IN FUNCTIONS ARE COMMONLY USED PROGRAMS ALREADY WRITTEN */
* AND STORED IN THE CBASIC COMPILER FOR YOUR USE. THERE ARE */
* FUNCTIONS TO FIND SQUARE ROOTS, LOGARITHMS, ABSOLUTE VALUES, */
* AND TRIGONOMETRIC VALUES. THE FORM FOR THE BUILT-IN FUNCTIONS IS */
* "/:10X,* FUNCTION NAME << (EXPRESSION) >>"/* WHERE THE EXPRESSION
50 READ(5,101)END=319)CARD
IF(CARDP(1),EQ.'ASTR]) GO TO 50

319 WRITE(6,119)
119 FORMAT('NO THE SQUARE ROOT OF A NEGATIVE NUMBER IS AN **
*UNDEFINED OPERATION * IN THE NEXT LESSON YOU WILL BE SHOW A */
*BASIC STATEMENT FOR TESTING AND BRANCHING TO ANOTHER SEGMENT */
*OF THE PROGRAM IF THE TEST IS TRUE * FOR EXAMPLE THE ABOVE */
*PROGRAM SEQUENCE MIGHT BE ALTERED AS FOLLOWS */5X,* LET B=-9**5X */
* IF B LT 0 THEN 100/5X,* 5C LET X=SQR(X)**1X,* ... */10X,* ... */
*/10X,* */5X,* 10C REM NEGATIVE ARGUMENT*/5X,* LET B=ABS(B)**5X */
* GO TO 5C */CX,* ... */10X,* ... */ THIS PROGRAM SEQUENCE TESTS */
*FOR A NEGATIVE ARGUMENT * IF B LT */ IT BRANCHES TO STATEMENT NUMB */
*ER 100 * MAKES THE ARGUMENT POSITIVE/" AND BRANCHES BACK TO STATE */
*MENT 5C TO COMPLETE THE PROGRAM */"
READ(5,101)END=320)CARD

320 WRITE(6,123)
123 FORMAT('10X,* C. SUMMARY */
*/ WITH THE LET STATEMENT AND BUILT-IN FUNCTIONS * PLUS THE PREVIOUS */
*/ BASIC STATEMENTS ( REM * READ * DATA * PRINT ) * YOU ARE */
*/ FAST GAINING AN EFFECTIVE REPertoire FOR PROGRAMMING USE * IN */
*/ THE NEXT LESSON YOU WILL LEARN HOW TO SET UP LOOPS IN A PROGRAM */
*/ SO THAT THE MAIN BODY OF A PROGRAM MAY BE EXECUTED AS OFTEN */
*/ AS DESIRED * AS YOU ARE DOING YOUR REVIEW PROBLEMS * THINK ABOUT */
*/ HOW YOU COULD SET UP A LOOP TO READ IN ANY AMOUNT OF DATA */
*/ PROCESS IT AND THEN HALT FOR SOME TEST CONDITION */"
READ(5,101)END=324)CARD

324 WRITE(6,124)
124 FORMAT('16X,* THE FOLLOWING REVIEW PROBLEMS WILL EXERCISE */
*/ YOUR PROGRAMMING SKILLS TO DATE */1X,* 1.) WRITE A PROGRAM TO COG */
*/ COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR */ SOME NUMBER OF YEA */
*/ RS HENCE * THE FORMULA IS */10X,* P=S/(1/(1+1)**N))*/ WHERE P */
*/ IS THE PRESENT WORTH OF AN AMOUNT S IN N YEARS HENCE */ AT AN */
*/ INTEREST RATE OF I */ FOR DATA USE I=3.38 * S=5000 * N=20 */5X; */
*/ 2.) WRITE A PROGRAM TO FIND SIDES A, AND C OF A TRIANGLE USING */
*/ THE LAWS FORMULA : */10X,* A/SIN(A) = B/SIN(B) = C/SIN(C */
*/ WHERE THE NUMERATOR IS THE SIDE AND THE DENOMINATOR IS THE */
*/ SIDE OF THE ANGLES */ THE CONVERSION FACTOR FROM DEGREES TO RADIANS */
*/ IS */10X,* 1 DEG!=3.1416/180) RADIANS */ THE DATA FOR THE PROGR */
*/ AM IS */5X,* S=3.5X * A=30.71 DEG. */5X,* ANGL */
*/ E B=49.97 DEG. */5X,* ANGLE C=31.32 DEG. */7/) IF YOU WISH TO RUN THE */
*/ CALL CRUNCH(LNGTH) */1X,* IF(CARDP(1),EQ.'ASTR)) GO TO 60 */
*/ IF(CARDP(1),EQ.'ALPHA(14)) GO TO 65 */
*/ IF(CARDP(1),EQ.'ALPHA(25)) GO TO 70 */
325 WRITE(6,125)
125 FORMAT('0,* YOUR 35 GO TO 60
65 WRITE(6,126)
126 FORMAT('0', * THE ANSWERS TO THE PROBLEMS WILL NOW BE GIVEN SO '/
* THAT YOU MAY CHECK YOUR RESULTS LATER :''/5X,'1.5) $1072.74/5X,'2
* ) SIDE A=45.06 IN. AND SIDE C=23.69 IN.//'
CALL EXIT
70 CALL TEST
WRITE(*,127)
127 FORMAT('0', 'ICX, * REPLY WITH ANSWER TO QUESTION 1.0 */)
75 READ(5,101,END=80) CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ. ASTRSK) GO TO 75
IF(CARDP(1).NE.DIGIT(2) OR CARDP(4).NE.DIGIT(3)) GO TO 80
IF(CARDP(5).EQ.DEcimal AND CARDP(6).EQ.DIGIT(8)) GO TO 90
80 WRITE(*,128)
128 FORMAT('0', 'ICX, * THE CORRECT ANSWER IS $1075.74 AND THE PROGRAM'/
* SHOULD HAVE BEEN SIMILAR TO :''/5X,'_ (*((1+!)**N))/5X,' PRINT*PRESENT WORTH='',P'/5X' DATA 5000,,08,20'
*5X,' END//'
90 WRITE(*,129)
129 FORMAT('0', 'ICX, * REPLY WITH ANSWER TO SIDE A FOR QUESTION 2.0 */)
91 READ(5,101,END=95) CARD
IF(CARDP(1).EQ. ASTRSK) GO TO 91
IF(CARDP(1).NE.DIGIT(5) OR CARDP(2).NE.DIGIT(6)) GO TO 95
IF(CARDP(3).EQ.DEcimal AND CARDP(4).EQ.DIGIT(1)) GO TO 97
95 WRITE(6,130)
130 FORMAT('0', 'ICX, * THE CORRECT ANSWER IS SIDE A=45.06 IN. AND /
* SIDE C=23.69 IN. THE PROGRAM SHOULD HAVE BEEN SIMILAR TO :''/5X,' *
* READ A1,'01/5X,' LET A1=(B1*IN(A1*3,1416/180))/SIN(B1*3,1416/180)*/
*5X,' PRINT*SIDE A='',A1,**SIDE C='',C1'/5X,' DATA 98.71,49.97,
*31.32,34.91/5X,' END//'
97 CALL EXIT
END

THIS LESSON INTRODUCES BRANCHING, BOTH CONDITIONAL AND UNCONDITIONAL.

COMMON
- STACK(100), PROG(250), CARD(80), CARDP(80), ALPHA(48), LEOO0030
- IAPTR, INPTR, IADATA(500), XDATA(500), STRING(50), LEO00040
- DIGIT(10), IPRRT(10), LISTT(100), ISTRT(100), LEO00050
- PRT(2500), NERRS, INST, NLIST, DEBUG, DLSGN, QUOTE, LEO00060
- EQUALS, PARRT, DECIMAL, PLUS, CMNUS, SLASH, COMMA, LEO00070
- PARLFT, ASTRSK, BLANK, LEO00080
COMMON INTERP, EXERR
REAL*8 LESS4*LESON, LEO00090
CALL LOAD(LESS4,N1)
WRITE(*,100)
100 FORMAT('0', '4, ** LESSON 4 *** ',100)
   }
**THIS INSTRUCTION SEQUENCE WILL COVER BRANCHES** • AS YOU HAVE

**SEEN FROM PREVIOUS PROGRAMS** • A PROGRAM'S EXECUTION USUALLY

**TAKES A DIFFERENT ROUTE FROM THE FIRST TO THE LAST STATEMENT**

**THE CONDITION THAT ALLOWS DETOURS TO OCCUR IN PROGRAMS IS CALLED**

**BRANCHING** • THERE ARE TWO TYPES OF BRANCHING: UNCONDITIONAL

**AND CONDITIONAL**•)

READ(5, 101, END=321 CARD

101 FORMAT(80A1)

102 WRITE(6, 102) • 'THAT IS: */5X,' A UNCONDITIONAL BRANCHES

*/5X,' • AN UNCONDITIONAL BRANCH IS AN IMPERATIVE TRANSFER OF CONTROL

*/5X,' • FROM ONE POINT IN A PROGRAM TO ANOTHER • THERE ARE TWO FORMS

*/5X,' • OF THE UNCONDITIONAL BRANCH • 'GO TO' AND THE 'COMPUTED '

*/5X,' • GO TO' */IX,' 1') GO TO '< STATEMENT NUMBER >' */IX,' THIS COMMAND LESCO250

*/5X,' • TRANSFERS PROGRAM CONTROL DIRECTLY TO THE STATEMENT NUMBER */IX,' AND LESCO270

*/5X,' • CONTINUES EXECUTION FROM THAT \* THE 'GO TO' IS USED */IX,' FOR LESCO280

*/5X,' • FORMING LOOPS IN A PROGRAM */IX,' READ(5, 101, END=3031 CARD

103 WRITE(6, 103) • 'A SAMPLE LOOPfollows: */5X,' REM PROGRAM TO COLES322

*/5X,' • PUTERESENTWORTH/*/5X,' REM P=INVESTMENT, S=PRINCIPAL, 1=INTEREST LESO330

*/5X,' • RATE,N,NR,YEARS/*/5X,' PRINT'INVESTMENT','PRINCIPAL','INTEREST' LESO340

*/5X,' • ** N=N, YEARS/*/5X,' 10 READ S, I, N/*/5X,' LET P=S/(1/(1+I)**N) LESOC350

*/5X,' • 6X,' PRINT P, S, I, N/*/5X,' GO TO 10/*/5X,' DATA 5000, 08, 20, 5600, 08, 10 LESO360

*/5X,' • 500, 06, 10/*/5X,' END */IX,' PRODUCES THE OUTPUT */IX,' INVESTMENT */IX,' LESO370

*/5X,' • X, PRINCIPAL */IX,' INTEREST */IX,' N, YEARS */IX,' 1072.74 */IX,' 16X,' LESO380

*/5X,' • 5000 */IX,' 16X,' 2X, */IX,' 231.49 */IX,' 15X,' LESO390

*/5X,' • 5000 */IX,' 16X,' 10 */IX,' 15X,' LESO340

*/5X,' • 10 */IX,' 2X, */IX,' 291.69 */IX,' 15X,' 24C */IX,' LESO340

*/5X,' • 10 */IX,' 16X, */IX,' 15X,' 10 */IX,' */IX,' ERROLESO340

*/5X,' • R, YOU TRIED TO READ MORE NUMERIC DATA THAN YOU PUT IN */IX,' LESO410

*/5X,' • READ(5, 101, END=3041 CARD

104 WRITE(6, 104) • 'THE ERROR OCCURS BECAUSE YOU RUN OUT OF DATA DURING*/IX,' COLES420

*/IX,' • THE EXECUTION OF THE LOOP SET-UP BY THE UNCONDITIONAL TRANSFER

*/IX,' • IF THE READ STATEMENT WERE NOT IN THE LOOP TO CAUSE THE

*/IX,' • PROGRAM TO STOP, THEN YOU WOULD BE IN AN 'INFINITE LOOP' */IX,' LESO430

*/IX,' • A CONDITION IN WHICH THERE IS NO WAY TO STOP • YOU MUST ALWAYS */IX,' COLES450

*/IX,' • CHECK FOR THE 'INFINITE LOOP' CONDITION BY MAKING SURE THAT */IX,' LESO460

*/IX,' • YOUR PROGRAM HAS AN EXIT */IX,' READ(5, 101, END=3051 CARD

105 WRITE(6, 105) ON '< EXPRESSION >' GO TO '< STATEMENT NUMBER >' */IX,' THIS SPECIAL FORM OF THE '/IX,' LESO470

*/IX,' • COMMAND */IX,' IS CALLED THE 'COMPUTED GO TO', THE 'EXPRESSION' */IX,' LESO480

*/IX,' • IN THE FORM OF THE STATEMENT MUST EVALUATE TO AN INTEGER NUMBER */IX,' LESO490

*/IX,' • BETWEEN 1->9999 • IF IT IS NOT AN INTEGER, OR OUTSIDE THIS */IX,' LESO500

*/IX,' • RANGE AN ERROR WILL OCCUR */IX,' WHEN THE 'COMPUTED GO TO' IS LESO510

*/IX,' • EXECUTED THE EXPRESSION IS EVALUATED */IX,' AND PROGRAM CONTROL TRANSLESO520

*/IX,' • FERS TO THE N-TH STATEMENT NUMBER */IX,' WHERE N-TH REPRESENTS THE LESO530
*VALUE OF THE EXPRESSION * FOR EXAMPLE :*/5X* LET I=3*/5X * ON I LEO0610
*GO TO 100,33,475,9999*1/)
READ(5,101) END=3061CARD

306 WRITE(6,106)
106 FORMAT(0*) * EXECUTION OF THE **COMPUTED GO TO** WOULD CAUSE */
*PROGRAM CONTROL TO TRANSFER UNCONDITIONALLY TO STATEMENT NUMBER 47 LEO0C660
*5 */5X* YOU MUST BE CAREFUL WHEN USING THE **COMPUTED GO TO** */
*NOT ONLY BECAUSE OF INFINITE LOOPS BUT BECAUSE THE EXPRESSION LEO0660
* MUST BE AN INTEGER BETWEEN 1->9999 AND THERE MUST BE A */
*STATEMENT NUMBER FOR **ALL** POSSIBLE VALUES OF THE EXPRESSION */
*/)
READ(5,101) END=3771CARD

307 WRITE(6,107)
107 FORMAT(0*) * 9. CONDITIONAL BRANCHING */5X* LEO0740
*THE CONDITIONAL BRANCH TRANSFERS CONTROL ONLY IF CERTAIN */
*RELATIONS ARE TRUE * IF THE THE TEST OF RELATIONS IS TRUE THEN*/ LEO0760
*TRANSFER OF CONTROL OCCURS OTHERWISE PROGRAM CONTROL CONTINUES LEO0C770
** WITH THE NEXT STATEMENT * THE FORM OF THE CONDITIONAL BRANCH IS LEO0780
*/ ** 1X * IF << EXPRESSION >> ** RELATION >> ** EXPRESSION >> THEN */
** STATEMENT NUMBER >> */ ** NOTE THAT ALPHA VARIABLES ARE NOT ALLOWED LEO0800
*EO AS AN EXPRESSION IN A/** CONDITIONAL BRANCH */**
READ(5,101) END=3431CARD

308 WRITE(6,108)
108 FORMAT(0*) * THE RELATIONS ARE : */** LEO0820
*SYMBOLS* 2X* EXAMPLE* 5X* MEANING */** LEO0840
*/*//2X* GT* 5X* A GT B//2X* A GREATER THAN B//2X* GE* 5X* A GE B LEO0860
*B*//5X* A GREATER THAN OR EQUAL TO B//5X* LT* 5X* A LT B//5X* LEO0870
*A B LE A//2X* B//2X* LET A//2X* LE B//2X* A LESS THAN OR EQUAL TO LEO0880
*B//5X* NE* 5X* A NOT EQUAL TO B//5X* A= B LEO0890
*16X* A EQUAL B//5X*/**
READ(5,101) END=3991CARD

309 WRITE(6,109)
109 FORMAT(0*) * WHEN THE CONDITIONAL STATEMENT IS EXECUTED LEO0920
* (EXPRESSION RELATION EXPRESSION ) IS TESTED AND IF THE RELATION LEO0940
** TRUE THEN THE STATEMENT NUMBER LEO0C950
** OTHERWISE PROGRAM CONTROL CONTINUES TO THE NEXT SEQUENTIAL */** LEO0960
** STATEMENT FOR EXAMPLE : */5X* I READ A*/5X* *** */5X* *** /*5X* LEO0C970
** IF A=5 READ X=50(A)//5X** IF X=50 X=90 LEO0C980
** PRINT** ILLEGAL ARGUMENT**A*/5X** GO TO */5X** THE ONLY LEO0990
**IME THAT THE CONDITIONAL BRANCH IS EXECUTED IS WHEN** A IS LESS LEO01000
*MAN 0.* */**
READ(5,101) END=3101CARD

