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></td>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
</tr>
<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.
at the Naval Postgraduate School.

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 the 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 Post-graduate 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 yyyygxx          (xx is the terminal nr.)
(yyy is your user nr.)

ENTER PASSWORD
npq
ENTER 4-DIGIT PROJECT NUMBER.... ETC.
0623cs04
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 pro-
gram 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 state-
ment 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 ALLOAD 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 CAIBASIC PROGRAM:

ACON COMPLR ERR INSNO
APRINT CONV 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
ILNGTH LENGTH OF INPUT IN VECTOR CARDP
ASTRKS WHEN CARDP(1)=ASTRKS THE USER HAS MADE A TYPING ERROR
ALPHA VECTOR HOLDING THE LETTERS OF THE ALPHABET
DIGIT VECTOR HOLDING THE DIGITS **S**
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 INTERPRETS STUDENT RESPONSES. SOME ADDED FEATURES ARE EXECUTION AND EDITING OF "BASIC" PROGRAMS IN "TESTI" SUBROUTINES CALLED FROM CAIBAS ARE:

INITIAL
CRUNCH
TESTI
LESON1 --> LESON7
SEARCH
ALOAD

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(2500), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XDATA(50C), STRING(5),
DIGIT(10), IPRTB(10), LIST(100), INSLST(100),
PR2(2500), NERAS, INST, NSLST, DEBUG, DOLSEN, QUOTE,
EQUALS, PARTR, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLEFT, ASTRSK, BLANK

CCMGN INTERP, EXERR
REAL*8 LES1/LES7, LESO1/LESO7

99 FORMAT(1X, 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 OR* CAI00260
* ANYWHERE ON THAT INPUT LINE AND HIT CARRIAGE RETURN, THE* CAI00270
* ENTIRE LINE WILL THEN BE IGNORED AND YOU CAN TYPE IN THE CORRECT* CAI00280
* INPUT OR RESPONSE* // 5X,
* 3* IF AT ANY POINT IN THE SESSION YOU WANT TO STOP THE SESSION* CAI00300
* TYPE IN THE WORD "QUIT" AS SOON AS YOU ARE ASKED FOR THE NEXT* CAI00310
* RESPONSE,HIT CARRIAGE RETURN, THEN HIT ATTN KEY AND TYPE LOGOUT
* // 5X,*
* 4* DURING YOUR TERMINAL SESSION CAI-BASIC WILL HALT OCCASIONALLY* CAI00330
* TO LET YOU READ A SEQUENCE OF INFORMATION. WHEN YOU ARE* CAI00340
* READY TO CONTINUE TYPE IN "GO", AND HIT CARRIAGE RETURN// 5X
* 5* DURING YOUR TERMINAL SESSION YOU MAY NOTICE THAT
* THE TYPING IS NOT ALWAYS PERFECT. SOME DAYS THE COMPUTER IS* CAI00370
* NOT UP TO PAR AND YOU WILL HAVE TO ADJUST TO THE MINOR IRITANT//
* 6* IF YOU ONLY WANT TO EXECUTE PROGRAMS AT THIS TIME THEN
* 7* "REPLY : YES : OTHERWISE REPLY : NO *" */

1 READ(5,101,END=300)CARD
CALL CRUNCH(LINPS)
IF(CARDP(1).EQ.AST(5K)) GO TO 1
IF(CARDP(1).NE.ALPHA(25)) GO TO 2

EXECUTION ONLY PHASE

CALL TEST1
WRITE(6,109)
WRITE(6,104)
GO TO 400
2 IF(CARDP(1).EQ.ALPHA(14)) GO TO 3
WRITE(6,104)
GO TO 400
3 WRITE(6,100)
FORMAT(101) IF THIS IS YOUR FIRST SESSION WITH CAIBASIC, THEN* CAI00490
* "REPLY : YES : OTHERWISE REPLY : NO */
4 READS(101,END=3021CARD
101 FORMAT(83A1) CAI00530
CALL CRUNCH(LINPS)
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) CAI00590
102 FORMAT(6,102) NOW YOU MAY CHOOSE TO REVIEW ANY LESSONS*/ CAI0G650
*OR TO BEGIN WHERE YOU ENDED DURING YOUR LAST SESSION.*