310 WRITE(6,110)
110 FORMAT(0*) * YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT BRANCHING LEO01040
**//0//15X* 1 GO TO END*//** LEO01060
5 READ(5,101) END=3111CARD
CALL CRUNCH(LNGTH) LEO01070
LEO01080
* LOOPS AND ARE AS FOLLOWS :/*//
* READ(5,101) END=320) CARD
320 WRITE(6,120)
120 FORMAT('0*5X,1*COUNT AND TEST METHOD, IN WHICH A COUNTER */', LESO1600)
*IS INCREMENTED IN THE LOOP AND WHEN THE COUNTER REACHES A */', LESO1610)
*CERTAIN VALUE BRANCH OUT OF THE LOOP * EXAMPLE PROGRAM */10X, * LESO1620
*REM COUNT AND TEST METHOD/10X, * REM N IS COUNTER, INITIALIZED TO LESO1630
*0, AND COUNTS FROM 1-->=10'1/10X,' LET N=Z=0/1/10X,' READ X'/10X,' 1LESO1640
*U LET Z=I+N'10X,' LET N=N+1'/10X,' IF N GT 10 THEN 100'1/10X,' GO TO 1LESO1650
* TO 10'1/10X,' 100 PRINT*SUM='*,Z'/10X,' DATA 10'1/10X,' END//// RELESO1660
*PLY WITH VALUE CF Z,'*/
45 READ(5,101) END=50) CARD
CALL CRUNCH(116) GO TO 45
IF(CARDP(11),EQ.'ASTRISK) GO TO 45
IF(CARDP(11),NE.'DIGIT12) OR CARDP(2),NE.'DIGIT2) GO TO 50
IF(CARDP(3),EQ.'DIGIT(1) GO TO 60
50 WRITE(6.121)
121 FORMAT('0*1') YOUR ANSWER IS INCORRECT * THE ONLY WAY TO BE */,
*SURE OF VALUES IN A LOOP IS TO KEEP A TABLE OF THE VARIABLES* //LESO1750
* AND KEEP TRACK OF THESE VALUES IN THE LOOP */5X,'X'=5X,'N'=5X,'//LESO1760
*/10X,'11,5X,'10X,'21,5X,'20,1/10X,'31,5X,'20,1/10X,'41,5X,'20,1/10X,'51,5X,'20,1/10X,'61,5X,'20,1/10X,'71,5X,'20,1/10X,'81,5X,'20,1/10X,'91,5X,'20,1/10X,'101,4X,'3LESO1783
*1,4X,'110X,' /TUS Z=110'//)
60 WRITE(6.122)
122 FORMAT('0*1) EX,* 2. READ AND TEST METHOD, IN WHICH A VALUE IS */
*READ IN AND TESTED FOR THE END OF LOOP CONDITION */1/10X,' *X=1/10X,' IF X=9999 THEN 9999'1/10X,' LET Z=LEY1840
*Z/X'10X,' GO TO 1/10X,' DATA 1,2,3,4,5,6,9999'/10X,' 9999 PRINT */
*SUM='*,Z'/10X,' END//// REPVL WITH VALUE OF Z//'//)
65 READ(5,101) END=320) CARD
CALL CRUNCH(116) GO TO 65
IF(CARDP(11),EQ.'ASTRISK) GO TO 65
IF(CARDP(11),NE.'DIGIT13) AND CARDP(2).EQ.'DIGIT(2) GO TO 70
323 WRITE(6.123)
123 FORMAT('0*1) YOUR ANSWER IS INCORRECT * VARIABLE VALUES IN THE*/
*LOOP ARE */5X,'X'=5X,'Z'=5X,'11,5X,'15,5X,'21,5X,'31,5X,'31,5X,'6LESO1930
*/5X,'4,5X,'10*/5X,'5X,'15,5X,'6,5X,'21,5X,'9999//' /TUS Z=LESO1940
*21'//)
70 WRITE(6.124)
124 FORMAT('0*1) IOX,' C. SUMMARY
*YOU HAVE SEEN HOW THE UNCONDITIONAL BRANCHES **GO TO** AND */,
*COMPUTED GO TO** TRANSFER PROGRAM CONTROL * AND HOW THE */
*CONDITIONAL BRANCH (*IF/THEN*) TESTS FOR TRANSFER OF PROGRAM*/
*CONTROL * AND YOU HAVE OBSERVED THE CONTROL OF LOOPS SO THAT */,
*A PROGRAM SEGMENT MAY BE REPEATED UNTIL A SPECIFIED CONDITION */,
*IS MET. IN THE NEXT LESSON YOU WILL LEARN A BASIC STATEMENT TO*/,
*CONTROL A LOOP BY THE INCREASE AND TEST METHOD */ */ THE FOLLOWINGLESO2040
*G PROBLEMS WILL TEST YOUR NEW SKILLS :/*/ READ(5,101,END=325)
325 WRITE(6,125) 125 FORMAT(1X,' 1.') WRITE A PROGRAM TO COUNT THE NUMBERS
* BETWEEN 50 AND 60 ; AND ALSO PRINT THEM OUT .THE INPUT DATA */
* IS 10,50,35,75,62,60,54,54 ,'/ 5X,' Z. WRITE A PROGRAM TO COMPUT
* THE PRESENT WORTH OF AN INVESTMENT */ FOR SOME YEARS HENCE
* AT VARYING INTEREST RATES . THE FORMULA IS: */ 5X,' P=S(1/(1+I)**N)**LES02120
** WHERE P=PRESENT WORTH,S=PRINCIPAL,I=INTEREST,AND N=NR. OF YE
** ARS */ FOR DATA USE S=500G,AND I=0.04-->5X, IN INCREMENTS OF .01 ,
* AND N=20 */ LES02150
*/ IF YOU DESIRE TO EXECUTE THESE PROGRAMS NOW */ REPLY: YES ; OTH
*/ ERWISE REPLY : NO */
80 READ(5,101,END=326)CARD
CALL CRUNCH(LIGTH)
1F(CARDP(1).EQ.ASTRSK) GO TO 80
1F(CARDP(1).EQ.ALPHA(251)) GO TO 80
326 CALL EXIT
65 CALL TEST1
127 FORMAT(1X,' IF YOU DESIRE TO SEE THE SOLUTION TO THE PROBLEMS */
* REPLY : YES , OTHERWISE REPLY : NO AND GO ON TO THE NEXT LESSON */
**
90 READ(5,101,END=325)CARD
CALL CRUNCH(LIGTH)
1F(CARDP(1).EQ.ASTRSK) GO TO 90
1F(CARDP(1).EQ.ALPHA(141)) CALL EXIT
328 WRITE(6,128)
128 FORMAT('G',10X,' PROBLEM 1',1/5X,' REM SORT AND COUNT PROBLEM */ 5X LES02330
** * REM C= COUNTER, N=NUMBER /5X , PRINT NUMBERS IN RANGE 50-->260 */ LES02340
** 5X = NUMBER IN RANGE 50-260 */LES02350
** N GT 60 THEN 1/5X, IF N LT 50 THEN 1/5X, LET C=C+1',5X,' PRINTLES02360
** T N/5X, GO TO 1/5X, DATA IC,50,35,75,56,62,60,54,54,5X,9999 PRLES02370
** INT C, NUMBERS IN RANGE 5G-->CO */5X,' END */10X,' PROBLEM 2 */ LES02380
** 5X, REM COMPUTE PRESENT WORTH OF AN INVESTMENT */5X, REM P=PRESLEN02390
** T WORTH=S=PRINCIPAL,N=NR , YEARS=VARIABLE INTERES*/5X, PRINT*/INTE02400
** REST */5X,' INVESTMENT */5X, 'PRINCIPAL*/5X, YEARS */5X , READLES02410
** S=N/5X, LET I=1 '5X, ' IF I GT .68 THEN 20C */5X, ' LET P=P+S */1(1/LES02420
** (1+N)**N))'5X, ' PRINT I,P,S,N/5X, ' LET I=I+1 '5X, ' GO TO 1/5X LES02430
** DATA 500G,20/5X,' 20C END */
CALL EXIT END
THIS LESSON INTRODUCES ITERATION , SUBSCRIPTED VARIABLES , AND
LISTS(VECTORS) AND TABLES(MATRICES)
COMMON
- STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48), LES02020
- LES00030
100 FORMAT(6,5X)  *** LESSON 5 ***
*THIS LESSON WILL INTRODUCE YOU TO ITERATION, SUBSCRIPTED
*VARIABLES, AND LISTS (VECTORS) AND TABLES (MATRICES).  */5X;  LES00120
*A.  ITERATION (LOOPING)  */5X;  LES00140
*IN THE LAST LESSON YOU WERE SHOWN HOW TO USE CONDITIONAL AND  */5X;  LES00150
*UNCONDITIONAL BRANCHES TO CONTROL LOOPING.  THE COUNT AND TEST  */5X;  LES00160
*ITERATIVE LOOP OCCURS SO FREQUENTLY THAT AN ABBREVIATED BASIC  */5X;  LES00170
*STATEMENT HAS BEEN DEVISED TO CONTROL LOOPING.  THE ITERATIVE  */5X;  LES00180
*LOOP HAS THE FOLLOWING FORM:  */5X;  LES00190
*FOR << SIMPLE VARIABLE >> = << EXPRESSION >> TO << EXPRESSION >>/*/5X;  LES00195
*2X;  STEP << EXPRESSION >>/12X;  */5X;  LES00197
*/5X;  NEXT << SIMPLE VARIABLE >>  */5X;  LES00199
*READ(5,101), END=3C21 CARD  LES00230
301 WRITE(6,102)  LES00250
102 FORMAT(6,4X), FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT:  */5X;  LES00260
*FOR I=1 TO 10 STEP 2/5X;  LET X=I/5X;  NEXT I/5X;  LES00270
*THE SIMPLE VARIABLE FOLLOWING "FOR" IS THE LOOP INDEX.  WHEN  */5X;  LES00280
*THE FOR/NEXT PAIR IS EXECUTED, THE LOOP INDEX IS GIVEN THE  */5X;  LES00290
*VALUE (INITIALIZED) OF THE FIRST EXPRESSION I=1 IN EXAMPLE).  */5X;  LES00300
*THIS INDEX IS THEN TESTED TO DETERMINE WHETHER IT IS GREATER THAN  */5X;  LES00310
*"**, THE SECOND EXPRESSION AFTER "TO" (10 IN EXAMPLE).  IF IT IS  */5X;  LES00320
*GREATER, CONTROL IS TRANSFERRED TO THE STATEMENT FOLLOWING  */5X;  LES00330
*"**, OTHERWISE THE REMAINING STATEMENTS WITHIN THE LOOP  */5X;  LES00340
*(FOR/NEXT) ARE EXECUTED SEQUENTIALLY UNTIL THE "NEXT" STATEMENT  */5X;  LES00350
*IS REACHED.  */5X;  LES00360
303 WRITE(6,103)  LES00370
103 FORMAT(6,4X), WHEN THE "NEXT" STATEMENT IS REACHED, THE LOOP  */5X;  LES00380
*INDEX IS INCREASED (INCREMENTED) BY THE AMOUNT OF THE EXPRESSION/*5X;  LES00390
*FOLLOWING "STEP", AND CONTROL IS TRANSFERRED BACK TO THE "FOR"  */5X;  LES00400
*STATEMENT WHERE THE LOOP CONTINUES UNTIL THE INDEX VALUE IS  */5X;  LES00410
*GREATER THAN THE FINAL VALUE.  FOR EXAMPLE:  */5X;  LES00420
*READ(5,101), END=3C21 CARD  LES00430
304 WRITE(6,104)  LES00450
104 FORMAT(6,4X), YOU WILL NOTICE THAT THE SIMPLE VARIABLE FOLLOWING"*/5X;  LES00460
*
**NEXT** IS THE SAME AS THE SIMPLE VARIABLE FOLLOWING**FOR**. /*
*AND THAT THE**NEXT** STATEMENT MARKS THE END OF THE LOOP* /*5X* LESO0500
*BEBECAUSE THE INCREMENT VALUE OF A LOOP IS CONSIDERED ONE(1) THE */ LESO0510
**STEP** MODIFIER AND ITS EXPRESSION MAY BE OMITTED, AND THE */ LESO0520
* INCREMENT VALUE WILL BE ASSUMED TO BE ONE(+1). FOR EXAMPLE */ LESO0530
* ABOVE**, FOR** STATEMENT COULD BE WRITTEN : */5X* LESO0540
*FOR I=1 TO 10**/*5X*/ THE INCREMENT VALUE AFTER**STEP** MAY BE */ LESO0550
*POSITIVE OR NEGATIVE ALLOWING THE FLEXIBILITY OF LOOPING FORWARD*/ LESO0560
** OR BACKWARD FOR A NEGATIVE**STEP** VALUE THE TEST BECOMES */ LESO0570
**LESS THAN ** FOR EXAMPLE THE FOLLOWING** FOR** STATEMENTS ARE/** LESO0580
**EQUIVALENT**: /*5X*/ FOR I=1 TO 10**/5X*/ FOR I=10 TO I 1**/-1**/ LESO0590
READ(5, 101) END=3051CARD

305 WRITE(10, 105)
105 FORMAT(1G, 10X)** ANOTHER USEFUL TECHNIQUE OF LOOPING IS** NESTING**/LESO0600
**/Nesting refers to placing one loop inside another loop. **/ LESO0630
*THE INNER LOOP**/* IS** AROUND AS MANY TIMES AS THE OUTER LOOP*/ LESO0640
*/IS INCREMENTED*. FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT */5X* LESO0650
*/FOR I=1 TO 10**/*5X*/ FOR J=1 TO 30**6X*/ NEXT J**/*6X*/ NEXT I**/*5X*/ LESO0660
*/NEXT J*/**6X*/ NEXT I**/6X*/ IN THIS EXAMPLE THE OUTSIDE LOOP(1) IS*/ LESO0670
*REPEATED 10 TIMES, AND THE INNER LOOP(J) WOULD BE REPEATED 30 /*
*/TIMES FOR EACH INCREMENT OF THE OUTSIDE LOOP, OR 200 REPETITIONS* LESO0690
*/ LOOPS MAY BE NESTED UP TO A MAXIMUM OF 20; HOWEVER, THEY */ LESO0700
*/CANNOT OVERLAP THE INNERMOST LOOP MUST BE CLOSED WITH**/ LESO0710
**/NEXT** STATEMENT BEFORE ENCOUNTERING THE NEXT OUTER LOOP**/S */ LESO0720
*NEXT** STATEMENT FOR EXAMPLE: */5X*/ LESO0730
*/ FOR X=1 TO 3**/*5X*/ FOR Y=2 TO 5**5X*/ FOR Z=5 TO -5**/*5X*/ LESO0740
*/ FOR X=1 TO 3**/*5X*/ FOR Y=2 TO 5**5X*/ FOR Z=5 TO -5**/*5X*/ LESO0750
READ(5, 101) END=3061CARD

306 WRITE(106)
106 FORMAT(15**/*15X*/ WITHIN A FOR/NEXT LOOP CONDITIONAL AND UNCONDITIONAL LESO0780
*/NATIONAL BRANCHES MAY BE USED TO TRANSFER CONTROL OUT OF A LOOP*/ LESO0800
*/OR WITHIN LIMITS OF THE SAME LOOP IF IT IS NOT POSSIBLE*/ LESO0810
*/ TO BRANCH INTO THE MIDDLE OF A FOR/NEXT LOOP BECAUSE LOGIC */ LESO0820
*/PROBLEMS OCCUR AND AN ERROR WILL RESULT. AN ADDITIONAL ITEM */ LESO0830
*/ TO BE CAREFUL ABOUT IS USING THE INDEX VARIABLE OF THE FOR/NEXT*/ LESO0840
*/ LOOP IN COMPUTATIONS IF YOU REFERENCE THE VALUE OF THE LOOP INDEX OR*/ LESO0850
*/ YOU WILL EFFECT THE ACTION OF THE LOOP. FOR EXAMPLE: */5X*/ LESO0860
*/ FOR I=1 TO 10**/*5X*/ LET I=I+1**/*5X*/ NEXT I**/*5X*/ PRINT I**/*5X*/ ENDLESO0860
*P*/**/ REPLY WITH VALUE OF I THAT IS PRINTED */ LESO0870
5 READ(5, 101) END=3071CARD
CALL CRUNCH(1, LENGTH)
IF(CARD1*EQ.11**/*5X*/ LET I=I+1**/*5X*/ NEXT I**/*5X*/ ENDLESO0880
10 CALL CRUNCH(2, LENGTH)
IF(CARD2*EQ.12**/*5X*/ LET I=I+1**/*5X*/ NEXT I**/*5X*/ ENDLESO0890
307 WRITE(10, 107)
107 FORMAT(10*)** FOR YOUR ANSWER IS INCORRECT. THE LOOP INDEX(1) IS */ LESO0920
*/ALTERED AND THE LOOP (I=11) WOULD THEN BE INCREMENTED AND TESTED (I=12)*/ LESO0930
*/ THE FINAL VALUE OF I=12***/ LESO0940
10 WRITE(10, 108)
108 FORMAT('O',' YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT WHAT YOU H\$E00970
*AVE JUST LEARNED */)  \$E00980
169 FORMAT('O','16X,' 1.) SAMPLE PROGRAM*/15X,' LET S=0*/15X,' FOR K=$5\$E01000
* TO 5*/15X,' LET S=S+K*/15X,' NEXT K*/15X,' PRINT 'SUM=',' S*/15X,' \$E01000
* END// REPLY WITH VALUE OF S */)  \$E01020
15 READ(5,101), END=310) CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1)=EQ.ASTRSK) GO TO 15
IF(CARDP(1).EQ.DIGIT(1).AND.CARDP(2).EQ.BLANK) GO TO 20
310 WRITE(6,110)
110 FORMAT('C,' YOUR ANSWER IS INCORRECT. THE VALUES OF K AND S */)
*IN THE LOOP ARE */5X,' K,' S/5X,' S,' 5*/5X,' -5*/5X,' 5*/5X,' -4*/5X,' 5*/5X,' 0/5X,' \$E01070
*5*/5X,' 5*/5X,' 0*/5X,' 0*/5X,' 0*/5X,' 0*/5X,' 0*/5X,' 0*/5X,' 0*/5X,' 0*/5X,' 0*/5X,' \$E01100
*15*/5X,' 15*/5X,' 15*/5X,' 15*/5X,' 15*/5X,' 15*/5X,' 15*/5X,' 15*/5X,' 15*/5X,' 15*/5X,' \$E01120
10 WRITE(6,111)
111 FORMAT('O','16X,' 2.) SAMPLE PROGRAM */15X,' FOR I=100 TO 1 STEP 2*/\$E01144
*15X,' LET J=1**2*/15X,' LET Y=2+10*/15X,' NEXT I*/15X,' PRINT I*/15X,' \$E01150
*15X,' END// REPLY WITH VALUE OF I */)  \$E01160
25 READ(5,101), END=312) CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1)=EQ.ASTRSK) GO TO 25
IF(CARDP(1).EQ.DIGIT(2).AND.CARDP(3).EQ.DIGIT(1)) GO TO 30
312 WRITE(6,112)
112 FORMAT('O,' YOUR ANSWER IS INCORRECT. THE INDEX VALUE OF THE */)
*LOOP IN 160 IS GREATER THAN THE FINAL VALUE(1) IN THE FIRST */)
*TEST. THIS I=100. IF THE LOOPS WERE DECREMENTED IN STEPS OF -2*/\$E01220
* THEN THE LOOP WOULD BE EXECUTED AND THEN I=0 //)  \$E01250
30 WRITE(6,113)
113 FORMAT('O','10X,' 3.) SAMPLE PROGRAM */15X,' FOR I=5*/\$E01290
* TO 1 STEP -1*/15X,' FOR J=1 TO 5*/15X,' IF J=0 THEN 10*/15X,' GO
* TO 15*/15X,' 10 LET T=I+1*/15X,' PRINT I*/15X,' 15 NEXT J*/15X,' \$E01280
* NEXT I*/15X,' PRINT T*/15X,' ENC// REPLY WITH VALUE OF T */)  \$E01300
35 READ(5,101), END=314) CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1)=EQ.ASTRSK) GO TO 35
IF(CARDP(1).EQ.DIGIT(6).AND.CARDP(2).EQ.BLANK) GO TO 40
314 WRITE(6,114)
114 FORMAT('O,' YOUR ANSWER IS INCORRECT. THE OUTSIDE LOOP(1) */)
* IS REPEATED 5 TIMES(I=5-->1): AND FOR EACH VALUE OF I THE */\$E01370
*INNER LOOP(j) IS REPEATED 5 TIMES(J=1-->5) THEREFORE THE */\$E01380
*THE CONDITIONAL BRANCH IS EXECUTED FIVE TIMES */5X,' I=5,' J=5*/5X,' \$E01390
*5*/5X,' J=2,' J=2*/5X,' I=3*/5X,' J=3*/5X,' I=4*/5X,' J=4*/5X,' \$E01410
* I=1,' J=1//)  \$E01410
40 WRITE(6,115)
115 FORMAT('O','10X,' 8.) SUBSCRIPTED VARIABLES/LISTS AND TABLES */5X,' \$E01420
*1. IN LESSON 1 YOU LEARNED THAT SIMPLE VARIABLES WERE A LETTER */\$E01440
*OR A LETTER FOLLOWED BY A DIGIT * SUBSCRIPTED VARIABLES CONSIST ** ESO1450
* OF A LETTER FOLLOWED BY A SINGLE OR DOUBLE SUBSCRIPT IN PARENTHESES ** ESO1460
** IS " THE SUBSCRIPTS MAY BE ANY LEGAL EXPRESSION THAT EVALUATES ** ESO1470
** TO AN INTEGER VALUE * FOR EXAMPLE : //1X,**
**A(1) , B(3) , D(1,3) , Z(1, J), X(3*A) , D(A(1)) ARE LEGAL SUBSCRIPT ** ESO1490
**RIPTED VARIABLES //1X,** BUT A(1) , Z(1,2,3) ARE ILLEGAL SUBSCRIPT
** PTD VARIABLES. // A USE OF SUBSCRIPTED VARIABLES FOLLOWS :/**
READ(5,101); END=316) CARD

316 WRITE(6,116)
116 FORMAT('6*,5X', '2. A NUMERIC LIST, OR VECTOR, OR SINGLE DIMENSION ** ESO1540
** NED*/ ARRAY, IT'S A SET OF NUMERIC VALUES ARRANGED IN AN ORDERLY ** ESO1550
** MANNER. FOR EXAMPLE, SUPPOSE YOU WOULD LIKE TO READ SOME //** ESO1560
** NUMBERS INTO YOUR PROGRAM AND HAVE THESE NUMBERS AVAILABLE AND ** ESO1570
** IDENTIFIED FOR USE AT A LATER TIME YOU COULD ASSIGN SIMPLE /** ESO1580
** VARIABLES TO EACH VALUE, BUT WHAT IF YOU HAD 100 NUMBERS ? ** ESO1590
** IT IS EASIER TO THINK OF THE NUMBERS AS A LIST OF VALUES OR A /** ESO1600
** VECTOR THE SIZE OF WHICH IS DETERMINED BY HOW MANY NUMBERS YOU /** ESO1610
** HAVE // YOU THEN SIMPLY ASSIGN THE VALUES TO THE LIST. //** ESO1620
READ(5,101); END=317) CARD