WRITE(6,105) CAI06660

105 FORMAT('10X,'THE LESSONS ARE AS FOLLOWS: /*10X,'LESSON 1 CAI06860
   *PROGRAM FORMAT AND BASIC DEFINITIONS
   * '/10X,'LESSON 2 REMARKS, INPUT/OUTPUT
   * AND DATA'/10X,'LESSON 3 ASSIGNMENT STATEMENTS AND BUILT IN FUNCTION
   *IONS'/10X,'LESSON 4 BRANCHING '/10X,'LESSON 5 LOOPING AND SUBSCR
   *PTED VARIABLES'/10X,'LESSON 6 SUBROUTINES AND RECURSION '/10X,'
   *LESSON 7 SUMMARY OF BASIC STATEMENTS
   * '/*** WHEN YOU ARE READY TO CONTINUE, TYPE IN GO AND HIT */ CAI00730
   *THE CARRIAGE RETURN */ **//)
READ(9,101,END=5)CARD
CAI00750
5 WRITE(6,103) CAI00760

103 FORMAT('10X, 5X,'REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO C
   *OVER*/1
   6 READ(5,1C1,END=3C1)CARD
   CALL CRUNC(:ILNGTH)
   DO 7 I=2,8
   IF (CARDP(I).EQ.DIGIT(I)) GO TO 9
   7 CONTINUE
301 WRITE(6,104)

104 FORMAT('10X,*** YOUR REPLY WAS TYPED INCORRECTLY; CHECK THE */ CAI00850
   *QUESTION AND REPLY AGAIN ***//)
   GO TO 6
9 LSNUM=I-1
10 GO TO (11,12,13,14,15,16,17),LSNUM
11 CALL LESSCN
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 25