317 WRITE(6,117)
117 FORMAT('6*,5X', 'THE FOLLOWING PROGRAM WILL ASSIGN 10 VALUES TO THE */ ESO1650
** LIST**A** WHICH CONTAINS 10 ELEMENTS : //5X,** DIM A(10)/6X,** FOR /* ESO1760
** L=1 TO 10/5X,** READ A(L)/5X,** NEXT L/5X,** DATA 1,3,7,5,6,9,4,2,** ESO1670
**14,4/5X,** END// THE LIST A, CONTAINING 10 ELEMENTS, IS REPRESENT ** ESO1680
**EATED BY THE SUBSCRIPTED** VARIABLE A(L) READ A(L) IS IN A FOR/ESO1690
**NEXT LOOP AND ** END ** WITHIN THE LOOP THE LIST ELEMENT A(L)=A(1) TO A(10) ** ESO1700
**ARE ASSIGNED VALUES */ ** NOW THE LIST A HOLDS THE 10 VALUES AND EA /** ESO1710
**CH VALUE IS IDENTIFIABLE */ ** CONSIDER THE FOLLOWING PROBLEM WHICH ESO1720
** SEARCHES A LIST TO FIND THE LARGEST ELEMENT */**
READ(5,101); END=318) CARD

318 WRITE(6,118)
118 FORMAT('6*,5X', 'DIM N(10)/5X,** FOR L=1 TO 10/5X,** LET N(L)=0/5X,** ESO1750
** */** NEXT L/5X,** READ K/5X,** FOR J=1 TO K/5X,** READ N(J)/5X,** ESO1760
**/** NEXT J/5X,** LET L=N(1)/5X,** FOR J=2 TO K/5X,** IF N(J) LT L THEN /** ESO1770
** L=J/5X,** PRINT **LARGEST NUMBER= ** L/5X,** DATA 5,3,5,9,10,6,5/5X,** END**** ESO1780
**/** 45 READ(5,101); END=319) CARD
CALL CCHARLENGTH ** ESO1820
IFICARDP(1).EQ.ASTRSKX) GO TO 45 ** ESO1830
IFICARDP(1).EQ.DIGIT(2).AND.CARDP(2).EQ.DIGIT(1) GO TO 50 ** ESO1850
319 WRITE(6,119)
119 FORMAT('6*,5X', 'YOUR ANSWER IS INCORRECT * L IS ASSIGNED VALUES AS FOL ** ESO1860
**LLOWS : //5X,** L=3-->5-->9 //** ESO1870
50 WRITE(6,120)
120 FORMAT('6*,5X', 'THE FIRST FOR/NEXT LOOP ZEROS OUT THE LIST */** ESO1890
**IT IS A GOOD PRACTICE TO PUT SOME VALUES IN THE LISTS YOU USE */** ESO1900
**OTHERWISE THE COMPUTER MAY ASSIGN RANDOM VALUES * BY PUTTING */** ESO1910
*ZERO IN EACH ELEMENT OF THE LIST YOU ARE ASSURED THAT THE LIST/*
*IS CLEANED UP BEFORE YOU USE IT.** /LES01930
*READ(5,101),END=321ICARD
*
**121 WRITE(6,121)
**FORMAT(*'C',5X,'*') IN ADDITION TO LISTS(VECTORS), BASIC ALLOWS/**
**YOU THE ABILITY TO USE TABLES, OR MATRICES, OR TWO DIMENSIONAL /**/LES01950
**ARRAYS. THE SUBSCRIPTS OF A TABLE REPRESENT THE ROWS AND /**/LES01970
**COLUMNS OF THE TABLE. THE FIRST SUBSCRIPT IS THE ROW AND THE /**/LES01980
**SECOND SUBSCRIPT IS THE COLUMN. FOR EXAMPLE IN TABLE**D**'=S'/5X', **LES01990
**D(3,4) REFERS TO THE VALUE OF THE ELEMENT IN ROW 3, COLUMN 4/**/LES0200C
**OF TABLE**D**'=S'/5X', **LES02010
**READ(5,101),END=322ICARD
*
**122 WRITE(6,122)
**FORMAT(*'C',5X,'*') AS WITH A LIST. ANY ELEMENT OF A TABLE MAY BE **/*
**REFERENCED BY DEFINING THE PAIR OF SUBSCRIPTS AS DESIRED. **/*LE02050
**DIMX(COLUMN ORDER) IN A 3X3 TABLE(MATRIX) THE TABLE IS REFERENCED. **LES0260
**/* AS FOLLOWS : '*/5X, '1,1,5X, '1,2,5X, '1,3,5X, '2,1,5X, '2,2,5X', **LES0270
**X, '2,3,5X, '3,1,5X, '3,2,5X, '3,3,5X', **WHERE THE FIRST SUBSCRIPT **LES0280
**R1PT IDENTIFIES THE ROW AND THE SECOND THE COLUMN. **/5X', **LES0290
**USING THIS REFERENCE SYSTEM CONSIDER HOW TO FILL THE FOLLOWING TABLES: **LES0210D
**LE : '*/5X, '1,1,5X, '1,2,5X, '1,3,5X, '2,1,5X, '2,2,5X', **LES02110
**X, '2,3,5X, '3,1,5X, '3,2,5X, '3,3,5X', **/* 0,0,5X, '9,9,5X', **LES02120
**READ(5,101),END=323ICARD
*
**123 WRITE(6,123)
**FORMAT(*'C',5X,'*') THIS TABLE**X** COULD BE FILLED AS FOLLOWS : '*/5X', **LES02150
**DIMX(3,3)/6X,'*' FOR I=1 TO 3'/'6X,'*' FOR J=1 TO 3'/'5X,'*' READ XI,J','/**
**X, 'NEXT J'/'5X', 'NEXT I'/'5X', 'DATA 1,2,3,4,5,6,7,8,9'/'5X', END,'/**
**/* REPLY WITH VALUE OF XI(2,3)/**)
**55 READ(5,101),END=324ICARD
**CALL CRUNCH(1,LNTH)
**IF(CARDP(1).EQ.(ASTPSK)) GO TO 55
**IF(CARDP(1).EQ.(DIGIT(7))) GO TO 60
**52 WRITE(6,124)
**124 FORMAT(*'C',5X,'*') YOUR ANSWER IS INCORRECT. THE VALUE OF XI(2,3)=6.**/*
**GO BACK AND LOOK AT THE REFERENCE SYSTEM. XI(2,3) IS THE ELEMENT/**/LES02250
**IN ROW 2, COLUMN 3/**/LES02260
**READ(6,125)
**125 FORMAT(*'C',5X,'*') NOW WITH TABLE**X** ASSIGNED VALUES CONSIDER THIS**/*LES02280
**PROGRAM: THE SUSPENDING TABLE**X** HAS BEEN FILLED BY THE ABOVE PROGRAM**LES0230
**/* '*/5X,'*' LET S=0'/'5X,'*' FOR I=1 TO 3'/'5X,'*' FOR J=1 TO 3'/'5X,'*' IF I .LE. **LES02330
**J THEN S=S+XI(J,J)'/'5X,'*' NEXT J'/'5X,'*' NEXT I'/'5X,'*' PRINT'/'SUM=S'/'5X,'*' END,'/**
**/* REPLY WITH VALUE OF S/**)
**65 READ(5,101),END=323ICARD
**CALL CRUNCH(1,LNTH)
**IF(CARDP(1).EQ.(ASTPSK)) GO TO 65
**IF(CARDP(1).EQ.(DIGIT(2)) AND CARDP(2).EQ.(DIGIT(6))) GO TO 70
**62 WRITE(6,126)
**126 FORMAT(*'C',5X,'*') YOUR ANSWER IS INCORRECT. THIS PROGRAM SUMS THE **/*LES02380
*ELEMENTS OF THE TABLE ON THE MAJOR DIAGONAL (UPPER LEFT TO LOWER  
*RIGHT) */ 'THE SUM=15 */ /// 
70 WRITE(6,127) LESO2390 
127 FORMAT(6,5X,'YOU HAVE SEEN HOW SUBSCRIPTED VARIABLES ARE */ /// 
*USED TO SET UP LISTS (VECTORS) AND TABLES (MATRICES). HOWEVER TO** /// 
*USE LISTS AND TABLES IN A PROGRAM, YOU MUST DIMENSION THE */ /// 
*MAXIMUM SIZE OF YOUR LIST OR TABLE SO THAT ENOUGH SPACE WILL */ /// 
*BE ALLOCATED IN THE COMPUTER MEMORY. THIS DIMENSIONING IS DONE*/ /// 
*WITH THE DIM STATEMENT. */ /// 
READ(5,101),END=3291 CARD /// 
328 WRITE(6,128) LESO2490 
128 FORMAT(10,10X,'C DIM STATEMENT */ /// 
*THE DIM STATEMENT TELLS THE COMPUTER THE MAXIMUM SIZE OF VECTORS; LESO2510 
** AND TABLES THAT WILL BE USED IN YOUR PROGRAM; THE DIM STATEMENT LESO2520 
*"MUST APPEAR IN THE PROGRAM BEFORE ANY" REFERENCE IS MADE; LESO2530 
*IT TO THE LIST OR TABLE. IN GENERAL PRACTICE THE DIM STATEMENT */ /// 
*IS USUALLY THE FIRST STATEMENT IN THE PROGRAM. THE FORM OF THE */ /// 
*DIM STATEMENT IS: */ /// 
*DIM STATEMENT IS: */ /// 
*DIM  
*DIM ** X ** DIM ** <TABLE VARIABLES> ( **SIZE** ) /*1/ /// 
*DIM ** A ** (10,3) , (11,4,2) /// 
READ(5,101),END=3291 CARD /// 
329 WRITE(6,129) LESO2590 
129 FORMAT(10,5X,' THE LIST AND TABLE VARIABLES ARE SUBSCRIPTED VARIABLES LESO2660 
*5/) AS DEFINED EARLIER. THE SIZE IS AN UNSIGNED INTEGER IN */ /// 
*PARENTHESES WHICH DENOTES THE MAXIMUM SIZE OF THE LIST OR TABLE; LESO2620 
*THE DIM STATEMENT MAY CONTAIN A NUMBER OF LISTS OR TABLES. LESO2630 
*WITH THEIR SIZES SEPARATED BY COMMAS; FOR EXAMPLE: */ /// 
*DIM ** A ** (10,3) , (11,4,2) /// 
READ(5,101),END=3301 CARD /// 
330 WRITE(6,130) LESO2670 
130 FORMAT(10,5X,' IF YOU TRY TO REFERENCE AN ELEMENT IN A LIST */ /// 
*OR TABLE BEYOND THE MAXIMUM SIZE IN THE DIM STATEMENT YOU WILL; LESO2690 
*GET AN ERROR IN THE MAXIMUM SIZE LIST (VECTOR) OR TABLE (MATRIX) LESO2700 
*ALLOWED BY THE C++ BASIC COMPILER IN ANY ONE PROGRAM IS A TOTAL; LESO2710 
*OF 1600 COMPUTER MEMORY SPACES. */ /// 
*YOU WILL NOW BE ASKED SOME QUESTIONS: */ /// 
*1) ARE THE FOLLOWING DIM STATEMENTS CORRECT, REPLY: YES OR NO */ /// 
*DIM ** A ** (10) /// 
75 READ(5,101),END=331 CARD /// 
75 CALL CRUNCH LENGTH /// 
75 IF(CARDP(1),EQ.'ASTRSK') GO TO 75 /// 
75 IF(CARDP(1),EQ.'ALPHA') GO TO 80 /// 
331 WRITE(6,131) LESO2790 
131 FORMAT(10,5X,' THE STATEMENT IS INCORRECT. SUBSCRIPTED VARIABLES */ /// 
*ARE A SINGLE LETTER FOLLOWED BY ONE OR TWO SUBSCRIPTS IN */ /// 
*PARENTHESES */ /// 
80 WRITE(6,132) LESO2840 
132 FORMAT(10,5X,' DIM ** X ** (50C), (B11,10), (C5,5,5) /// 
85 READ(5,101),END=3331 CARD /// 
85 LESO2860
CALL CRUNCH(ILNGTH)
IF(ICARDP(1).EQ.ASTRSK) GO TO 85
IF(ICARDP(1).EQ.ALPHA(14)) GO TO 90
333 WRITE(6,133) LESO2930
133 FORMAT('Q'; 'THE STATEMENT IS INCORRECT. C(5,5) IS ILLEGAL '; '/;')
*ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.*/)
90 WRITE(6,134)
134 FORMAT('Q'; 'THE STATEMENT IS INCORRECT. C(5,5) IS ILLEGAL '; '/;')
*ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.*/)
333 WRITE(6,135) LESO2930
135 FORMAT('Q'; 'THE STATEMENT IS INCORRECT. C(5,5) IS ILLEGAL '; '/;')
*ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.*/)
336 WRITE(6,136) LESO2930
136 FORMAT('Q'; 'THE STATEMENT IS INCORRECT. C(5,5) IS ILLEGAL '; '/;')
*ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.*/)
95 READ(5,101,END=337)CARD
96 READ(5,101,END=338)CARD
337 WRITE(6,137) LESO3330
137 FORMAT('Q'; 'THE STATEMENT IS INCORRECT. C(5,5) IS ILLEGAL '; '/;')
*ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.*/)
95 READ(5,101,END=337)CARD
96 READ(5,101,END=338)CARD
337 WRITE(6,137) LESO3330
IF(CARDP(1).EQ.ASTRSK) GO TO 96
IF(CARDP(1).EQ.APHA(14)) CALL EXIT
338 WRITE(6,138)
138 FORMAT(10*',''PROBLEM 1,''/5X,''REM PROGRAM TO REVERSE ELEMENTS IN''//LES03370
* A LIST*/5X,' ins REM y IS A TEMPORARY LOCATION FOR THE ELEMENT BEING''//LES03370
* SWAPPED*/5X,' ins REM y IS A TEMPORARY LOCATION FOR THE ELEMENT BEING''//LES03370
* LET X(I)=X(I-1)*/5X,' ins REM y IS A TEMPORARY LOCATION FOR THE ELEMENT BEING''//LES03370
* NEXT I*/5X,' ins END''//P''//LES03390
* PROBLEK 2,''//5X,' ins REM SORT LIST IN DECENDING ORDER*/5X,' REM S. ''//LES03410
* S THE EXCHANGE COUNTER''//5X,' ins DIM D(5)''//LES03420
***/5X,' ins FOR I=1 TO 5*/5X,' ins READ D(I)*/5X,' ins NEXT I*/5X,' ins LET S(L)*/5X,' ins FOR I=1 TO 5*/5X,' ins PRINT D(I)*/5X,' ins NEXT I*/5X,' ins END''//P''//LES03460
*R I=1 TO 5*/5X,' ins PRINT D(I)*/5X,' ins NEXT I*/5X,' ins END''//P''//LES03470
CALL EXIT
END

THIS LESSON INTRODUCES SUBROUTINES AND RECURSION

COMMON
- STACK(100), PROG(200), CARD(60), CARDP(80), ALPHA(48),''//LES00020
- IAPTR, IMPTR, IADATA(500) + XADATA(500), STRING(5)''//LES00020
- DIGIT(10), IPRIT(10), LIST(100), ISTAT(100), ''//LES00250
- PR1(250), NERRS, ISTR, NSTLST, DEBUG, DOLGON, QUOTE, ''//LES00060
- EQUALS, PRT1, DECMAL, PLUS, CMINUS, SLASH, COMMA, ''//LES00070
- PARLFT, ASTRSK, BLANK''//LES00080
CMMCN INTERP, EXERR
REAL*8 LESS6/LES06''//REAL ALLOD(LESS6,N1)
WRITE(6,10C)
100 FORMAT(10*',''LESSON 6 '//'LES00100
* THIS LESSON WILL INTRODUCE YOU TO SUBROUTINES AND THE PROGRAMMING*''//LES00110
** TECHNIQUE CALLED 'RECURSION''///5X,' ins A SUBROUTINE */5X,' ins
** IN MANY PROGRAMS A BLOCK OF STATEMENTS MAY BE NEEDED ON MORE */5X,' ins THAN ONCE''//LES00140
** THE STATEMENTS ARE EXECUTED SEQUENTIALLY''//LES00150
** IS BURDENSOME, A TECHNIQUE TO ECONOMIZE ON CODING INSTRUCTIONS''//LES00160
** CALLED A SUBROUTINE ''//LES00170
** IS EXECUTED ONLY WHEN OCCASIONALLY CONTROLS европейский LES00180
** *//5X,' ins RETURN''///LES00190
** READS(5,101, END=3C2)''//LES00200
101 FORMAT(80A1)''//LES00210
302 WRITE(6,102)
102 FORMAT(10*',''EXECUTION OF THE GOSUB STATEMENT CAUSES THE''//LES00230
* COMPUTER TO TRANSFER CONTROL TO THE STATEMENT NUMBER AFTER''//GOSUB''//LES00240
** */5X,' ins WHEN CONTROL IS TRANSFERRED''//LES00250
** IALSO''//5X,' ins RETURN''///LES00260
** L*/5X,' ins RETURNED TO THE NEXT STATEMENT FOLLOWING THE ''//GOSUB''///LES00270
** STATEMENT WHICH *CALLED* THE SUBROUTINE . FOR EXAMPLE CONSIDER''//LES00280
*READ N'/5X,' IF N GT 0 THEN 20*/5X,' GO TO 50*/3X,' 20 PRINT N'/5X
*1' GOSUB 100*/5X,' PRINT F'/5X,' GO TO 10*/8X,' 100 REM ITERATIVE
*SOLUTION'/13X,' LET F=1*/13X,' FOR I=1 TO N'/14X,'LET F=F+1*/13X,'LES00790
* NEXT I*/10X,' REM RETURN FACTORIAL(N)'/*8X,' RETURN*/3X,' 50 REM
*FACTORIAL F(N)= IF N=0 THEN (N)/1*/5X,' REM OTHERWISE , F(N)=LES00810
*N*(N-1)/5X,' RESTORE*/5X,' PRINT 'RECURSIVE SOLUTION'/3X,' 5X R
*READ N'/5X,' IF N GT 0 THEN 70*/5X,' GO TO 9999*/3X,' 7G PRINT N'/5
*X,' GO SUB 200*/5X,' PRINT F'/6X,' GO TO 60*/8X,' 20C REM RECURSIVE
*SOLUTION'/13X,' IF N GT 0 THEN 210*/13X,' LET F=1*/13X,' RETURN*/8LES09050
*X,'*210 LET N=N-1*/13X,' REM RECURSIVE CALL'/13X,' GOSUB 200*/13X,'LES0860
* LET N=N+1*/13X,' LET F=F*N*/8X,'RETURN*/5X,' DATA 2,5,8,6,-1*/9 LES08780
*999 END/*13
*LES03880

308 WRITE(16,108)
*IN THE ITERATIVE METHOD THE FACTORIAL FUNCTION , 1/* LES0910
*F(N) WAS MULTIPLIED BY ITSELF IN A LOOP FROM I=1--N IN THE/* LES0920
* RECURSIVE SOLUTION THE FACTORIAL FUNCTION F(N), IS DEFINED /* LES0930
* IN TERMS OF ITS FINAL VALUE . WHEN N=0 , F(N)=1 ; OTHERWISE /* LES0940
*F(N)=N*(FACTORIAL CE N-1) .