INSTRUCTION PHASE ** NEW USER

20 IF(CARDP(1).EQ.ALPHA(25))GO TO 21
302 WRITE(6,104)

302 FORMAT('10X,CAI BASIC IS A PROGRAM TO TEACH YOU THE */ CAI01110
   */
*FUNDAMENTALS OF A PROGRAMMING LANGUAGE. THE LANGUAGE TO BE LEARNED: CAI01120
**/ THE befindic Language: A SIMPLE LANGUAGE FOR THE USER WHO HAS LITTLE KNOCADAI01130
**/ LEARNING COMPUTERS AND WHOSE PRIMARY INTEREST IS IN OBTAINING */
**/ RESULTS. */IX, THE SIMPLICITY OF THE BASIC LANGUAGE AND ITS RANGE CAI01150
**/ OF CAPABILITIES SHOULD ALLOW YOU TO LEARN THE LANGUAGE AND */ CAI01160
**/ WRITE PROGRAMS IN A MINIMAL AMOUNT OF TIME. */IX, THE REFERENCE T
**/ EXTENDED RECOMMENDATIONS FOR CARBASIC ARE */IX X, 1. BASIC LANGUAGE MANUA
**/ L, TN # C211-12 APRIL 1971 */ ( FREE UPON REQUEST IN 1-147 ) */IX
**/ 2. INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE R.L.
**/ NOLAN */ ( SOOHSTORE / MAIN LIBRARY ) */IX X, 3. BASIC PROGRAMMING
**/ V.E. HARE */ ( COMPUTER CENTER LIBRARY ) */**** AFTER YOU HAVECAI01170
**/ E FINISHED READING AN INPUT, TYPE IN GO AND HIT THE RETURN KEY CAI01180
**/ AND THE PROGRAM WILL CONTINUE */
**/
307 READ(5,101)END=3G7)CARD
307 WRITE(6,107)
107 FORMAT(*6.10X,'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. */
**/
308 WRITE(6,108)
108 FORMAT(*6.10X,'IN EACH LESSON YOU WILL BE GIVEN INSTRUCTION */ CAI01290
**/ SEQUENCES AND THEN YOU WILL BE ASKED QUESTIONS TO SEE IF YOU */ CAI01300
**/ 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 */IX X, IF YOU KEEP THE TELETAI01340
**/ TYPE OUTPUT FROM YOUR TERMINAL SESSION */ YOU WILL HAVE A READY REFERCAI01350
**/ ENCE FOR FUTURE USE */
**/
309 LSNUM=1
24 IF (LSNUM.GT.7) GO TO 40
GO TO (11,12,13,14,15,16,17),LSNUM
25 WRITE(6,109)
109 FORMAT(*6.10X,'DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION CAI01420
**/ ? '') (REPLY : YES OR NO) */
**/
26 READ(5,111)END=36G1)CARD
CALL CRUNCH(LENGTH)
IF(CRUMP(1).EQ.ALAHA(251) GO TO 40
IF (CRUMP(1).EQ.ALAHA(14)) GO TO 30
310 WRITE(6,104)
GO TO 26
104 C
26 OLD USERS PICK NEXT LESSON ; NEW USERS GET NEXT LESSON IN SEQUENC
C
30 WRITE(6,109)
209 FORMAT(*6.10X,'IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME: // CAI01520
**/ THEN REPLY : YES OTHERWISE REPLY : NO AND THE INSTRUCTION */ C
**/ WILL CONTINUE */
32 READ(5,101,END=33)CARD
CALL CRUNCH(ILNGTH)
IFICARDP(1),EQ.,ALPHA(14)) GO TO 34
IFICARDP(1),NE.,ALPHA(25)) GO TO 33
ENTER EXECUTION PHASE
CALL TEST1
GO TO 25
33 WRITE(6,104)
GO TO 22
34 IF(OLD.EQ.1) GO TO 5
WRITE(6,110)
110 FORMAT('01,IGX,* DO YOU WANT TO REVIEW A LESSON BEFORE GOING ON?*(CAI0150
# REPLY: YES OR NO''/')
27 READ(5,101,END=311)CARD
CALL CRUNCH(ILNGTH)
IFICARDP(1),EQ.,ALPHA(25)) GO TO 5
IFICARDP(1),NE.,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
END
IEXERR
NERKS
INTERP
NSTMT
SFILE
NREF

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
INIG   **
NFOR   **

COMMON
- STACK(100), PROG(200), CAI(80), CARD(80), ALPH(0),
- JAPRT, INPR, INDATA(500), Xndata(500), STRING(5),
- DIGIT(10), PRIB(10), LISTST(100), STRLIST(100),
- ERR(250), ERRX, INST, NSLIST, DEBUG, DOLG, QUTE,
- EQUALS, PARRT, DECIMAL, PLUS, MINUS, SLASH, COMMA,
- PARLFT, ASTROK, BLANK

EXECUTION PHASE
DIMENSION ServiceProvider(100,100)

IF NEW SET DON'T PRINT INTRO AGAIN
IF[new.eq.1]GO TO 3

WRITE{16,99}
99 FORMAT(*15x,*** CAIBASIC COMPILER ***
*THE CAIBASIC COMPILER IS A LINE BY LINE INTERPRETER 'THE BASIC'
*INTERPRETER ACCEPTS STANDARD BASIC STATEMENTS AND IT ANALYZES!
*THE CAIBASIC COMPILER IS IN A TESTING AND A DEBUGGING ROUTINE
*THAT ALLOWS THE USER TO ADD, DELETE AND CORRECT STATEMENTS;
*AN TO GET A LISTING OF ALPHABETIC AND NUMERIC DATA USED;
*AND A TRACE OF ALL SIMPLE VARIABLES AS THEY ARE ASSIGNED VALUES
*THE BUILD 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 */

3 WRITE{16,30C}
300 FORMAT(*10x,*** CAIBASIC EXECUTION ***/11)}
CALL INITIAL
NEW=1
DEBUG=C=0
NSTMT=0
IEND=0
INBIG=1
IARRAY=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)
101 FORMAT(8CA1)
CALL CRUNCH(ILNGTH)
IF(CARDP(I).EQ.ASTSK) GO TO 10

FIND ' END ' STATEMENT
DO 699 I=1,80
IF(CARDP(I).NE.ALPHA(5).OR.CARDP(I+1).NE.ALPHA(14)) GO TO 699
IF(CARDP(I+2).EQ.ALPHA(4).AND.CARDP(I+3).EQ.BLANK) GO TO 701
699 CONTINUE
CALL CCPLR(NFOR,IARRAY,ILNGTH,INBIG)

FIND NO ERRORS , FILE INPUT STATEMENT
IF(INERRS.EQ.0) GO TO 9
IF(IEXERR.EQ.1) GO TO 9
NERRS=0
GO TO 10

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

EXECUTION ERROR GO TO EDIT ROUTINE

12 IF (IEEXERR.EQ.1) GO TO 31
513 WRITE (6, 205)
205 FORMAT ('** Do you want to edit your program */*' REPLY : Y* SP OR NO */')
312 READ (5, 131, END=315) CARD
CALL CRUNCH (ILNTH)
IF (CARDP(1).EQ.ALPHA (25)) GO TO 33
ELSE IF (CARDP(1).EQ.ALPHA (14)) GO TO 313
315 WRITE (6, 106)
313 WRITE (6, 105)
105 FORMAT ('** If you desire to have a smooth copy of your */' REPLY : YES SP OTHERWISE */')
* PROGRAM WITH ITS EXECUTION THEN REPLY: YES ; OTHERWISE * /
** REPLY : NO */'
20 READ (5, 101, END=326) CARD
CALL CRUNCH (ILNTH)
IF (CARDP(1).NE.ALPHA (25)) GO TO 25
PRINT SMOOTH COPY OF PROGRAM AND EXECUTE IT

314 WRITE (6, 104)
104 FORMAT (**----------------------------------------------')
DO 13 I = 1, NSTMT
13 WRITE (6, 103) (FILENAME(I,J), J=1, 80)
103 FORMAT (**----5X, 8G1)"
135 CALL INITIAL
DEBUG=0*0
136 INBIG=1
IARRAY=808
NFOR=0
INTERP=0
IEEXERR=0
DO 15 K = 1, NSTMT
DO 14 L = 1, 80
14 CARD(L)=FILENAME(K,L)
CALL CRUNCH (ILNTH)
15 CALL COMPLR (INFOR, IARRAY, ILNTH, INBIG)

EXECUTION ERROR GO TO EDIT ROUTINE

16 IF (IEEXERR.EQ.1) GO TO 31
WRITE (6, 104)
GO TO 26
25 IF(CARDP(11).EQ.ALPHA(14)) GO TO 26
326 WRITE(6,106)
106 FORMAT(0,.*** YOUR REPLY IS INCORRECTLY TYPED REPLY AGAIN***//
* 1) TESO1660
* 2) TESO1670
* 3) TESO1680
GO TO 20

26 WRITE(6,107)
107 FORMAT(0,.*** DO YOU WANT TO EXECUTE ANOTHER PROGRAM***//
*: YES OR NO.*//*
29 READ(5,101,END=330) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(11).EQ.ALPHA(14)) GO TO 30
IF(CARDP(11).EQ.ALPHA(25)) GO TO 3
330 WRITE(6,106)
GO TO 29
30 RETURN

EDIT ROUTINE.

31 WRITE(6,108)
108 FORMAT(0,.**** EXECUTION ERROR ****
*IF YOU WANT TO CORRECT YOUR PROGRAM NOW THEN REPLY : YES AND *
*YOU WILL ENTER THE EDIT MODE TO CORRECT YOUR ERRORS *
*OTHERWISE REPLY : NO **//
208 READ(5,101,END=343) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(11).EQ.ALPHA(251)) GO TO 43
33 IF(OLD.EQ.-1) GO TO 321
WRITE(6,115)
110 FORMAT(0,.**** 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
* 2. CORRECT
* THE BASIC STATEMENT REFERENCED IS DELETED BY TYPING THE
* LETTERS OR
* TO CORRECT THE BASIC STATEMENT REFERENCED TYPE IN THE COMPLETE***
*CORRECT BASIC STATEMENT **//
* 3. ADD
* A BASIC STATEMENT IS ADDED **AFTER** THE BASIC STATEMENT**//
*REFERENCES 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 G**//
*OLD=1**//
OLD=1