309 WRITE(16,109)
* TO FULLY UNDERSTAND THE CONCEPT OF RECURSION YOU /* LES0950
* SHOULD STEP THROUGH THE FACTORIAL PROBLEM BY HAND USING THE /* LES0960
*HELP OF THE STACK TO SEE HOW RECURSION WORKS . IF YOU UNDERSTAND /* LES0970
* THE CONCEPT OF RECURSION YOU ARE READY TO SOLVE THIS PROBLEM : /* LES0980
* 5X ' LET X=1*/5X,' GOSUB 100*/5X,' PRINT X'/5X,' GO TO + .9997*/8X,'LES0990
*GU LET X=X+1*/13X,' IF X GT 3 THEN 150*/13X,'GOSUB 100*/8X,'15G LES1000
*LET X=X+1*/8X,'RETURN*/ 9999 END/// REPLY WITH VALUE OF X ///
5 READ(S,IC1,END=309ICARD
CALL GUNCH(LINCGTH)
IF CARDP(1).EQ.'ASTR3X') GO TO 5
IF CARDP(1).EQ.'DPLICIT') AND CARDP(2).EQ.'BLANK') GO TO 10

310 WRITE(16,110)
* YOUR ANSWER IS INCORRECT . CONSIDER THE FOLLOWING /// LES1100
*TABLE OF VALUES FOR X AND ITEMS IN THE STACK (FIRST GOSUB; 2-SELS0180
*GOSUB 100*/5X,' PRINT F'/5X,' GO TO 50*/8X,' 100 REM ITERATIVE
*5X ' LET X=X+1*/5X,' GOSUB 100*/5X,' PRINT X'/5X,' GO TO + .9997*/8X,'LES0990
* GU LET X=X+1*/13X,' IF X GT 3 THEN 150*/13X,'GOSUB 100*/8X,'15G LES1000
*LET X=X+1*/8X,'RETURN*/ 9999 END/// REPLY WITH VALUE OF X ///
5 READ(S,IC1,END=309ICARD
CALL GUNCH(LINCGTH)
IF CARDP(1).EQ.'ASTR3X') GO TO 5
IF CARDP(1).EQ.'DPLICATE') AND CARDP(2).EQ.'BLANK') GO TO 10

310 WRITE(16,110)
* EXAMPLE UNTIL YOU UNDERSTAND THE RECURSION TECHNIQUE HOWEVER , /* LES1150
* KNOWING HOW TO USE RECURSION IS NOT A REQUIREMENT FOR KNOWING /* LES1170
* HOW TO USE BASIC, IT IS ONLY A CLASSIC PROGRAMMING TECHNIQUE */LES01250
*)
10 WRITE(6,111)
111 FORMAT(*5X,'C SUMMARY') /LES01260
* SUBROUTINES AND RECURSION ARE A VALUABLE ADDITION TO YOUR REPERTOIRE
* */ OF BASIC STATEMENTS * THEY SAVE BOTH PROGRAMMER TIME AND */LES01300
* COMPUTER STORAGE SPACE * THE GOSUB/RETURN COMMAND AND THE OTHER */LES01310
* TWELVE BASIC STATEMENTS THAT YOU HAVE ALREADY LEARNED AND USED */LES01320
* FORM THE BASIC LANGUAGE * */LES01330
* THE FACILITY THAT YOU GAIN IN PROGRAMMING BY USING THE BASIC */LES01340
* LANGUAGE WILL DEPEND UPON HOW OFTEN YOU EXERCISE YOUR SKILLS */LES01350
*, THE LAST LESSON WILL PROVIDE YOU WITH A BRIEF SUMMARY OF THE */LES01360
* BASIC LANGUAGE */)
READ(5,10) END=3121CARD LEO1370
312 WRITE(6,112) LEO1380
112 FORMAT(*5X,'THE FOLLOWING PROBLEMS WILL TEST YOUR LATEST SKILLS') LEO1430
* $ */5X.*) 1. COMPUTE THE NET SALARY - RETIREMENT CONTRIBUTION*/LES01410
* AND TAX FOR N EMPLOYEES. NET PAY= GROSS SALARY-RETIRED-TAX */LES01420
* USE SUBROUTINES TO CALCULATE */10X*) A RETIREMENT CONTRIBUTION*/LES01430
* 15X*) IF SALARY < $600, R=$0.0/15X*) IF SALARY < $2500, R=$10.0/LES01440
* /15X*) IF SALARY > $2500, R=$20.0/10X*) B) TAX=15X*) IF SALARY < $4500 */LES01450
* /$600) T=$5; OTHERWISE1/15X*) T=03{SALARY})/ LES01460
* E) EMPLOYEE, 10X*) GROSS SALARY/7X*) JONES, 15X*) 3000/7X*) SMITH, 15X*) LEO1470
* X*) 5200/7X*) BROWN, 15X*) 75/7X*) DALEY, 15X*) 10, 500/7X*) BERRY, 15X*) LEO1480
* 5X*) 14300/7X*) 2) WRITE A RECURSIVE PROGRAM TO SUM THE NUMBERS*/LES01490
* FROM X TO Y * READ IN YOUR OWN DATA AND TEST YOUR PROGRAM */)
READ(5,10) END=3131CARD LEO1510
313 WRITE(6,113) LEO1520
113 FORMAT(*5X.*) IF YOU WANT TO EXECUTE THESE PROGRAMS NOW , THEN */
* REPLY: YES, OTHERWISE NO//)
20 READ(5,101) END=3131CARD CALL CRUNCH(LENGTH) LEO1550
IF(CARDP(1).EQ.0) GO TO 20 LEO1550
IF(CARDP(1).EQ.0) CALL EXIT LEO1570
CALL TEST1 LEO1590
WRITE(6,114) LEO1630
114 FORMAT(*5X.*) IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEMS*/
* REPLY: YES//)
25 READ(5,101) END=3151CARD CALL CRUNCH(LENGTH) LEO1630
IF(CARDP(1).EQ.0) GO TO 25 LEO1640
IF(CARDP(1).EQ.0) CALL EXIT LEO1650
315 WRITE(6,115) LEO1670
115 FORMAT(*5X.*) PROBLEM 1*/5X.*) REM ES=EMPLOYEE, S=GROSS SALARY, R=RET */LES01690
* IREMENT*/5X.*) REM T=TAX, P=NET PAY*/5X.*) READ N*/5X.*) FOR I=1 TO N*/LES01690
* */9X.*) REM T=TAX, P=NET PAY*/5X.*) READ N*/5X.*) FOR I=1 TO N*/LES01700
* PRINT ES,S,T,P */9X.*) GO TO 9999*/9X.*) REM TAX CAL */LES01700
* CULATION*/10X.*) IF 'S LT 600 THEN 40*/13X.*) LET T=0.03*/S*/13X.*) GO
*TO 50/9X, .40 LET T=5/9X, .50 RETURN/9X, .30 REM RETIREMENT CALCULLESQ01730
*ACTION//13X, .40 IF S GE 2500 THEN 60/13X, .40 IF S GE 800 THEN 70/13X, .70LES01740
** LET R=60/13X, .60 GOTO 30/9X, .60 LET R=20/13X, .60 GOTO 30/9X, .70LES01750
* LET R=10/9X, .80 RETURN/5X, .90 DATA **JONES**/300/85
**X, . DATA **SMITH**/550/5X, . DATA **BROWN**/475/5X, . DATA **DALE**
**Y, .10500/5X, . DATA **BERRY**, .430U**, .9999 END//** PROBLEM 2//7/5
**X, . REM SUM FROM X-->Y**//6X, .PRINT** SUM FROM X-->Y**//6X, .LET S=
**C/3X, .1 READ X,Y,S/5X, . IF X=9999 THEN 9999/5X, . GCSUB 10/5X, . P
**PRINT X,Y,S/5X, . GO TO 1/9X, .10 REM SUM NUMBERS RECURSIVELY//13X, .LES01810
** IF X NE Y THEN 20/13X, . LET S=X/13X, . RETURN/9X, .20 LET X=X+1LES01820
**/13X, . GCSUB 10/13X, . LET Y=Y+1/13X, . RETURN*/5LES01830
**X, . DATA 3,6/5X, . DATA 1,10 */5X, . DATA 9999,1 */9999 END//** LES01840
CALL EXIT
END

THIS LESSON GIVES A COMPLETE SUMMARY OF BASIC LANGUAGE STATEMENTS

COMMON
- STACK(100), PROG(200), CARD(80), CARDP(80), ALPH(48),
- IA(10), IN(10), LIST(100), LSTLST(100),
- PR(250), NERRS, INST, NTLST, DEBUG, DLOGN, QUOTE,
- EQUALS, PART, DECIMAL, PLUS, MINUS, SLASH, COMMA,
- PARLFT, ASTRK, BLANK

COMMON
- interp, IEXERR
- real*8 les7*leson */
- call alogr(les7,n1)

100 FORMAT('10'/*5X,'*** LESSON 7 ***'//
** THIS LESSON SUMMARIZES THE BASIC DEFINITIONS AND BASIC STATEMENTS:
** THAT YOU HAVE LEARNED INX(A)-BASIC /**5X, ** A. BASIC DEFINITION
**S/** 1. ALPHANUMERIC CHARACTERS :
**A) DIGITS : 0--9
**B) LETTERS : A--Z
**C) SPECIAL CHARACTERS : ** */ + - ( ) = ", "$
** 2. STATEMENT NUMBERS : ONE TO FOUR DIGITS (0-9999)
** 3. STRING : ANY SEQUENCE OF ALPHANUMERIC CHARACTERS ENCLOSED
** IN *"* SINGULAR QUOTES. EG. **"HELP** /*
** READS(101,101=0, END=302)+CARD

101 FORMAT('80A1')
102 FORMAT('6.2')

* A) NUMBERS : (LIMITED TO 9 DIGITS)
**A) INTEGERS : DIGITS WITH NO FRACTIONAL PART. EG. 5, 7, 10 */5X,
**B) REAL : DIGIT WITH A FRACTIONAL PART. EG. 5.0, 7.31, 16.0 */
**A NUMBER MAY BE PRECEDED BY A SIGN (+), BUT IS ASSUMED TO
**BE POSITIVE IF NONE IS GIVEN.
** 5. VARIABLES :
**A) SIMPLE VARIABLES : A SINGLE LETTER, OR A SINGLE LETTER /*5X,
*FOLLOWED BY A DIGIT  *  EG  A, A1, B6, 20  

* B)  ALPHA VARIABLES  *  A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN ($)  

* C)  SUBSCRIPTED VARIABLE  *  (SINGLE LETTER)  

* D)  SINGLE SUBSCRIPT  *  (LETTER)  

* (E)  DOUBLE SUBSCRIPT  *  (LETTER)  

* /  EG  (A)[(5, 10), X(A*B, X**2)] */ 

READ(5, 101, END=306) CARD 

303 WRITE(6, 101) 5, OPERATORS: (LISTED IN DECENDING HIERARCHY) */ 5X, *  

* A)  EXPONENTIATION  **  */  5X, *  

* B)  MULTIPLICATION  *  */  5X, *  

* C)  DIVISION  */  5X, *  

* D)  ADDITION  */  5X, *  

* E)  SUBTRACTION  */  5X, *  

* G)  OPERATIONS:  */  5X, *  

* A)  SINGLE NUMBER OR VARIABLE  *  EG.  10, -5, D, X(I, J)  */  5X, *  

* B)  BUILT-IN FUNCTION  *  EG. SQR(10), TAN(25)  */  5X, *  

* C)  ARITHMETIC EXPRESSION  *  OPERATIONS SEPARATED BY OPERATORS  */  

AND GROUPED BY PARENTHESIS  *  EG.  5+6.0 , A**2(3*X-4) */  

READ(5, 101, END=306) CARD 

304 WRITE(6, 104) 7, RELATIONS: */ 5X, *  

* A)  GT  A GT B  A GREATER THAN B  */  5X, *  

* B)  GE  A GE B  A GREATER THAN OR EQUAL TO B  */  5X, *  

* C)  LT  A LT B  A LESS THAN B  */  5X, *  

* D)  LE  A LE B  A LESS THAN OR EQUAL TO B  */  5X, *  

* E)  NE  A NE B  A NOT EQUAL TO B  */  5X, *  

* F)  =  A = B  A EQUAL TO B  */  5X, *  

* G)  BASIC STATEMENTS  */  5X, *  

* H)  REM << ANY SET OF ALPHANUMERIC COMMENTS >> */  

* I)  READ << VARIABLE.....VARIABLE >> */  

* J)  END  */  5X, *  

READ(5, 101, END=320) CARD 

305 WRITE(6, 105) 4, LET << VARIABLE >> = << EXPRESSION >> */  5X, *  

* A)  PRINT << "STRING" OR EXPRESSION::"STRING" OR EXPRESSION  

*/ /* 5X, * SIMPLY PRINT  */  

* B)  DATA << "STRING" OR NUMBER::"STRING" OR NUMBER >> */  

* C)  RESTORE OR RESTOR  */  5X, *  

* D)  IF << EXPRESSION >> << RELATION >> THEN << STATEMENT NUMBER >> */  

* E)  GO TO << STATEMENT NUMBER >> */  

* F)  GO ON << EXPRESSION >> GO TO << STATEMENT NUMBER >> /* 30X  

* G)  TO << EXPRESSION >> STEP << EXPRESSION >> */  5X, *  

* H)  NEXT << SIMPLE VARIABLE >> * << EXPRESSION >> */  5X, *  

* I)  DIM << LETTER >> 1 << INTEGER EXPRESSION >> */  

DIM << LETTER >> 1 << INTEGER EXPRESSION >>
* 13. GOSUB << STATEMENT NUMBER >>
* 13. RETURN
*READ(S,101,END=306)CARD
306 WRITE(4,106)
106 FORMAT('C', ' THAT CONCLUDES YOUR INSTRUCTION WITH CAI-BASIC.
* HOWEVER, YOU ARE INVITED TO USE WHATEVER FACILITIES OF CAI-BASIC
* YOU DESIRE AT ANY TIME. IF YOU HAVE NOT RUN PROGRAMS UNDER
* THE OS/BATCH MODE (PUNCHING YOUR OWN CARDS AND HANDING THEM
* ACROSS THE COUNTER TO BE RUN), Y'ALL SHOULD GET THE BASIC MANUAL.
* TECHNICAL NOTE NR. 3211-12, IN ROOM 1-147 TO FIND THE PROPER
* JCB CONTROL CARDS REQUIRED. GOOD LUCK WITH THE COMPUTER, AND
* REMEMBER IT ONLY DOES WHAT YOU TELL IT TO DC - GIGO (GARBAGE IN
* GARBAGE OUT).
')
CALL EXIT
END

LES02030
APPENDIX C

13.13.46 START EXECUTION BEGINS...

Hi, WELCOME TO CAI-BASIC , THERE ARE ONLY A FEW SIMPLE RULES TO REMEMBER IN ORDER TO HAVE A SUCCESSFUL SESSION ON THE TERMINAL WITH CAI-BASIC:

1. WHEN ASKED FOR A RESPONSE , TYPE IN THE CORRECT REPLY AND HIT THE CARRIAGE RETURN KEY ON THE RIGHT SIDE OF THE KEYBOARD.

2. IF YOU MAKE A TYPING ERROR WHILE MAKING ANY RESPONSE OR INPUT , TYPE IN FOUR DOLLAR SIGNS ($$$*) AFTER THE ERROR OR ANYWHERE ON THAT INPUT LINE AND HIT CARRIAGE RETURN .THE ENTIRE LINE WILL THEN BE IGNORED AND YOU CAN TYPE IN THE CORRECT INPUT OR RESPONSE.

3. IF AT ANY POINT IN THE SESSION YOU WANT TO STOP THE SESSION TYPE IN THE WORD ‘QUIT’ AS SOON AS YOU ARE ASKED FOR THE NEXT RESPONSE ,HIT CARRIAGE RETURN, THEN HIT ATTN KEY AND TYPE LOGOUT.

4. DURING YOUR TERMINAL SESSION CAI-BASIC WILL HALT OCCASIONALLY TO LET YOU READ A SEQUENCE OF INFORMATION . WHEN YOU ARE READY TO CONTINUE , TYPE IN 'GO' , AND HIT CARRIAGE RETURN.

5. DURING YOUR TERMINAL SESSION YOU MAY NOTICE THAT THE TYPING IS NOT ALWAYS PERFECT . SOME DAYS THE COMPUTER IS NOT UP TO PAR AND YOU WILL HAVE TO ADJUST TO THE MINOR IRRITANT.

IF YOU ONLY WANT TO EXECUTE PROGRAMS AT THIS TIME THEN REPLY : YES ; OTHERWISE REPLY : NO .

no

IF THIS IS YOUR FIRST SESSION WITH CAIBASIC, THEN REPLY: YES ;OTHERWISE REPLY: NO .

yes

CAIBASIC IS A PROGRAM TO TEACH YOU THE FUNDAMENTALS OF A PROGRAMMING LANGUAGE. THE LANGUAGE TO BE LEARNED IS BASIC ; A SIMPLE LANGUAGE FOR THE USER WHO HAS LITTLE KNOWLEDGE OF COMPUTERS AND WHOSE PRIMARY INTEREST IS IN OBTAINING RESULTS.

THE SIMPLICITY OF THE BASIC LANGUAGE AND ITS RANGE OF CAPABILITIES SHOULD ALLOW YOU TO LEARN THE LANGUAGE AND
WRITE PROGRAMS IN A MINIMAL AMOUNT OF TIME.

THE REFERENCE TEXTS RECOMMENDED FOR CALLBASIC ARE:

1. BASIC LANGUAGE MANUAL, TN 0211-12 APRIL 1971
   (FREE UPON REQUEST IN 1-167)

2. INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE, R.L. NOLAN
   (BOOKSTORE / MAIN LIBRARY)

3. BASIC PROGRAMMING, V.C. HARE
   (COMPUTER CENTER LIBRARY)

*** AFTER YOU HAVE FINISHED READING AN INPUT, TYPE IN GO AND HIT THE RETURN
AND THE PROGRAM WILL CONTINUE ***

DURING YOUR TERMINAL SESSION YOU WILL BE LEARNING
THE STRUCTURE OF THE LANGUAGE BASIC. THE INSTRUCTION SET
CONTAINS 7 LESSONS AND YOU MAY PROCEED THROUGH THE LESSONS AT
YOUR OWN SPEED.

THE LESSONS ARE AS FOLLOWS:

LESSON 1 PROGRAM FORMAT AND BASIC DEFINITIONS
LESSON 2 REMARKS, INPUT/OUTPUT AND DATA
LESSON 3 ASSIGNMENT STATEMENTS AND BUILT IN FUNCTIONS
LESSON 4 BRANCHING
LESSON 5 LOOPS AND SUBSCRIPTED VARIABLES
LESSON 6 SUBROUTINES AND RECURSION
LESSON 7 SUMMARY OF BASIC STATEMENTS

** WHEN YOU ARE READY TO CONTINUE, TYPE IN GO AND HIT
THE CARRIAGE RETURN ***

IN EACH LESSON YOU WILL BE GIVEN INSTRUCTION
SEQUENCES AND THEN YOU WILL BE ASKED QUESTIONS TO SEE IF YOU
UNDERSTOOD THE INSTRUCTIONS. THE QUESTIONS WILL BE OF VARIOUS
TYPES: MULTIPLE CHOICE, TRUE/FALSE, ACTUAL PROGRAM STATEMENTS, ETC.
YOU WILL BE PROMPTED FOR YOUR ANSWER, AND WHEN READY TYPE IN
OUR RESPONSE AND HIT THE RETURN KEY.

IF YOU KEEP THE TELETYPE OUTPUT FROM YOUR TERMINAL SESSION
YOU WILL HAVE A READY REFERENCE FOR FUTURE USE.
LESSON 1

THIS INSTRUCTION SET WILL INTRODUCE YOU TO THE STRUCTURE OF BASIC LANGUAGE STATEMENTS, THE RULES FOR VARIABLES AND NUMBERS, AND THE SYMBOLS FOR ARITHMETIC OPERATIONS.

DON'T FORGET TO TYPE GO AND HIT RETURN WHEN READY TO CONTINUE.

A. PROGRAM STRUCTURE

THE BASIC LANGUAGE, AND ALL OTHER LANGUAGES, HAS A SPECIFIED STRUCTURE. EACH BASIC STATEMENT HAS A REQUIRED FORM WITH POSSIBLY ONE OR MORE VARIATIONS. THE FOLLOWING EXAMPLE WILL ILLUSTRATE SOME SIMPLE BASIC STATEMENTS IN THE PROPER PROGRAM STRUCTURE.

```
REM PROGRAM TO COMPUTE GAS MILEAGE
REM M=MILES TRAVELED, G=GAS USED
READ M,G
LET T=M / G
PRINT 'MILES TRAVELED', 'GAS USED', 'MILES/GAL'
PRINT M,G,T
DATA 500,25
END
```

OUTPUT PRODUCED IS:

```
MILES TRAVELED     GAS    MILES/GAL
500               25      20
```

AS YOU CAN SEE FROM THE ABOVE SAMPLE, THE PROGRAM STRUCTURE CONSISTS OF BASIC LANGUAGE STATEMENTS FOLLOWED BY AN END STATEMENT. THE WORDS : REM, READ, LET, PRINT, DATA, AND END ARE KEY WORDS THAT MAKE UP A BASIC STATEMENT.