TESO1660
TESO1670
TESO1680
TESO1690
TESO1700
TESO1710
TESO1720
TESO1730
TESO1740
TESO1750
TESO1760
TESO1770
TESO1780
TESO1790
TESO1800
TESO1810
TESO1820
TESO1830
TESO1840
TESO1850
TESO1860
TESO1870
TESO1880
TESO1890
TESO1900
TESO1910
TESO1920
TESO1930
TESO1940
TESO1950
TESO1960
TESO1970
TESO1980
TESO1990
TESO2000
TESO2010
TESO2020
TESO2030
TESO2040
TESO2050
TESO2060
TESO2070
TESO2080
TESO2090
TESO2100
TESO2110
WRITE PROGRAM WITH REFERENCE NUMBERS

321 DO 32 1=1,NSTMT
32 WRITE(*,109) I,(SFILF(I,J),J=1,80)
10 FORMAT(*,14,10X,80A1)
34 WRITE(6,111)
111 FORMAT('G1', 'INPUT REFERENCE NUMBER NOW')
35 KNT=0
36 READ(5,101,END=331)CARD
37 CALL CRUNCH(ILNGTH)
38 IF(CARDP(I).EQ.*ASTRSK) GO TO 35
39 DO 237 1=1,2
40 DO 235 J=1,16
41 WRITE(6,108)
42 CONTINUE
43 GO TO 34
44 CONVVERT ALPHA CHARACTER TO NUMBER
45 KNT=KNT+1
46 IF(KNT.EQ.1) NREF=J-1
47 IF(KNT.EQ.2) NREF=10*NREF+(J-1)
48 CONTINUE
49 CHECK FOR LEGAL REFERENCE NUMBER
50 IF(NREF.LE.NSTMT.AND.NREF.GE.0) GO TO 36
51 WRITE(6,106)
52 CONTINUE
53 WRITE(6,112)
54 IF(CARDP(I).EQ.*BLANK) GO TO 36
55 WRITE(6,113)
56 READ(5,101,END=36)CARD
57 CALL CRUNCH(ILNGTH)
58 IF(CARDP(I).EQ.*ASTRSK) GO TO 36
59 DELETE CURRENT LINE
60 IF(CARDP(I).EQ.*MINUS) GO TO 50
61 ADD A STATEMENT AFTER CURRENT LINE
62 IF(CARDP(I).EQ.*PLUS) GO TO 55
CORRECT CURRENT LINE

DEBUG=0,0
INTERP=1
IXERR=0
DO 370 I=1,80
IF(CARDP(I).EQ.ALPHA(14).AND.CARDP(I+3).EQ.ALPHA(20)) GO TO 39
370 CONTINUE

IF NO ERRORS , FILE INPUT STATEMENT
CALL CCMPLR(INFOR,IAARRAY,ILNGTH,INBIG)
IF(NERRS.EQ.0) GO TO 39
NERRS=0
GO TO 34
39 DO 40 J=1,80
40 SFILE(NREP,J)=CARD(J)
GO TO 47
43 WRITE(6,106)
GO TO 208
44 WRITE(6,113)
113 FORMAT(10,T10X,'IF YOU DESIRE TO HAVE THE PROGRAM RUN WITH
*THE DEBUG FEATURE REPLY: YES ; OTHERWISE REPLY: NO AND THIS */
*PROGRAM WILL BE LOST *'//
CALL CRUNCH(ILNGTH)
45 READ(5,101,END=346)CARD
IF(CARDP(I).NE.ALPHA(25)) GO TO 46
EXECUTE WITH DEBUG FEATURE ADDED
CALL INITIAL
DEBUG=1,0
GO TO 136
46 IF(CARDP(I).EQ.ALPHA(14)) GO TO 26
346 WRITE(6,106)
GO TO 45
47 WRITE(6,114)
114 FORMAT(10,T10X,'MORE CORRECTIONS TO BE MADE ?? REPLY: YES */
*OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED *'//'
48 READ(5,101,END=348)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(I).EQ.ALPHA(25)) GO TO 321
348 WRITE(6,116)
GO TO 48
DELETE STATEMENT
DO 50 NSTMT=NSTMT+1, NSTATEM
GO TO 52, 65
DO 52 NSTMT=NSTMT+1, NSTATEM
GO TO 3, 65
IF(NSTATEM.EQ.99) GO TO 65
ADD A BASIC STATEMENT AFTER CURRENT LINE