Most of the key words used in the basic statements are self-explanatory:

- REM allows remarks/comments
- READ M, G assigns numbers in the data statement to the variables M and G
- LET assigns the result of M divided by G into variable T
PRINT 'STRING' CAUSES THE STRING IN SINGLE QUOTES TO BE PRINTED LITERALLY
END TELLS THE COMPUTER THAT THE INPUT PROGRAM IS TO BE EXECUTED

IF AT THIS POINT YOU WOULD LIKE TO RUN THE SAMPLE PROGRAM TO GAIN SOME CONFIDENCE IN THE COMPUTER AND ITS ABILITY TO PROVIDE SPEEDY RESULTS, THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION WILL CONTINUE .

B. PROGRAM FORMAT

A BASIC PROGRAM CONSISTS OF A SEQUENCE OF BASIC STATEMENTS , ONE STATEMENT PER INPUT LINE , FOLLOWED BY AN END STATEMENT . BECAUSE NO BASIC STATEMENT MAY BE LONGER THAN ONE INPUT LINE (80 SPACES) , THERE IS NO PROVISION FOR CONTINUING STATEMENTS FROM ONE LINE TO THE NEXT . HOWEVER ; YOU MAY SPACE THE INPUT LINE AS DESIRED FOR READABILITY SINCE THE COMPUTER IGNORES BLANKS IN BASIC .

1. STATEMENT NUMBERS
EACH BASIC STATEMENT MAY HAVE AN OPTIONAL STATEMENT NUMBER PRECEEDING IT FOR IDENTIFICATION PURPOSES . THIS STATEMENT NUMBER MUST BE AN INTEGER BETWEEN 1 -> 9999 FOR EXAMPLE : 12 READ M,G

2. KEY WORDS
THE KEY WORDS THAT MAKE UP A BASIC STATEMENT (REM ,READ ,LET ,ETC ,) ARE SPECIAL TERMINAL SYMBOLS THAT ARE RECOGNIZED BY THE COMPUTER AND FOR THIS REASON THEY MUST BE SPELLED CORRECTLY AND ONLY USED IN BASIC STATEMENTS . THE END STATEMENT INDICATES THAT THE INPUT PROGRAM IS COMPLETED AND THAT PROGRAM EXECUTION IS TO BEGIN . THE 'END' STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM .

YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT WHAT YOU HAVE JUST LEARNED .

1.) IS THIS A LEGAL BASIC PROGRAM(REPLY : YES OR NO)??

73
REM ONE LINE DO NOTHING PROGRAM
END

yes

YES , THE SIMPLEST BASIC PROGRAM CONSISTS OF JUST AN 'END' STATEMENT.

2.) WHICH OF THE FOLLOWING BASIC STATEMENTS IS IN THE PROPER FORMAT:
   A.) IS LET T=M/G
   B.) 1SLET T=M/G
   C.) IS LET T=M /G

REPLY A, B, C OR ALL.

ALL

ALL OF THE ABOVE BASIC STATEMENTS ARE CORRECT BECAUSE SPACES ARE DISREGARDED. NOTE THAT STATEMENT B., ALTHOUGH CORRECT IS CONFUSING TO READ. BLANKS AND INDENTATIONS MAKE A PROGRAM EASY TO READ.

G. ALPHA-NUMERIC CHARACTERS

ALPHA-NUMERIC CHARACTERS ARE THE LEGAL CHARACTERS, DIGITS, AND SPECIAL CHARACTERS THAT CAN BE USED IN BASIC.

1. CHARACTERS

CHARACTERS CONSIST OF THE LETTERS IN THE ALPHABET A-->Z

2. DIGITS

DIGITS ARE THE SINGLE NUMBERS 0-->9

3. SPECIAL CHARACTERS

THE SPECIAL CHARACTERS ARE ** * / * - ( ) = ' $

4. STRING

A STRING IS ANY LIST OF ALPHA-NUMERIC CHARACTERS ENCLOSED IN 'SINGLE' QUOTES. FOR EXAMPLE:

'THIS IS A STRING'

80
D. VARIABLES

IN BASIC THERE ARE THREE TYPES OF VARIABLES: SIMPLE, ALPHA AND SUBSCRIPTED. SUBSCRIPTED VARIABLES WILL BE COVERED IN LESSON 5.

1. SIMPLE VARIABLES

SIMPLE VARIABLES ARE IDENTIFIED BY A SINGLE LETTER OR A SINGLE LETTER FOLLOWED BY A DIGIT BETWEEN 0—9.

FOR EXAMPLE: A, A3, Z0, AND X ARE LEGAL VARIABLES; BUT A26, 92, ABC, AND X1Y ARE ILLEGAL VARIABLES. THEREFORE; YOU HAVE 216 SIMPLE VARIABLES FOR USE IN YOUR PROGRAMS.

2. ALPHA VARIABLES

ALPHA VARIABLES ARE USED FOR ALPHA-NUMERIC MANIPULATIONS IN WHICH A GROUP OF ALPHA-NUMERIC CHARACTERS, CALLED A STRING, ARE REPRESENTED BY AN ALPHA VARIABLE. THE ALPHA VARIABLE CONSISTS OF A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN, $.

THE MAXIMUM LENGTH OF THE ALPHA-NUMERIC STRING ASSIGNED TO THE ALPHA VARIABLE IS 16 CHARACTERS.

FOR EXAMPLE: A$, X$, X$ ARE LEGAL ALPHA VARIABLES.

A SIMPLE USE OF ALPHA VARIABLES FOLLOWS:

READ A$, B$
PRINT A$, B$
DATA 'MONDAY', '21 JUNE'
END

THIS PROGRAM WILL PRODUCE THE OUTPUT:

MONDAY 21 JUNE

E. NUMBERS

NUMBERS MAY BE EXPRESSED AS INTEGERS OR AS REALS, AND A NUMBER WHETHER INTEGER OR REAL IS LIMITED TO 9 DIGITS, NOT INCLUDING DECIMAL POINT. A NUMBER IS ASSUMED POSITIVE UNLESS IT IS PRECEDED BY A - SIGN.

1. INTEGERS

INTEGERS ARE NUMBERS WITH NO FRACTIONAL PART, i.e., 3, 15
2. REALS

REAL (FIXED-POINT) NUMBERS HAVE A DECIMAL POINT AND A FRACTIONAL PART, IE. 3.0, 15.31, 729.1, 0.0

F. EXPRESSIONS

AN EXPRESSION MAY BE A SINGLE CONSTANT OR VARIABLE, OR AN ARITHMETIC EXPRESSION. ARITHMETIC EXPRESSIONS ARE FORMED BY USING OPERATORS AND PARENTHESIS.

1. OPERATORS

NUMBERS AND SIMPLE VARIABLES MAY BE COMBINED INTO ARITHMETIC EXPRESSIONS BY USING ONE OR MORE OF THE ARITHMETIC OPERATORS:

- ** EXPONENTIATION
- * MULTIPLICATION
- / DIVISION
- + ADDITION
- - SUBTRACTION

IN WRITING AN EXPRESSION, EACH ARITHMETIC OPERATION MUST BE SHOWN. NO TWO OPERATIONS MAY BE ADJACENT, NOR MAY TWO NUMBERS OR VARIABLES BE ADJACENT IN AN EXPRESSION.

FOR EXAMPLE: 2*2, A/B, X+C, X+2, AND Y-3.0 ARE LEGAL EXPRESSIONS.

2. PARENTHESIS

PARENTHESIS MAY BE USED TO GROUP EXPRESSIONS AND TO CONTROL THE ORDER IN WHICH AN ARITHMETIC EXPRESSION IS EVALUATED.

THE NORMAL HIERARCHY OF OPERATORS IS:

1. **
2. * AND /
3. + AND -

IF PARENTHESIS ARE USED, THE EXPRESSION WITHIN A PARENTHESIS PAIR IS EVALUATED FIRST. PARENTHESIS IN EXPRESSIONS MUST ALWAYS OCCUR IN PAIRS, IE. 5*(4+3), (5*(4+3)) ARE EQUIVALENT EXPRESSIONS HAVING THE VALUE 35.

OPERATIONS ARE EVALUATED ACCORDING TO THE OPERATORS HIERARCHY IF TWO OPERATORS OF THE SAME HIERARCHY OCCUR IN AN EXPRESSION EVALUATION IS FROM LEFT TO RIGHT.

FOR EXAMPLE: 2*4 + 6/3

IN THIS EXPRESSION 2*4 WILL BE EVALUATED FIRST, THEN 6/3, AND THEN THE TWO RESULTS WILL BE ADDED.
YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT WHAT YOU HAVE JUST LEARNED.

1.) 6 + 6 / 2 HAS THE VALUE (REPLY WITH VALUE)

5

YOUR ANSWER IS INCORRECT. THE EXPRESSION IS EVALUATED AS FOLLOWS:

6 + 6 / 2 → 6 + 3 → 7

2.) (4*6)/2 HAS THE VALUE (REPLY WITH VALUE)

5

3.) ((4*6)/2)**2 HAS THE VALUE (REPLY : VALUE)

25

F. SUMMARY

THIS CONCLUDES THE INSTRUCTION SET FOR LESSON 1.
YOU WILL NOW BE GIVEN A SAMPLE BASIC PROGRAM AND ASKED TO FIND THE MISTAKES IN IT, CONCERNING WHAT YOU HAVE LEARNED IN THIS LESSON:

1 REM REVIEW PROGRAM
2 READ A1,B2,Z1,$Y
3 LET A1=(A1+Z1)
4 LET B2=(A2+3)+Z1**2)
5 LET Z1=3.141592763
6 PRINT I1,$2,21
7 DATA 5.0,3,3.1418,'FINI'

AFTER LOOKING AT THE SAMPLE PROBLEM, YOU WILL BE ASKED QUESTIONS ABOUT EACH STATEMENT.

1.) LINE 2 CONTAINS A READ STATEMENT FOLLOWED BY A LIST OF VARIABLES. ANY ERRORS (REPLY: YES OR NO)?

YES
INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY.

yes

2.) LINE 3 CONTAINS A LET STATEMENT FOLLOWED BY A1 = EXPRESSION. ANY ERRORS (REPLY: YES OR NO)?

yes

INPUT CORRECTION TO ILLEGAL EXPRESSION; EVERYTHING AFTER = SIGN.

b2*(a1+z1)

3.) LINE 4 IS SIMILAR TO LINE 3. ANY ERRORS?

no

4.) LINE 5 CONTAINS A LET STATEMENT FOLLOWED BY A NUMBER. ANY ERRORS (REPLY: YES OR NO)?

no

THE NUMBER CONTAINS 10 DIGITS AND THE MAXIMUM ALLOWED IS 9 DIGITS.

5.) LINE 6 CONTAINS A PRINT STATEMENT FOLLOWED BY A LIST OF VARIABLES. ANY ERRORS (REPLY: YES OR NO)?

yes

INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY.

a1

THERE ARE NO ERRORS IN LINE 7. IS THE PROGRAM READY TO EXECUTE (ASSUMING ABOVE ERRORS CORRECTED) (REPLY: YES OR NO)

yes

THE PROGRAM FORMAT REQUIRES THAT AN END STATEMENT
BE THE LAST STATEMENT OF THE PROGRAM. THUS THIS SAMPLE PROGRAM
WOULD NOT EXECUTE.

THIS CONCLUDES THE REVIEW QUESTIONS
FROM LESSON 1.

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION?
(REPLY: YES OR NO)

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY: YES; OTHERWISE REPLY: NO AND THE INSTRUCTION
WILL CONTINUE.

no
EXECUTION BEGINS...

*** LESSON 2. ***

THIS INSTRUCTION SEQUENCE WILL INTRODUCE YOU TO THE BASIC LANGUAGE STATEMENTS REM, PRINT, READ, AND DATA. USING THESE STATEMENTS YOU WILL BE ABLE TO CONSTRUCT AND EXECUTE ELEMENTARY PROGRAMS.

THE FORM FOR EACH BASIC STATEMENT WILL INCLUDE:

'KEY WORD' << ELEMENTS OR LIST OF ELEMENTS SEPARATED BY COMMAS >>

THE CARET SYMBOLS '<< >>' DELINIMATE THE LEGAL ITEMS THAT MAY FOLLOW THE KEYWORD AND MAKE UP THE BASIC STATEMENT.

A. REM

THE BASIC STATEMENT WHICH ALLOWS YOU TO INSERT REMARKS INTO YOUR PROGRAM IS IDENTIFIED BY THE KEY WORD 'REM'. REM IS A NON-EXECUTING STATEMENT WHICH MAY BE USED OPTIONALLY AT ANY PLACE IN YOUR PROGRAM TO INTRODUCE A PROGRAM NAME, TO EXPLAIN VARIABLES, TO DOCUMENT YOUR PROGRAM, ETC.

THE FORM FOR THE REM STATEMENT IS:

REM << ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS >>

THE REM STATEMENT IS IGNORED BY THE CAIBASIC COMPILER, AND IS ONLY FOR YOUR INFORMATION. FOR EXAMPLE:

REM PROGRAM TO COMPUTE INCOME TAX

B. PRINT

THE PRINT STATEMENT IS THE METHOD OF WRITING OUT THE RESULTS OF THE BASIC PROGRAM, TO DISPLAY VALUES OF VARIABLES, TO LABEL STATEMENT IS:

PRINT << EXPRESSION OR 'STRING', . . . . , EXPRESSION OR 'STRING' >>

MULTIPLE ELEMENTS IN THE PRINT LIST ARE SEPARATED BY COMMAS.

I. PRINT << EXPRESSION >> WILL WRITE OUT THE CURRENT VALUE OF THE EXPRESSION, WHERE AN EXPRESSION AS DEFINED IN LESSON 1 WAS A NUMBER, A VARIABLE, OR AN ARITHMETIC
EXPRESSION THAT IS TO BE EVALUATED.

FOR EXAMPLE:

PRINT 1, A, B, 5**2

ASSUMING THAT A=10.0, B=15.5 WOULD PRINT:

1 10.0 15.5 25

2. PRINT <<'STRING'>> WILL PRINT OUT ALL THE ALPHA-NUMERIC CHARACTERS OF THE STRING WITHIN 'SINGLE' QUOTES. THIS FORM IS USED FOR LABELING THE COMPUTER OUTPUT.

FOR EXAMPLE: PRINT 'THE ANSWER IS:' PRODUCES THE RESULT:

THE ANSWER IS:

3. PRINT BY ITSELF IS USED TO SKIP A LINE ON THE COMPUTER OUTPUT.

THE COMPUTER OUTPUT SHEET IS DIVIDED INTO 8 ZONES EACH 15 COLUMNS WIDE. PRINT ZONES CAN BE SKIPPED BY PUTTING A BLANK IN THE PRINT LIST. FOR EXAMPLE:

PRINT A, #, X

OUTPUTS THE VALUE OF A IN THE FIRST ZONE, # IN THE SECOND, SKIPS THE THIRD ZONE, AND PUTS X IN THE FOURTH ZONE.

ALPHA VARIABLES AND 'STRINGS' MAY EXTEND OVER SEVERAL ZONES, BUT NUMERIC RESULTS ARE LEFT ADJUSTED IN THE SPECIFIED ZONE. IF MORE THAN EIGHT ITEMS OCCUR IN THE PRINT LIST, THE ITEMS WILL OVERFLOW AND BE PRINTED ON THE NEXT LINE.

USING THE PRINT AND END STATEMENT YOU NOW HAVE THE FACILITY TO EXECUTE YOUR FIRST PROGRAMS. FOR EXAMPLE:

REM PROGRAM TO COMPUTE THE SQUARE OF A NUMBER
PRINT 'SQUARED = ', 5**2
END

PRODUCES THE RESULT:

S SQUARED = 25

YOU WILL NOW BE GIVEN TWO PROBLEMS TO SOLVE. YOU WILL ENTER
THE EXECUTION PHASE OF CAIBASIC WHERE YOU CAN RUN YOUR PROBLEMS AND THEN YOU WILL RETURN TO FINISH THE LESSON.

1) FIND THE SQUARE ROOT OF $S^2 - 4 \times 2 \times 2$.

2) FIND THE VALUE OF $3.1416(8 \text{ CUBED})/12$ WHERE $B=2.50$, $H=3.03$.

IF YOU WISH TO SKIP THESE PROBLEMS REPLY: YES AND THE LESSON WILL CONTINUE.

C. READ
THE READ STATEMENT IS THE METHOD WHICH PROVIDES INPUT TO THE PROGRAM. THE FORM OF THE READ STATEMENT IS:

READ << VARIABLE, ..., VARIABLE >>

FOR EVERY VARIABLE IN THE READ LIST THERE MUST BE A CORRESPONDING ELEMENT IN A DATA STATEMENT. THE READ AND DATA STATEMENTS ARE USED TOGETHER TO ASSIGN INPUT VALUES TO PROGRAM VARIABLES. WHEN THE READ STATEMENT IS EXECUTED, EACH VARIABLE IS ASSIGNED SUCCESSIVE NUMBERS FROM A STACK OF NUMERIC DATA OR SUCCESSIVE 'STRINGS' FROM A STACK OF ALPHA-NUMERIC DATA. AS EACH VARIABLE IS READ, IT TAKES THE TOP ELEMENT OF THE APPROPRIATE DATA STACK.

D. DATA
THE DATA STATEMENT IS A LIST OF INPUT NUMBERS OR 'STRINGS' THAT WILL BE ASSIGNED TO VARIABLES IN A READ STATEMENT. THE FORM OF THE DATA STATEMENT IS:

DATA << NUMBER OR 'STRING', ..., NUMBER OR 'STRING' >>

THE 'STRING' OF ALPHA-NUMERIC CHARACTERS MUST BE ENCLOSED IN SINGLE QUOTES. DATA STATEMENTS MAY BE PLACED ANYWHERE IN A PROGRAM, BUT THERE IS AN UPPER LIMIT OF 500 NUMERIC AND 500 ALPHA-NUMERIC DATA ELEMENTS FOR EACH PROGRAM.
WHEN THE FIRST DATA STATEMENT IS INTERPRETED BY THE CAIASCIC COMPILER A FIRST IN , FIRST OUT STACK IS FORMED FOR NUMERIC AND ALPHA-NUMERIC DATA . AS EACH NUMBER OR STRING IN A DATA LIST IS INTERPRETED , IT IS PLACED ON THE BOTTOM OF ITS RESPECTIVE DATA STACK . AS OTHER DATA STATEMENTS ARE LOCATED IN THE PROGRAM ITS ELEMENTS ARE PLACED ON THE BOTTOM OF THE PROPER stack .

1.) EXAMPLE :
DATA 10.0,13.33,'J.E.SMITH',18
DATA 'Z.X. DOE',18

PRODUCES THE FOLLOWING DATA STACKS :
NUMERIC       ALPHA-NUMERIC
10.0           J.E.SMITH
13.33          Z.X. DOE
18
19


2.) EXAMPLE :
READ A,B1,Z1
ASSUMING THAT THE DATA FROM EXAMPLE 1.) IS AVAILABLE , THE VARIABLES IN THE READ LIST ARE ASSIGNED VALUES AS FOLLOWS :
A <- 10.0
B1 <- 13.33
Z1 <- J.E.SMITH

THE RESULTING DATA STACKS ARE AS FOLLOWS :
NUMERIC       ALPHA-NUMERIC
18             Z.X. DOE
19

C. RESTORE , RESTORES

THE RESTORE , RESTORES STATEMENTS ARE USED TO RETURN THE NUMERIC AND ALPHA-NUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO
That the data may be used again, the form of the restore statement is:

```
RESTORE ( RESTORES NUMERIC DATA )
RESTORE$ ( RESTORES ALPHA-NUMERIC DATA )
```

You will now be asked some questions about the read, data and restore statements:

Consider the following statements as part of a basic program:

```
DATA 'ANS='
DATA 'CORRECT',25,'WRONG',0.0
READ A,B3,IS,CI,D
READ J6,E4,KS,X
```

1.) What is the value of C1? Reply with value

Now consider that the following statements were added to the above program:

```
RESTORE
READ Z4,A
```

2.) What is the value of Z4? Reply with value.

The correct answer is Z4=2. The restore command returns the numeric data stack to its original condition, and the read statement assigns values from the top of the data stack to the variables in the read list as follows:

```
Z4 <- 2
A <- 5
```

F. Summary

Now that you have seen how read and data statements work together to input values into your program, and how the print statement is used to output and label results you have the facility to write simple programs using input data.

For example

```
(( 8**2)**.5)
```

Could be written in symbolic form and the data
COULD BE READ IN AS FOLLOWS:

```plaintext
READ A,B
PRINT'ANS=',((A+B)**.5)
DATA 8,2
END
```

IN THE NEXT LESSON YOU WILL BE SHOWN HOW TO USE ASSIGNMENT STATEMENTS
THIS WILL GIVE YOU GREATER FLEXIBILITY IN WRITING EXPRESSIONS
AND WILL ALLOW YOU TO DO ASSIGNMENTS SUCH AS:

THE RESULTS OF BOTH EXPRESSIONS WOULD BE DISPLAYED.

SO

YOU WILL NOW BE GIVEN SOME REPRESENTATIVE PROBLEMS
TO GIVE YOU A CHANCE TO EXERCISE YOUR NEW PROGRAMMING TOOLS

1.) WRITE A PROGRAM TO SOLVE THE EQUATION X**2+10Y=24
WHERE THE INPUT DATA IS X=10 ,Y=3.

2.) WRITE A PROGRAM TO SOLVE THE QUADRATIC EQUATION

\((-B+\sqrt{B^2-4AC})/2A\)

INPUT DATA IS \(A=2,B=5,C=2\).

IF YOU WISH TO WRITE AND EXECUTE THESE PROGRAMS NOW, REPLY: YES
AND YOU WILL ENTER THE C/C++ BASIC COMPILER; OTHERWISE REPLY: NO
AND YOU WILL GO ON TO THE NEXT LESSON.

IF YOU DECIDE TO RUN THESE PROBLEMS LATER THE ANSWERS
WILL NOW BE GIVEN SO YOU MAY CHECK YOUR RESULTS:

1.) 106
2.) -.500

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION?
(REPLY: YES OR NO)

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY: YES; OTHERWISE REPLY: NO AND THE INSTRUCTION
WILL CONTINUE.

85
EXECUTION BEGINS...

*** LESSON 5. ***

THIS INSTRUCTION SET WILL INTRODUCE YOU TO ASSIGNMENT STATEMENTS AND BUILT-IN FUNCTIONS. THE LET STATEMENT AND THE 10 BUILT-IN FUNCTION WILL ENABLE YOU TO EVALUATE AND ASSIGN VARIABLES TO COMPLEX ARITHMETIC EXPRESSIONS.

A. LET

THE LET STATEMENT IS AN ASSIGNMENT OR SUBSTITUTION COMMAND. IT CAUSES THE EVALUATION OF AN EXPRESSION TO BE SUBSTITUTED FOR THE CURRENT VALUE OF A VARIABLE. THE FORM OF THE LET STATEMENT IS:

\[
\text{LET} \langle \text{Variable} \rangle = \langle \text{Expression} \rangle \quad \text{OR} \quad \text{LET} \langle \text{Variable} \rangle = \langle \text{Variable} \rangle \ldots \langle \text{Variable} \rangle = \langle \text{Expression} \rangle
\]

THERE MAY BE ANY NUMBER OF VARIABLE = VARIABLE IN THE FORM OF THE LET STATEMENT. AN EXPRESSION IS A NUMBER, VARIABLE, OR AN ARITHMETIC EXPRESSION.

WHEN THE LET STATEMENT IS EXECUTED THE EXPRESSION ON THE RIGHT SIDE OF THE EQUAL SIGN IS EVALUATED AND THE RESULTING VALUE IS ASSIGNED TO THE VARIABLE OR VARIABLES ON THE LEFT SIDE OF THE EQUAL SIGN. THE PREVIOUS VALUE ASSIGNED TO THE VARIABLE OR VARIABLES WILL BE LOST. FOR EXAMPLE:

\[
\text{LET } A = 12.5 \\
\text{LET } B = 13.4 \\
\text{LET } A = 25
\]

THE VALUE OF THE VARIABLES A AND B IS NOW \( A = 25 \).

THE ONLY RESTRICTION ON THE USE OF VARIABLES IS THAT SIMPLE AND SUBSCRIPTED VARIABLES CAN ONLY BE ASSIGNED NUMERIC VALUES, AND ALPHA VARIABLES CAN ONLY BE ASSIGNED ALPHA-NUMERIC STRINGS. FOR EXAMPLE:

\[
\text{LET } A = \text{D} = \pi \times 5 = (3 + 2 - 6) / 4 \\
\text{LET } A = \text{HELP}
\]
LET Y=(B - 4*A*C)**.5
LET US='ANSWER ='
THE FOLLOWING ASSIGNMENT IS ILLEGAL :
LET A=D$=(S+4)

YOU WILL NOW BE ASKED QUESTIONS CONCERNING
WHAT YOU HAVE JUST LEARNED :

1.) REPLY WITH VALUE OF X IN BELOW PROGRAM .
   LET A=-4
   LET B=6
   LET C=3
   LET X=A**2 - 4*B + 3*C
   PRINT 'ANSWER =', X
   END

2.) REPLY WITH VALUE OF Z IN BELOW PROGRAM .
   DATA |#|f |# | S |# | l
   DATA 'RIGHT ON',7,8
   READ A,B,X
   READ GS,Y,C,D
   LET ?*X+Y
   END

YOUR RESPONSE WAS INCORRECT , THE VARIABLES ARE
ASSIGNED VALUES AS FOLLOWS :
   NUMERIC  ALPHA
   A<- 1    Y$<- RIGHT ON
   B<- 2
   X<- 3
   Y<- 4
   C<- 6
   D<-7
   Z <- X+Y ; Z <- 7

3.) LET R= A*B/C-D
   WHICH FORMULA DOES THIS STATEMENT REPRESENT ?
EPLY WITH CORRECT LETTER

A.) \( R = \frac{(A+B)}{(C-D)} \)

B.) \( R = A^*\left(\frac{B}{C}\right) - D \)

8. BUILT-IN FUNCTIONS

BUILT-IN FUNCTIONS ARE COMMONLY USED PROGRAMS ALREADY WRITTEN AND STORED IN THE CARIBASIC COMPILER FOR YOUR USE. THERE ARE FUNCTIONS TO FIND SQUARE ROOTS, LOGARITHMS, ABSOLUTE VALUES, AND TRIGONOMETRIC VALUES. THE FORM FOR THE BUILT-IN FUNCTIONS IS:

FUNCTION NAME (EXPRESSİON)

WHERE THE EXPRESSION IS ENCLOSED IN PARENTHESES.

THE BUILT-IN FUNCTIONS AND DEFINITIONS ARE:

- SQR(X) --- SQUARE ROOT OF ARGUMENT (MUST BE POSITIVE)
- ABS(X) --- ABSOLUTE VALUE OF ARGUMENT
- LOG(X) --- NATURAL LOGARITHM OF ARGUMENT
- EXP(X) --- EXPONENTIAL FUNCTION, VALUE OF 2.71828 ** X
- INT(X) --- INTEGER PART OF ARGUMENT IS RETURNED
- SIN(X) --- SINE OF THE ARGUMENT
- COS(X) --- COSINE OF ARGUMENT
- TAN(X) --- TANGENT OF ARGUMENT
- ATN(X) --- ARCTANGENT IN RADIANS OF ARGUMENT

IN ALL THE ABOVE BUILT-IN FUNCTIONS THE ARGUMENT IS ANY LEGAL EXPRESSION; AND AS NOTED THE SQR FUNCTION REQUIRES A POSITIVE ARGUMENT. THE TRIGONOMETRIC FUNCTIONS REQUIRE AN ARGUMENT VALUE IN RADIANS.

THE BUILT-IN FUNCTIONS ARE USED BY SIMPLY CALLING THEM WITH THE APPROPRIATE FUNCTION NAME AND ARGUMENT. THESE BUILT-IN FUNCTIONS ARE CONSIDERED TO BE EXPRESSIONS, AND THEY MAY BE USED ANY PLACE WHERE AN EXPRESSION IS LEGAL, FOR EXAMPLE:

LET Z = SQR(ABS(-5)) IS A CORRECT USE OF BUILT-IN FUNCTIONS.
THE FOLLOWING PROGRAM IS AN EXAMPLE OF HOW TO USE BUILT-IN FUNCTIONS:

```plaintext
REM PROGRAM TO COMPUTE SQUARE ROOT AND LOGARITHMS
READ A
LET Y=SQR(A)
LET Z=LOG(A)
PRINT 'SQUARE ROOT=',Y,'LOG=',Z
DATA 4,6,8
END
```

PRODUCES RESULT
SQUARE ROOT= 2.0
LOG= 1.386

YES

YOU WILL NOW BE ASKED QUESTIONS CONCERNING WHAT YOU HAVE JUST LEARNED:

1.) IS THE FOLLOWING STATEMENT LEGAL
REPLY: YES, OR NO.
LET Z1=\(4*\text{SQR}(3.0*\text{EXP}(3.3))\)

YES

2.) IS THE FOLLOWING SEQUENCE OF PROGRAM STATEMENTS LEGAL
REPLY: YES OR NO.
LET B=9
LET X=SQR(B)
...
...

NO

THE SQUARE ROOT OF A NEGATIVE NUMBER IS AN UNDEFINED OPERATION. IN THE NEXT LESSON YOU WILL BE SHOWN A BASIC STATEMENT FOR TESTING AND BRANCHING TO ANOTHER SEGMENT OF THE PROGRAM IF THE TEST IS TRUE. FOR EXAMPLE THE ABOVE PROGRAM SEQUENCE MIGHT BE ALTERED AS FOLLOWS:

```plaintext
LET B=9
IF B LT 0 THEN 100
SO LET X=SQR(X)
...
```
100 REM NEGATIVE ARGUMENT
LET B=ABS(B)
GO TO 50

THIS PROGRAM SEQUENCE TESTS FOR A NEGATIVE ARGUMENT, IF TRUE
IT BRANCHES TO STATEMENT NUMBER 100, MAKES THE ARGUMENT POSITIVE
AND BRANCHES BACK TO STATEMENT 50 TO COMPLETE THE PROGRAM.

C. SUMMARY
WITH THE LET STATEMENT AND BUILT-IN FUNCTIONS, PLUS THE PREVIOUS
BASIC STATEMENTS (REM, READ, DATA, PRINT), YOU ARE
FAST GAINING AN EFFECTIVE REPERTOIRE FOR PROGRAMMING USE. IN
THE NEXT LESSON YOU WILL LEARN HOW TO SET UP LOOPS IN A PROGRAM
SO THAT THE MAIN BODY OF A PROGRAM MAY BE EXECUTED AS OFTEN
AS DESIRED. AS YOU ARE DOING YOUR REVIEW PROBLEMS, THINK ABOUT
HOW YOU COULD SET UP A LOOP TO READING ANY AMOUNT OF DATA,
PROCESS IT AND THEN HALT FOR SOME TEST CONDITION.

THE FOLLOWING REVIEW PROBLEMS WILL EXERCISE
YOUR PROGRAMMING SKILLS TO DATE:

1.) WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR
SOME NUMBER OF YEARS HENCE. THE FORMULA IS:

\[ P = S \left( \frac{1}{(1+i)^n} \right) \]

WHERE \( P \) IS THE PRESENT WORTH OF A PAYMENT \( S \) IN \( n \) YEARS HENCE
AT AN INTEREST RATE OF \( i \). FOR DATA USE \( i=0.08 \), \( S=5000 \), \( n=20 \).

2.) WRITE A PROGRAM TO FIND SIDES \( a \) AND \( c \) OF A TRIANGLE USING
THE LAW OF SINES FORMULA:

\[ \frac{a}{\sin(A)} = \frac{b}{\sin(B)} = \frac{c}{\sin(C)} \]

WHERE THE NUMERATOR IS THE SIDE AND THE DENOMINATOR IS THE
SINE OF THE ANGLE, THE CONVERSION FACTOR FROM DEGREES TO RADIANS IS:

\( 1 \) DEGREE \( \times \left( \frac{\pi}{180} \right) \) RADIANS. THE DATA FOR THE PROGRAM IS:
SIDE \( a=34.91 \) IN. ANGLE \( A=98.71 \) DEG.
SIDE \( b=44.97 \) DEG. ANGLE \( B=31.32 \) DEG.

IF YOU WISH TO RUN THESE PROGRAMS NOW, THEN REPLY : YES
OTHERWISE REPLY : NO.
THE ANSWERS TO THE PROBLEMS WILL NOW BE GIVEN SO
THAT YOU MAY CHECK YOUR RESULTS LATER:

1.) $1072.7$
2.) SIDE A=45.06 IN. AND SIDE C=23.69 IN.

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION?
(REPLY: YES OR NO)

YES
R; T=4.84/16.96 14.03.44
EXECUTION BEGINS...

*** LESSON 4 ***

THIS INSTRUCTION SEQUENCE WILL COVER BRANCHES. AS YOU HAVE SEEN FROM PREVIOUS PROGRAMS, A PROGRAM'S EXECUTION USUALLY TAKES A DIRECT ROUTE FROM THE FIRST TO THE LAST STATEMENT. THE CONDITION THAT ALLOWS DETOURS TO OCCUR IN PROGRAMS IS CALLED BRANCHING. THERE ARE TWO TYPES OF BRANCHING: UNCONDITIONAL AND CONDITIONAL.

80

A. UNCONDITIONAL BRANCHES
AN UNCONDITIONAL BRANCH IS AN IMPERATIVE TRANSFER OF CONTROL FROM ONE POINT IN A PROGRAM TO ANOTHER. THERE ARE TWO FORMS OF THE UNCONDITIONAL BRANCH: THE 'GO TO' AND THE 'COMPUTED GO TO'

1.) GO TO << STATEMENT NUMBER >>

THIS COMMAND TRANSFERS PROGRAM CONTROL DIRECTLY TO THE STATEMENT NUMBER AND CONTINUES EXECUTION FROM THAT POINT. THE 'GO TO' IS USED FOR FORMING LOOPS IN A PROGRAM.

80

A SAMPLE LOOP FOLLOWS:

REM PROGRAM TO COMPUTE PRESENT WORTH
REM P=INVESTMENT, S=PRINCIPAL, I=INTEREST RATE, N=NR.YEARS
PRINT INVESTMENT', 'PRINCIPAL', 'INTEREST', 'NR.YEARS'
READ S,I,N
LET P=S*(1/(1+(I*I)**N))
PRINT S,I,N
GO TO 10
DATA 5000,.08,20.5
END

PRODUCES THE OUTPUT:

INVESTMENT  PRINCIPAL  INTEREST  NR.YEARS
1072.74  5000  .08  20
2315.97  5000  .08  10
2791.99  5000  .06  10

*** ERROR, YOU TRIED TO READ MORE NUMERIC DATA THAN YOU PUT IN ***
THE ERROR OCCURS BECAUSE YOU RUN OUT OF DATA DURING THE EXECUTION OF THE LOOP SET-UP BY THE UNCONDITIONAL TRANSFER. IF THE READ STATEMENT WERE NOT IN THE LOOP TO CAUSE THE PROGRAM TO STOP, THEN YOU WOULD BE IN AN "INFINITE LOOP," A CONDITION IN WHICH THERE IS NO WAY TO STOP. YOU MUST ALWAYS CHECK FOR THE "INFINITE LOOP" CONDITION BY MAKING SURE THAT YOUR PROGRAM HAS AN EXIT.

2) ON << EXPRESSION >> GO TO << STATEMENT NUMBER, STATEMENT NUMBER >>

THIS SPECIAL FORM OF THE "GO TO" COMMAND IS CALLED THE "COMPUTED GO TO", THE EXPRESSION IN THE FORM OF THE STATEMENT MUST EVALUATE TO AN INTEGER NUMBER BETWEEN 1--9999. IF IT IS NOT AN INTEGER, OR OUTSIDE THIS RANGE AN ERROR WILL OCCUR. WHEN THE "COMPUTED GO TO" IS EXECUTED THE EXPRESSION IS EVALUATED, AND PROGRAM CONTROL TRANSFERS TO THE N-TH STATEMENT NUMBER, WHERE N-TH REPRESENTS THE VALUE OF THE EXPRESSION. FOR EXAMPLE:

LET I=3
ON I GO TO 100, 33, 475, 9999

EXECUTION OF THE "COMPUTED GO TO" WOULD CAUSE PROGRAM CONTROL TO TRANSFER UNCONDITIONALLY TO STATEMENT NUMBER 475. YOU MUST BE CAREFUL WHEN USING THE "COMPUTED GO TO" NOT ONLY BECAUSE OF INFINITE LOOPS; BUT BECAUSE THE EXPRESSION MUST BE AN INTEGER BETWEEN 1--9999, AND THERE MUST BE A STATEMENT NUMBER FOR "ALL" POSSIBLE VALUES OF THE EXPRESSION.

CONDITIONAL BRANCHING

THE CONDITIONAL BRANCH TRANSFERS CONTROL ONLY IF CERTAIN RELATIONS ARE TRUE. IF THE TEST OF RELATIONS IS TRUE THEN TRANSFER OF CONTROL OCCURS; OTHERWISE PROGRAM CONTROL CONTINUES WITH THE NEXT STATEMENT. THE FORM OF THE CONDITIONAL BRANCH IS:

IF << EXPRESSION >> << RELATION >> << EXPRESSION >> THEN << STATEMENT NUMBER >>

NOTE THAT ALPHA VARIABLES ARE NOT ALLOWED AS AN EXPRESSION IN A CONDITIONAL BRANCH.
THE RELATIONS ARE:

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>EXAMPLE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT</td>
<td>A GT B</td>
<td>A GREATER THAN B</td>
</tr>
<tr>
<td>GE</td>
<td>A GE B</td>
<td>A GREATER THAN OR EQUAL TO B</td>
</tr>
<tr>
<td>LT</td>
<td>A LT B</td>
<td>A LESS THAN B</td>
</tr>
<tr>
<td>LE</td>
<td>A LE B</td>
<td>A LESS THAN OR EQUAL TO B</td>
</tr>
<tr>
<td>NE</td>
<td>A NE B</td>
<td>A NOT EQUAL TO B</td>
</tr>
<tr>
<td>=</td>
<td>A = B</td>
<td>A EQUAL B</td>
</tr>
</tbody>
</table>

WHEN THE CONDITIONAL STATEMENT IS EXECUTED THE (EXPRESSION RELATION EXPRESSION) IS TESTED, AND IF THE RELATION IS TRUE, THEN CONTROL IS TRANSFERRED TO THE STATEMENT NUMBER. OTHERWISE PROGRAM CONTROL CONTINUES TO THE NEXT SEQUENTIAL STATEMENT, FOR EXAMPLE:

```plaintext
1 READ A
***
IF A LT 0 THEN 90
LET X=SQR(A)
***
90 PRINT"ILLEGAL ARGUMENT", A
GO TO 1
***
```

THE ONLY TIME THAT THE CONDITIONAL BRANCH IS EXECUTED IS WHEN A IS LESS THAN 0.

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT BRANCHING:
1.) ARE THE FOLLOWING STATEMENTS CORRECT
REPLY: YES OR NO

GO TO END

no

IF XI LT A$ THEN 10
IF(X**5-4) GE 10 THEN GO TO 100

yes

YOUR ANSWER IS INCORRECT. THE ONLY THING ALLOWED AFTER THEN IS A STATEMENT NUMBER.

LET Y=6,
ON Y GO TO 1,3,5,7,999

no

1.) CONSIDER THIS PROGRAM SEGMENT, ANY ERRORS (REPLY: YES OR NO).