FIN BEGINNING OF STATEMENT
DO 56 K=1,80
IF(CARD(K).EQ.BLANK) GO TO 56
IF(CARD(K).EQ.IFALPHA(3)) GO TO 56
IF(CARD(K).EQ.IFALPHA(4)) GO TO 56
IF(CARD(K).EQ.IFALPHA(5)) GO TO 56
IF(CARD(K).EQ.KSP) GO TO 56
IF(CARD(K).EQ.IFALPHA(6)) GO TO 56
IF(CARD(K).EQ.IFALPHA(7)) GO TO 56
IF(CARD(K).EQ.IFALPHA(8)) GO TO 56
IF(CARD(K).EQ.IFALPHA(9)) GO TO 56
IF(CARD(K).EQ.IFALPHA(10)) GO TO 56
IF(CARD(K).EQ.IFALPHA(11)) GO TO 56
IF(CARD(K).EQ.IFALPHA(12)) GO TO 56
IF(CARD(K).EQ.IFALPHA(13)) GO TO 56
IF(CARD(K).EQ.IFALPHA(14)) GO TO 56
IF(CARD(K).EQ.IFALPHA(15)) GO TO 56
IF(CARD(K).EQ.IFALPHA(16)) GO TO 56
IF(CARD(K).EQ.IFALPHA(17)) GO TO 56
IF(CARD(K).EQ.IFALPHA(18)) GO TO 56
IF(CARD(K).EQ.IFALPHA(19)) GO TO 56
IF(CARD(K).EQ.IFALPHA(20)) GO TO 56
CONTINUE
CALL COMPLR(INFOR, ARRAY, LNGTH, INBIG)
CALL CONTINUE(106)
IF NO ERRORS THEN FILE STATEMENT
NEARER(S.EQ.1) GO TO 602
NEARER(S.EQ.3) GO TO 602
NEARER(S.EQ.5) GO TO 602
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
ADD STATEMENT TO PROGRAM FILE· MOVE STATEMENTS UP AND DOWN
61 DO 62 I=1,80  
62 SFILE(IITEMP,I)=SFILE(IITEMP-1,I)  
       ITEMP=ITEMP-1  
       IF(ITEMP.GT.IADD) GO TO 61  
DO 63 J=1,80  
63 SFILE(IADD,J)=CARD(J)  
   GO TO 47  
65 WRITE(6,512)  
   GO TO 47  
END

SUBROUTINE CRUNCH(ILNGTH)  
C  
GIVEN THE VECTOR CARD ; THIS SUBROUTINE REMOVES ALL BLANKS IN CARDC  
AND RETURNS THE BLANK-LESS VERSION IN CARDPC  
THIS ROUTINE ALSO CHECKS FOR TYPING ERRORS AND CHECKS IF THE USERC  
IS COMPLETED WITH HIS SESSION OR IN AN EDIT MODE  
C  
COMMON
-  
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48), CRU00090
-  
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5), CRU00120
-  
DIGIT(10), IPRTIB(10), LIST(100), ISTRING(100), CRU00110
-  
PRP2(500), NERRS, INST, NSTRING, DEBUG, DOLGDN, QUOTE, CRU00120
-  
EQUALS, PARL, DECIMAL, PLUS, CMINUS, SLASH, COMMA, CRU00160
-  
PARLET, ASTRAS, BLANK  
COMMON  
INTERP, IXERR  
DO 19 I=1,80  
19 CARDP(I)=BLANK  
C  
REMOKE BLANKS
C  
ILNGTH=0  
DO 20 I=1,80  
20 IF (CARD(I).EQ.BLANK) GO TO 20  
   ILNGTH=ILNGTH+1  
   CARDP(ILNGTH)=CARD(I)  
   CONTINUE  
C  
CHECK FOR BLANK INPUT
   IF (ILNGTH.EQ.0) CARDP(1)=ASTRAS  
   CONTINUE  
   IF (CARDP(1).NE.DOLGDN.OR.CARDP(1+1).NE.DOLGDN) GO TO 26  
   IF (CARDP(1+2).EQ.DOLGDN.AND.CARDP(1+3).EQ.DOLGDN) CARDP(1)=ASTRAS  
   CONTINUE
IAPTR=0
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
DO 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
DO 12 I=1,10
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 PRT(I)=0.0