1 READ A,B
3 LET Z=A*B
IF Z LT 0 GO TO 999
...    
GO TO 3
DATA 5,5,7,10
999 END

yes

THERE IS AN 'INFINITE' LOOP IN THIS PROGRAM. IT COULD BE CORRECTED BY CHANGING THE GO TO 3 TO GO TO 1.

WHEN YOU WRITE PROGRAMS HAVING BRANCHES OR LOOPS YOU MUST ALWAYS CONSIDER HOW THE PROGRAM WILL STOP. YOU HAVE OBSERVED THAT AN 'INFINITE LOOP' CAN BE STOPPED BY HAVING A READ STATEMENT IN THE LOOP AND JUST RUN OUT OF DATA. HOWEVER, ENDING A PROGRAM ON AN ERROR IS AN INELIGENT METHOD. THE MOST COMMON METHODS USE THE 'GO TO' AND 'IF/THEN' COMMANDS TO CONTROL PROGRAM LOOPS AND ARE AS FOLLOWS:

GO

1. COUNT AND TEST METHOD, IN WHICH A COUNTER IS INCREMENTED IN THE LOOP, AND WHEN THE COUNTER REACHES A CERTAIN VALUE, BRANCH OUT OF THE LOOP. EXAMPLE PROGRAM

REM COUNT AND TEST METHOD
REM N IS COUNTER, INITIALIZED TO 0, AND COUNTS FROM 1--10
LET N=Z=0
READ X
10 LET Z=Z+X
LET N=N+1
IF N GT 10 THEN 100
GO TO 10
100 PRINT 'SUM=',Z
DATA 10
END

REPLY WITH VALUE OF Z.

11

YOUR ANSWER IS INCORRECT. THE ONLY WAY TO BE SURE OF VALUES IN A LOOP IS TO SET UP A TABLE OF THE VARIABLES AND KEEP TRACK OF THEIR VALUES IN THE LOOP:

<table>
<thead>
<tr>
<th>X</th>
<th>N</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>110</td>
</tr>
</tbody>
</table>

THUS Z=110

2. READ ANY VALUE IN AND TEST WHETHER, IN WHICH A VALUE IS READ IN AND TESTED FOR THE END OF LOOP CONDITION.

REM READ AND TEST DEMONSTRATION
REM 9999 IS END VALUE
LET Z=0
1 READ X
IF X=9999 THEN 9999
LET Z=Z+X
GO TO 1
DATA 1,2,3,4,5,6,9999
9999 PRINT 'SUM=',Z
END

REPLY WITH VALUE OF Z

21
C. SUMMARY

YOU HAVE SEEN HOW THE UNCONDITIONAL BRANCHES ('GO TO' AND 'COMPUTED GO TO') TRANSFER PROGRAM CONTROL, AND HOW THE CONDITIONAL BRANCH ('IF/THEN') TESTS FOR TRANSFER OF PROGRAM CONTROL; AND YOU HAVE OBSERVED THE CONTROL OF LOOPS SO THAT A PROGRAM SEGMENT MAY BE REPEATED UNTIL A SPECIFIED CONDITION IS MET. IN THE NEXT LESSON YOU WILL LEARN A BASIC STATEMENT TO CONTROL A LOOP BY THE INCREMENT AND TEST METHOD.

THE FOLLOWING PROBLEMS WILL TEST YOUR NEW SKILLS:

1.) WRITE A PROGRAM TO COUNT THE NUMBERS BETWEEN 50 AND 60, AND ALSO PRINT THEM OUT. THE INPUT DATA IS 10, 50, 35, 75, 60, 54, 54.

2.) WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME YEARS HENCE, AT VARYING INTEREST RATES. THE FORMULA IS: 
   
P=\$5000, AND I=\$0.04--\$0.09, IN INCREMENTS OF \$0.01, AND N=20

IF YOU DESIRE TO EXECUTE THESE PROGRAMS NOW

REPLY: YES; OTHERWISE REPLY: NO

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION?

(REPLY: YES OR NO)

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME

THEN REPLY: YES; OTHERWISE REPLY: NO AND THE INSTRUCTION WILL CONTINUE.

REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO COVER
EXECUTION BEGINS...

*** LESSON 5 ***
THIS LESSON WILL INTRODUCE YOU TO ITERATION, SUBSCRIPTED VARIABLES, AND LISTS (VECTORS) AND TABLES (MATRICES).

A. ITERATION (LOOPING)
IN THE LAST LESSON YOU WERE SHOWN HOW TO USE CONDITIONAL AND UNCONDITIONAL BRANCHES TO CONTROL LOOPING. THE COUNT AND TEST ITERATIVE LOOP OCCURS SO FREQUENTLY THAT AN ABBREVIATED BASIC STATEMENT HAS BEEN DEVISED TO CONTROL LOOPING. THE ITERATIVE LOOP HAS THE FOLLOWING FORM:

FOR << SIMPLE VARIABLE >> = << EXPRESSION >> TO << EXPRESSION >> 
STEP << EXPRESSION >>

***
***
NEXT << SIMPLE VARIABLE >>

FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT:

FOR I=1 TO 10 STEP 2 
LET X=X+1
NEXT I

THE SIMPLE VARIABLE FOLLOWING 'FOR' IS THE LOOP INDEX. WHEN THE FOR/NEXT PAIR IS EXECUTED, THE LOOP INDEX IS GIVEN THE VALUE (INITIALIZED) OF THE FIRST EXPRESSION (I=1 IN EXAMPLE). THIS INDEX IS THEN TESTED TO DETERMINE WHETHER IT IS GREATER THAN THE SECOND EXPRESSION AFTER 'TO'(10 IN EXAMPLE). IF IT IS GREATER, CONTROL IS TRANSFERRED TO THE STATEMENT FOLLOWING 'NEXT'. OTHERWISE THE REMAINING STATEMENTS WITHIN THE LOOP (FOR/NEXT) ARE EXECUTED SEQUENTIALLY UNTIL THE 'NEXT' STATEMENT IS REACHED.

WHEN THE 'NEXT' STATEMENT IS REACHED, THE LOOP INDEX IS INCREASED (INCREMENTED) BY THE AMOUNT OF THE EXPRESSION FOLLOWING 'STEP', AND CONTROL IS TRANSFERRED BACK TO THE 'FOR' STATEMENT WHERE THE LOOP CONTINUES UNTIL THE INDEX VALUE IS
GREATER THAN THE FINAL VALUE. FOR EXAMPLE:

```
REM DEMO LOOP, SUM THE NUMBERS FROM 1-->10.
LET C=0
FOR I=1 TO 10 STEP 1
    LET C=C+1
NEXT I
PRINT'SUM=',C
END
```

So you will notice that the simple variable following 'NEXT' is the same as the simple variable following 'FOR', and that the 'NEXT' statement marks the end of the loop. Because the increment value of a loop is commonly one (1), the 'STEP' modifier and its expression may be omitted, and the increment value will be assumed to be one (+1). For example the above 'FOR' statement could be written:

```
FOR I=1 TO 10
    THE INCREMENT VALUE AFTER 'STEP' MAY BE
    POSITIVE OR NEGATIVE ALLOWING THE FLEXIBILITY OF LOOPING FORWARD
    OR BACKWARD. FOR A NEGATIVE 'STEP' VALUE THE TEST BECOMES
    'LESS THAN'. FOR EXAMPLE THE FOLLOWING 'FOR' STATEMENTS ARE
    EQUIVALENT:
    FOR I=1 TO 10
    FOR I=10 TO 1 STEP -1
```

Another useful technique of looping is 'NESTING'. Nesting refers to placing one loop inside another loop. The inner loop 'SPINS' around as many times as the outer loop is incremented. For example consider this program segment:

```
FOR I=1 TO 10
    FOR J=1 TO 20 STEP 2
        ....
    NEXT J
NEXT I
```

In this example the outside loop (I) is
REPEATED 10 TIMES, AND THE INNER LOOP (J) WOULD BE REPEATED 20 TIMES FOR EACH INCREMENT OF THE OUTSIDE LOOP, OR 200 REPETITIONS LOOPS MAY BE NESTED UP TO A MAXIMUM OF 20; HOWEVER, THEY CANNOT OVERLAP. THE INNERMOST LOOP MUST BE CLOSED WITH ITS 'NEXT' STATEMENT BEFORE ENCOUNTERING THE NEXT OUTER LOOP'S 'NEXT' STATEMENT. FOR EXAMPLE:

```
FOR X=10 TO 1 STEP -1
FOR Y=3 TO 5
FOR Z=-5 TO -5 STEP -1
NEXT Z
NEXT Y
NEXT X
```

WITHIN A FOR/NEXT LOOP CONDITIONAL AND UNCONDITIONAL BRANCHES MAY BE USED TO TRANSFER CONTROL OUT OF A LOOP OR WITHIN LIMITS OF THE SAME LOOP. HOWEVER, IT IS NOT POSSIBLE TO BRANCH INTO THE MIDDLE OF A FOR/NEXT LOOP BECAUSE LOGIC PROBLEMS OCCUR AND AN ERROR WILL RESULT. AN ADDITIONAL ITEM TO BE CAREFUL ABOUT IS USING THE INDEX VARIABLE OF THE FOR/NEXT LOOP IN COMPUTATIONS. IF YOU ALTER THE VALUE OF THE LOOP INDEX YOU WILL EFFECT THE ACTION OF THE LOOP. FOR EXAMPLE:

```
FOR I=1 TO 10
LET I=I+10
NEXT I
PRINT I
END
```

REPLY WITH VALUE OF I THAT IS PRINTED.

11

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT WHAT YOU HAVE JUST LEARNED.

1.) SAMPLE PROGRAM

```
LET S=0
FOR K=-5 TO 5
LET S=S*K
NEXT K
PRINT 'SUM=',S
END
```

REPLY WITH VALUE OF S.

0

2.) SAMPLE PROGRAM

```
FOR I=100 TO 1 STEP 2
LET Z=I**2
LET Y=Z+10
```

REPLY WITH RESULT.
YOUR ANSWER IS INCORRECT. THE INDEX VALUE OF THE LOOP (I=100) IS GREATER THAN THE FINAL VALUE (1) IN THE FIRST TEST. THUS I=100. IF THE LOOP WERE DECREMENTED IN STEPS OF -2 THEN THE LOOP WOULD BE EXECUTED AND THEN I=0.

3. SAMPLE PROGRAM
LET T=0
FOR I=1 TO 5 STEP -1
FOR J=1 TO 5
IF I=J THEN 10
GO TO 15
10 LET T=T+I
PRINT I
15 NEXT J
NEXT I
PRINT T
END

REPLY WITH VALUE OF T.

B. SUBSCRIPTED VARIABLES/LISTS AND TABLES

1. IN LESSON 1 YOU LEARNED THAT SIMPLE VARIABLES WERE A LETTER OR A LETTER FOLLOWED BY A DIGIT. SUBSCRIPTED VARIABLES CONSIST OF A LETTER FOLLOWED BY A SINGLE OR DOUBLE SUBSCRIPT IN PARENTHESES. THE SUBSCRIPTS MAY BE ANY LEGAL EXPRESSION THAT EVALUATES TO AN INTEGER VALUE. FOR EXAMPLE:

A(1), B(3), D(1,3), Z(1,J), X(3**A), J(A(1)) ARE LEGAL SUBSCRIPTED VARI
BUT \( A(1) \), \( Z(1,2,3) \) ARE ILLEGAL SUBSCRIPTED VARIABLES.

A USE OF SUBSCRIPTED VARIABLES FOLLOWS:

2. A NUMERIC LIST, OR VECTOR, OR SINGLE DIMENSIONED ARRAY, IS A SET OF NUMERIC VALUES ARRANGED IN AN ORDERLY MANNER. FOR EXAMPLE, SUPPOSE YOU WOULD LIKE TO READ SOME NUMBERS INTO YOUR PROGRAM AND HAVE THESE NUMBERS AVAILABLE AND IDENTIFIED FOR USE AT A LATER TIME. YOU COULD ASSIGN SIMPLE VARIABLES TO EACH VALUE, BUT WHAT IF YOU HAD 100 NUMBERS? IT IS EASIER TO THINK OF THE NUMBERS AS A LIST OF VALUES OR A VECTOR THE SIZE OF WHICH IS DETERMINED BY HOW MANY NUMBERS YOU HAVE. YOU THEN SIMPLY ASSIGN THE VALUES TO THE LIST.

THE FOLLOWING PROGRAM WILL ASSIGN 10 VALUES TO THE LIST 'A' WHICH CONTAINS 10 ELEMENTS:

```
DIM A(10)
FOR L=1 TO 10
READ A(L)
NEXT L
DATA 1,3,7,5,6,9,4,2,14,4
END
```

THE LIST A, CONTAINING 10 ELEMENTS, IS REPRESENTED BY THE SUBSCRIPTED VARIABLE A(L). READ A(L) IS IN A FOR/NEXT LOOP, AND WITHIN THE LOOP THE 10 ELEMENTS A(1)--A(10) ARE ASSIGNED VALUES. NOW THE LIST A HOLDS THE 10 VALUES AND EACH VALUE IS IDENTIFIABLE.

CONSIDER THE FOLLOWING PROBLEM WHICH SEARCHES A LIST TO FIND THE LARGEST

```
DIM N(10)
FOR I=1 TO 10
LET N(I)=0
NEXT I
READ K
FOR J=1 TO K
READ N(J)
NEXT J
LET L=N(1)
FOR J=2 TO K
IF N(J) LT L THEN 10
LET L=N(J)
```

101
'10 NEXT J
PRINT 'LARGEST NUMBER=',L
DATA 5,3,9,10,6
END

REPLY WITH VALUE OF L

10

THE FIRST FOR/NEXT LOOP ZEROS OUT THE LIST.
IT IS A GOOD PRACTICE TO PUT SOME VALUES IN THE LISTS YOU USE
OTHERWISE THE COMPUTER MAY ASSIGN RANDOM VALUES. BY PUTTING
ZERO IN EACH ELEMENT OF THE LIST YOU ARE ASSURED THAT THE LIST
IS CLEANED UP BEFORE YOU USE IT.

30

3. IN ADDITION TO LISTS(VECTORS), BASIC ALLOWS
YOU THE ABILITY TO USE TABLES, OR MATRICES, OR TWO DIMENSIONAL
ARRAYS. THE SUBSCRIPTS OF A TABLE REPRESENT THE ROWS AND
COLUMNS OF THE TABLE. THE FIRST SUBSCRIPT IS THE ROW AND THE
SECOND SUBSCRIPT IS THE COLUMN. FOR EXAMPLE IN TABLE'D':
D(3,4) REFERS TO THE VALUE OF THE ELEMENT IN ROW 3, COLUMN 4
OF TABLE'D'.

50

AS WITH A LIST, ANY ELEMENT OF A TABLE MAY BE
REFERENCED BY DEFINING THE PAIR OF SUBSCRIPTS AS DESIRED IN
ROW-COLUMN ORDER. IN A 3X3 TABLE(MATRIX) THE TABLE IS REFERENCED
AS FOLLOWS:

1,1 1,2 1,3
2,1 2,2 2,3
3,1 3,2 3,3

WHERE THE FIRST SUBSCRIPT IDENTIFIES THE ROW AND THE SECOND THE COLUMN.
USING THIS REFERENCE SYSTEM CONSIDER HOW TO FILL THE FOLLOWING TABLE

1 2 3
4 5 6
7 8 9

80

THIS TABLE'X' COULD BE FILLED AS FOLLOWS:

DIMX(3,3)
FOR I=1 TO 3
FOR J=1 TO 3
READ X(I,J)
NEXT J
NEXT I
DATA 1,2,3,4,5,6,7,8,9

102
END

REPLY WITH VALUE OF \( X(2,3) \)

6

NOW WITH TABLE 'X' ASSIGNED VALUES CONSIDER THIS PROGRAM (ASSUMING TABLE 'X' HAS BEEN FILLED BY THE ABOVE PROGRAM)

\[
\begin{align*}
\text{LET } S &= 0 \\
\text{FOR } I &= 1 \text{ TO } 3 \\
\text{FOR } J &= 1 \text{ TO } 3 \\
\text{IF } I \neq J \text{ THEN } S \\
\text{LET } S &= S + X(I,J) \\
S \text{ NEXT } J \\
\text{NEXT } I \\
\text{PRINT ('SUM=', S)} \\
\text{END}
\end{align*}
\]

REPLY WITH VALUE OF \( S \)

15

YOU HAVE SEEN HOW SUBSCRIPTED VARIABLES ARE USED TO SET UP LISTS (VECTORS) AND TABLES (MATRICES). HOWEVER TO USE LISTS AND TABLES IN A PROGRAM, YOU MUST DIMENSION THE MAXIMUM SIZE OF YOUR LIST OR TABLE SO THAT ENOUGH SPACE WILL BE ALLOCATED IN THE COMPUTER MEMORY. THIS DIMENSIONING IS DONE WITH THE DIM STATEMENT.

C. DIM STATEMENT

THE DIM STATEMENT TELLS THE COMPUTER THE MAXIMUM SIZE OF VECTORS AND TABLES THAT WILL BE USED IN YOUR PROGRAM. THE DIM STATEMENT MUST APPEAR IN THE PROGRAM BEFORE ANY REFERENCE IS MADE TO THE LIST OR TABLE. IN GENERAL PRACTICE THE DIM STATEMENT IS USUALLY THE FIRST STATEMENT IN THE PROGRAM. THE FORM OF THE DIM STATEMENT IS:

\[
\text{DIM } \langle \text{LIST VARIABLE} \rangle ( \langle \text{SIZE} \rangle )
\]

\[
\text{DIM } \langle \text{TABLE VARIABLE} \rangle ( \langle \text{SIZE}, \text{SIZE} \rangle )
\]

103
THE LIST AND TABLE VARIABLES ARE SUBSCRIPTED VARIABLES AS DEFINED EARLIER. THE SIZE IS AN UNSIGNED INTEGER IN PARENTHESES WHICH DENOTES THE MAXIMUM SIZE OF THE LIST OR TABLE.

THE DIM STATEMENT MAY CONTAIN A NUMBER OF LISTS OR TABLES WITH THEIR SIZES SEPARATED BY COMMAS, FOR EXAMPLE:

```
DIM A(6), X(10, 3), Z(14, 21)
```

IF YOU TRY TO REFERENCED AN ELEMENT IN A LIST OR TABLE BEYOND THE MAXIMUM SIZE IN THE DIM STATEMENT YOU WILL GET AN ERROR. THE MAXIMUM SIZE LIST (VECTOR) OR TABLE (MATRIX) ALLOWED BY THE CAT-BASIC COMPILER IN ANY ONE PROGRAM IS A TOTAL OF 1600 COMPUTER MEMORY SPACES.

YOU WILL NOW BE ASKED SOME QUESTIONS:

1.) ARE THE FOLLOWING DIM STATEMENTS CORRECT, REPLY: YES OR NO

```
DIM A(10)
```

YES

THE STATEMENT IS INCORRECT. SUBSCRIPTED VARIABLES ARE A SINGLE LETTER FOLLOWED BY ONE OR TWO SUBSCRIPTS IN PARENTHESES.

```
DIM X(500), D(1, 10), C(5, 5)
```

NO

1. SUMMARY

IN THIS LESSON YOU HAVE LEARNED HOW TO USE THE FOR/NEXT STATEMENT AND HOW TO MANIPULATE LISTS (VECTORS) AND TABLES (MATRICES) BY USING SUBSCRIPTED VARIABLES AND THE DIM STATEMENT. YOU NOW HAVE ALL THE TOOLS TO BEGIN WRITING SOFISTICATIONED PROGRAMS AND AS YOU WRITE MORE COMPLICATED PROGRAMS, YOU WILL FIND A NEED FOR SUBROUTINES. SUBROUTINES ARE COMMONLY USED PROGRAM SEGMENTS THAT ARE USED OVER AGAIN IN OTHER PARTS OF YOUR PROGRAM. SUBROUTINES ALLOW YOU TO BRANCH TO THE COMMONLY USED SEGMENT AND THEN RETURN TO WHERE YOU WERE AND CONTINUE EXECUTING. THE GOSUB STATEMENT ALLOWS SUBROUTINES IN BASIC, AND YOU WILL BE
INTRODUCED TO IT IN THE NEXT LESSON.

YOU WILL NOW BE GIVEN TWO OPTIONAL PROGRAMMING PROBLEMS TO EXERCISE YOUR NEW TOOLS.

1.) WRITE A PROGRAM TO REVERSE THE NUMBERS IN A 10-ELEMENT LIST 'X'
IN OTHER WORDS INTERCHANGE X(1) WITH X(10), X(2) WITH X(9), ETC.
READ IN TEN VALUES AND TEST YOUR PROGRAM BY PRINTING THE LIST
BEFORE AND AFTER.

2.) WRITE A PROGRAM TO ARRANGE THE FOLLOWING LIST IN
DECENDING ORDER: 10,30,5,15,40. ONE METHOD OF APPROACHING THIS
IS TO CHECK THE FIRST ELEMENT OF THE LIST AGAINST THE SECOND.
IF THE FIRST IS NOT LARGER THEN EXCHANGE THE TWO, OTHERWISE
GO AND COMPARE THE NEXT TWO IN THE LIST. THIS PROCESS IS REPEATED
UNTIL THERE ARE NO MORE EXCHANGES TO BE MADE. A COUNT OF
THE EXCHANGES CAN BE MADE, AND WHEN THE COUNT EQUALS 0 THE LIST IS IN ORDER.

Go

IF YOU WANT TO EXECUTE THESE PROBLEMS REPLY: YES, OTHERWISE NO.

If you would like to see a solution to the problems REPLY: YES, OTHERWISE NO.

Do you want to terminate your instruction session?
(REPLY: YES OR NO)

If you want to execute programs at this time
THEN REPLY: YES; OTHERWISE REPLY: NO AND THE INSTRUCTION WILL CONTINUE.

Reply with the number of the lesson you wish to cover.

105
EXECUTION BEGINS...

*** LESSON 6 ***

THIS LESSON WILL INTRODUCE YOU TO SUBROUTINES AND THE PROGRAMMING TECHNIQUE CALLED 'RECURSION'.

A. SUBROUTINES

IN MANY PROGRAMS A BLOCK OF STATEMENTS MAY BE NEEDED ON MORE THAN ONE OCCASION. SINCE THE REPETITION OF A BLOCK OF STATEMENTS IS BURDENSOME, A TECHNIQUE TO ECONOMIZE ON CODING INSTRUCTIONS CALLED A SUBROUTINE, IS PRESENTED. THE FORM OF THE SUBROUTINE IS:

GOSUB << STATEMENT NUMBER >>

***

***

RETURN

EXECUTION F THE GOSUB STATEMENT CAUSES THE COMPUTER TO TRANSFER CONTROL TO THE STATEMENT NUMBER AFTER 'GOSUB'. WHEN CONTROL IS TRANSFERRED, STATEMENTS ARE EXECUTED SEQUENTIALLY UNTIL A 'RETURN' IS ENCOUNTERED. AT THAT TIME, CONTROL IS RETURNED TO THE NEXT STATEMENT FOLLOWING THE 'GOSUB' STATEMENT WHICH 'CALLED' THE SUBROUTINE. FOR EXAMPLE CONSIDER THE PROGRAM SEGMENT:

```
50 PRINT A,B,C,D,E
GOSUB 500
FOR I=1 TO E
***
***
500 REM SUBROUTINE
LET X=A*B
LET Z=B*C
LET R=SQR(D)
RETURN
***
```

106
THE SUBROUTINE CONSISTS OF A SEQUENCE OF BASIC STATEMENTS ENDING WITH A 'RETURN' STATEMENT. THE SUBROUTINE MAY ONLY BE ENTERED FROM A 'GOSUB' STATEMENT, AND WILL ONLY RETURN TO ITS PROPER PLACE AFTER ENCOUNTERING A 'RETURN' STATEMENT. ALL THE VARIABLES OF THE MAIN PROGRAM ARE AVAILABLE (PASSED) TO THE SUBROUTINE, AND VICE VERSA. THE VARIABLES IN THE SUBROUTINE MUST BE CHOSEN CAREFULLY SO AS NOT TO ACCIDENTALLY CONFLICT OR ALTER VARIABLES IN THE MAIN PROGRAM.

THE FOLLOWING PROGRAM TO COMPUTE THE GREATEST COMMON DENOMINATOR (GCD) OF THREE NUMBERS BY EUCLID'S ALGORITHM WILL DEMONSTRATE A USE OF SUBROUTINES:

```
REM FIND GCD OF A, B, C
PRINT 'A', ',B', ',C', ',GCD'
20 READ A, B, C
IF C=9999 THEN 9999
LET X=A
LET Y=B
REM G=GCD OF A, B
GOSUB 200
LET X=G
LET Y=C
REM G=GCD OF G, C
GOSUB 200
PRINT A, B, C, G
GO TO 20
200 LET Q=INT(X/Y)
LET R=X-Q*Y
IF R=0 THEN 500
LET X=R
LET Y=Y
GO TO 200
500 LET G=R
RETURN
DATA 60, 90, 120
DATA 38456, 64872, 98765
DATA 0, 0, 9999
9999 END
```

OUTPUT PRODUCED:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>GCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>90</td>
<td>120</td>
<td>30</td>
</tr>
<tr>
<td>38456</td>
<td>64872</td>
<td>98765</td>
<td>1</td>
</tr>
</tbody>
</table>

80
GOSUB STATEMENTS MAY BE NESTED TO LOGICALLY CALL ANOTHER GOSUB, SIMILAR TO FOR/NEXT NESTED LOOPS. YOU MUST ONCE MORE BE CAREFUL ABOUT THE STATUS OF VARIABLES BECAUSE VARIABLES WILL BE PASSED FROM ONE SUBROUTINE TO ANOTHER.

A SUBROUTINE THAT IS NESTED SO THAT IT CALLS ITSELF IS CALLED RECURSION OF SUBROUTINES OR RECURSION.

4. RECURSION

RECURSION IS A PROGRAMMING TECHNIQUE IN WHICH SUBROUTINES CAN CALL THEMSELVES. FOR EXAMPLE CONSIDER THIS PROGRAM WHICH FINDS THE FACTORIAL OF A NUMBER USING ITERATIVE METHODS AND THEN USING RECURSION:

```plaintext
REM FACTORIAL: F(N) = 1*2*3*...*(N-1)*N
PRINT 'ITERATIVE SOLUTION'
PRINT N, F(N)
10 READ N
IF N GT 0 THEN 20
GO TO 50
20 PRINT N,
GOSUB 100
PRINT F
GO TO 10
100 REM ITERATIVE SOLUTION
LET F = 1
FOR I = 1 TO N
LET F = F*I
NEXT I
REM RETURN F = FACTORIAL(N)
RETURN
50 REM FACTORIAL: F(N) = IF N=0 THEN F(N)=1 REM OTHERWISE, F(N) = N*F(N-1)
RESTORE
PRINT 'RECURSIVE SOLUTION'
60 READ N
IF N GT 0 THEN 70
GO TO 9999
70 PRINT N,
GO SUB 200
PRINT F
GO TO 60
200 REM RECURSIVE SOLUTION
IF N GT 0 THEN 210
LET F = 1
RETURN
210 LET N = N-1
REM RECURSIVE CALL
```

108
IN THE ITERATIVE METHOD THE FACTORIAL FUNCTION, F(N), WAS MULTIPLIED BY ITSELF IN A LOOP FROM I=1->N. IN THE RECURSIVE SOLUTION THE FACTORIAL FUNCTION, F(N), IS DEFINED IN TERMS OF ITS FINAL VALUE. WHEN N=0, F(N)=1; OTHERWISE F(N)=N*(FACTORIAL OF N-1).

TO KEEP TRACK OF WHERE THE SUBROUTINE CALLS RETURN IN RECURSION IT HELPS TO VISIONING A LAST-IN-FIRST-OUT (LIFO) STACK CONTAINING THE ADDRESS OF THE GOSUB STATEMENTS. EVER TIME A GOSUB IS ENCOUNTERED, PUT ITS ADDRESS ON TOP OF THE STACK (PUSHING DOWN ANYTHING PREVIOUSLY ON THE STACK). THEN EVERY TIME A RETURN IS ENCOUNTERED, RETURN TO THE TOP ADDRESS ON THE STACK (AND POP UP THE NEXT ADDRESS ON THE STACK).

TO FULLY UNDERSTAND THE CONCEPT OF RECURSION YOU SHOULD STEP THROUGH THE FACTORIAL PROBLEM BY HAND USING THE HELP OF THE STACK TO SEE HOW RECURSION WORKS. IF YOU UNDERSTAND THE CONCEPT OF RECURSION YOU ARE READY TO SOLVE THIS PROBLEM:

LET X=1
GOSUB 100
PRINT X
GO TO 9999
100 LET X=X*1
   IF X GT 3 THEN 150
   GOSUB 100
   RETURN
9999 END

REPLY WITH VALUE OF X

5
YOUR ANSWER IS INCORRECT. CONSIDER THE FOLLOWING
TABLE OF VALUES FOR X, AND ITEMS IN THE STACK (1-FIRST GOSUB, 2-SECOND GOSUB)

<table>
<thead>
<tr>
<th>X</th>
<th>STACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

YOU SHOULD STEP THROUGH THIS EXAMPLE UNTIL YOU UNDERSTAND THE RECURSION TECHNIQUE. HOWEVER, KNOWING HOW TO USE RECURSION IS NOT A REQUIREMENT FOR KNOWING HOW TO USE BASIC; IT IS ONLY A CLASSIC PROGRAMMING TECHNIQUE.

C. SUMMARY

SUBROUTINES AND RECURSION ARE A VALUABLE ADDITION TO YOUR REPERTOIRE OF BASIC STATEMENTS. THEY SAVE BOTH PROGRAMMER TIME AND COMPUTER STORAGE SPACE. THE GOSUB/RETURN COMMAND AND THE OTHER TWELVE BASIC STATEMENTS THAT YOU HAVE ALREADY LEARNED AND USED FORM THE BASIC LANGUAGE.

THE FACILITY THAT YOU GAIN IN PROGRAMMING BY USING THE BASIC LANGUAGE WILL DEPEND UPON HOW OFTEN YOU EXERCISE YOUR SKILLS. THE LAST LESSON WILL PROVIDE YOU WITH A BRIEF SUMMARY OF THE BASIC LANGUAGE.

THE FOLLOWING PROBLEMS WILL TEST YOUR LATEST SKILLS:

1. COMPUTE THE NET SALARY, RETIREMENT CONTRIBUTION AND TAX FOR N EMPLOYEES. NET PAY = GROSS SALARY - RETIREMENT - TAX

USE SUBROUTINES TO CALCULATE:

A) RETIREMENT CONTRIBUTION
   IF SALARY <$200, R=0.0
   IF SALARY <$2500, R=$100
   IF SALARY >$2500, R=$200

B) TAX
   IF SALARY <$800, T=0.5
   OTHERWISE
   T=0.03(SALARY)

THE DATA IS:

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>GROSS SALARY</th>
</tr>
</thead>
</table>

110
2. Write a recursive program to sum the numbers from X to Y. Read in your own data and test your program.

Then

If you want to execute these programs now, then
Reply: yes, otherwise no

No

Do you want to terminate your instruction session?
(Reply: yes or no)

No

If you want to execute programs at this time then reply: yes; otherwise reply: no and the instruction will continue.

No

Reply with the number of the lesson you wish to cover
EXECUTION BEGINS...

*** LESSON 7 ***

THIS LESSON SUMMARIZES THE BASIC DEFINITIONS AND BASIC STATEMENTS THAT YOU HAVE LEARNED IN CAI-BASIC.

A. BASIC DEFINITIONS

1. ALPHA-NUMERIC CHARACTERS:
   A) DIGITS : 0—>9
   B) LETTERS : A—>Z
   C) SPECIAL CHARACTERS : ** / \ - ( ) + , , $

2. STATEMENT NUMBERS : ONE TO FOUR DIGITS (0—>9999)

3. STRING : ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS ENCLOSED IN 'SINGLE' QUOTES. EG. 'HELP'

4. NUMBERS : (LIMITED TO 9 DIGITS)
   A) INTEGERS : DIGITS WITH NO FRACTIONAL PART. EG. 5,7,10
   B) REAL : DIGIT WITH A FRACTIONAL PART. EG. 5.0,7.3,16.0
   A NUMBER MAY BE PRECEEEDED BY A SIGN (+, -), BUT IS ASSUMED TO BE POSITIVE IF NONE IS GIVEN.

5. VARIABLES:
   A) SIMPLE VARIABLES : A SINGLE LETTER, OR A SINGLE LETTER FOLLOWED BY A DIGIT. EG. A, A1, B6, Z0
   B) ALPHA VARIABLES : A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN ($) EG. AS, VS
   C) SUBSCRIPTED VARIABLE : (SINGLE LETTER)
      (1) SINGLE SUBSCRIPT : <<LETTER>> ( <<EXPRESSION>> )
      (2) DOUBLE SUBSCRIPT : <<LETTER>> ( <<EXPRESSION,EXPRESSION>> )

EG
A(5),Z(5,10),X(A*B,X**2)

5. OPERATORS: (LISTED IN DECENDING HIERARCHY)
   A) EXPONENTIATION **
   B) MULTIPLICATION *
   C) DIVISION /
   D) ADDITION +
   E) SUBTRACTION -

6. EXPRESSIONS:
   A) SINGLE NUMBER OR VARIABLE EG. 10,-.53,D,X(1,J)
   B) BUILT IN FUNCTION EG. SQR(10),TAN(.75)
   C) ARITHMETIC EXPRESSION: EXPRESSIONS SEPARATED BY OPERATORS AND GROUPED BY PARENTHESIS. EG. 5+6.0, A**2(3*X-.6)

7. RELATIONS
   A) GT  A GT B  A GREATER THAN B
   B) GE  A GE B  A GREATER THAN OR EQUAL TO B
   C) LT  A LT B  A LESS THAN B
   D) LE  A LE B  A LESS THAN OR EQUAL TO B
   E) NE  A NE B  A NOT EQUAL TO B
   F) =  A = B  A EQUAL TO B

8. BASIC STATEMENTS
   1. REM << ANY SET OF ALPHA-NUMERIC COMMENTS >>
   2. READ << VARIABLE,...,VARIABLE >>
   3. END

4. LET << VARIABLE >> = << EXPRESSION >>

5. PRINT << 'STRING' OR EXPRESSION,...,'STRING' OR EXPRESSION >>
   OR SIMPLY PRINT

6. DATA << 'STRING' OR NUMBER,...,'STRING' OR NUMBER >>
7. RESTORE OR RESTORE$
8. IF << EXPRESSION >> << RELATION >> THEN << STATEMENT NUMBER >>
9. GO TO << STATEMENT NUMBER >>
10. ON << EXPRESSION >> GO TO << STATEMENT NUMBER, ..., STATEMENT NUMBER >>
11. FOR << SIMPLE VARIABLE >> = << EXPRESSION >>
   TO << EXPRESSION >> STEP << EXPRESSION >>
   ...\n   ...\n   NEXT << SIMPLE VARIABLE >>
12. DIM << LETTER >> ( << INTEGER EXPRESSION >> ) OR
    DIM << LETTER >> ( << INTEGER EXPRESSION, INTEGER EXPRESSION >> )
13. GOSUB << STATEMENT NUMBER >>
   ...\n   ...
    RETURN

go

THAT CONCLUDES YOUR INSTRUCTION WITH CAI-BASIC .
HOWEVER, YOU ARE INVITED TO USE WHATEVER FACILITIES OF CAI-BASIC
YOU DESIRE AT ANY TIME . IF YOU HAVE NOT RUN PROGRAMS UNDER
THE OS/BATCH MODE (PUNCHING YOUR OWN CARDS AND HANDING THEM
ACROSS THE COUNTER TO BE RUN), YOU SHOULD GET THE BASIC MANUAL,
TECHNICAL NOTE NR. 0211-12 , IN ROOM I-147 TO FIND THE PROPER
JOB CONTROL CARDS REQUIRED . GOOD LUCK WITH THE COMPUTER , AND
REMEMBER IT ONLY DOES WHAT YOU TELL IT TO DO - GIGO (GARBAGE IN
GARBAGE OUT ).

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO )

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION
WILL CONTINUE .

yes

**** CAIBASIC COMPILER ****

THE CAIBASIC COMPILER IS A LINE BY LINE INTERPRETER . THE BASIC
The interpreter accepts standard Basic statements, and it analyzes each Basic statement as it is input. An added feature of the CAIBasic compiler is an editing and a debugging routine that allows the user to add, delete, and correct statements; and to get a listing of alpha-numeric and numeric data used, and a trace of all simple variables as they are assigned values in the program.

The debug feature is used by adding the key word "DEBUG" as a statement to your program.

The editing mode is available to the user when an execution error occurs and after successful execution. In the event of an input error the interpreter will analyze the error and print an error message. If this occurs find the error, and input the correct Basic statement for the current line of input.

(Note: Typing errors can be deleted by typing four dollar signs ($$$) on the same line as the error, hitting carriage return, and inputting the line again.)

**** CAIBasic execution ****

Input Basic program now (one line at a time)

let x=1
gosub 100
print x
go to 9999
100 let x=x+1
   if x>3 then 150
      gosub 100
9999 return
150 return

Do you want to edit your program
Reply: Yes or No.

yes

115
*** CIA BASIC EDIT MODE ***

BY USING THE REFERENCE NUMBERS LISTED TO THE LEFT OF YOUR BASIC PROGRAM STATEMENTS YOU MAY ADD, DELETE, OR CORRECT ONE LINE OF THE PROGRAM AT A TIME. IN ALL EDITING THE FIRST STEP IS TO INPUT THE PROGRAM STATEMENT REFERENCE NUMBER AND HIT THE CARROIAGE RETURN. THE SECOND STEP DEPENDS ON WHAT EDITING YOU DO:

1. DELETE
   THE BASIC STATEMENT REFERENCED IS DELETED BY TYPING THE LETTERS DEL

2. CORRECT
   TO CORRECT THE BASIC STATEMENT REFERENCED TYPE IN THE COMPLETE CORRECT BASIC STATEMENT.

3. ADD
   A BASIC STATEMENT IS ADDED 'AFTER' THE BASIC STATEMENT REFERENCED BY TYPING THE LETTERS ADD1 FOLLOWED BY THE BASIC STATEMENT. ALL BLANKS FOLLOWING THE LETTERS ADD1 WILL BE INCLUDED IN THE BASIC STATEMENT. TO PLACE A STATEMENT 'BEFORE' THE FIRST STATEMENT IN THE PROGRAM, USE THE REFERENCE NUMBER 0.

1 LET X=1
2 GOSUB 100
3 PRINT X
4 GOTO 9999
5 100 LET X=X+1
6 IF X GT 3 THEN 150
7 GOSUB 100
8 150 RETURN
9 9999 END

INPUT REFERENCE NUMBER NOW

INPUT EDITING NOW (DEL, BASIC STATEMENT, ADD1, ...)

150 LET X=X+1

MORE CORRECTIONS TO BE MADE ?? REPLY: YES ; OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED.

YES
LET X=1
GOSUB 100
PRINT X
GO TO 9999
100 LET X=X+1
IF X GT 3 THEN 150
GOSUB 100
150 LET X=X+1
9999 END

INPUT REFERENCE NUMBER NOW
8
INPUT EDITING NOW (DEL, BASIC STATEMENT, ADD1....)
add1 return
MORE CORRECTIONS TO BE MADE ?? REPLY: YES ;
OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED .
nore

LET X=1
GOSUB 100
PRINT X
GO TO 9999
100 LET X=X+1
IF X GT 3 THEN 150
GOSUB 100
150 LET X=X+1
RETURN
9999 END

117
DO YOU WANT TO EXECUTE ANOTHER PROGRAM
REPLY: YES OR NO.

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION?
(REPLY: YES OR NO)

yes

R; T=6.99/23.96 15.56.04
BIBLIOGRAPHY