C
LOAD SPECIAL CHARACTERS
ASTRISK=CHARTM(1)
BLANK =CHARTM(2)
COMMA =CHARTM(3)
DECHAL =CHARTM(4)
EQUALS =CHARTM(5)
PARRT =CHARTM(6)
PARLEFT=CHARTM(7)
PLUS=CHARTM(8)
QUOTE=CHARTM(9)
CGLSGN=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.

COMMON
- STACK(100), PROG(200), CARD(80), CARDP(80), ALPHA(48),
- IAPTR, INPTR, IDATA(50), XDATA(50), STRNG(5),
- DIGIT(10), IPRITB(10), LISTST(100), ISSLST(100),
- PTR(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
- EQUALS, PARRT, DECIMAL, PLUS, CHINUS, SLASH, COMMA,
- PARLEFT, ASTRKS, BLANK

COMMON INTERP, IERROR
REAL*8 DUMY(128)
REAL*8 LES1, LESON1, ' /
WRITE(6, 100)
100 FORMAT('05x', '*** LESSON 1. ***', '///', 'LES00140')
* THIS INSTRUCTION SET WILL INTRODUCE YOU TO THE STRUCTURE OF BASIC
* / LANGUAGE STATEMENTS, THE RULES FOR VARIABLES AND NUMBERS,
* / THE SYMBOLS FOR ARITHMETIC OPERATIONS, /// ***COUNT FORGET TO TLE
* PE GO AND HIT RETURN WHEN READY TO CONTINUE. *** ///
READ(5, 101, END=301) CARD
101 FORMAT(80A1)
102 FORMAT('05x', 'A. PROGRAM STRUCTURE', '///', 'LES00230')
* 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*///10X,*REM PROGRAM TO COMPUTE GAS MILAGE
* ///10X,*REM MILES TRAVELED  = GAS USED/10X,*READ H,G/10X,*LES00290
* ET T=M / G / 10X,*PRINT "MILES TRAVELED":**GAS USED":**MILES/LES00330
* GAL":**MILES/10X,*PRINT M,G,T":**MILES/LES00330
* /* PUT PROPER IS : */ MILES TRAVELED GAS USED MILES/GAL":**MILES/LES00320
100, 11X", 10X", 20X", **MILES/LES00330
READ(5, 103, END=303) CARD
303 WRITE(6, 103)
103 FORMAT('05x', 'AS YOU CAN SEE FROM THE ABOVE SAMPLE, THE PROGRAM
*/// STRUCTURE CONSISTS OF BASIC LANGUAGE STATEMENTS FOLLOWED
*/// LES00370
*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=304ICARD

304 WRITE(6,104)
104 FORMAT(1X,'MOST OF THE KEY WORDS USED IN THE BASIC
*STATEMENTS ARE SELF-EXPLANATORY:
*REM ALLOWS REMARKS/COMMENTS
*READ M,G ASSIGN NUMBERS IN THE DATA STATEMENT TO THE
**VARAIBLES M AND G
**LET ASSIGN THE RESULT OF M DIVIDED BY G INTO VARIABLE T
**PRINT STRING" CAUSES THE STRING IN SINGLE QUOTES TO BE
**PRINT LITERALLY
**END TELLS THE COMPUTER THAT THE INPUT PROGRAM IS TO BE
**EXECUTED."*/
READ(5,101)END=305ICARD

305 WRITE(6,105)
105 FORMAT(1X,'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=306ICARD
CALL CRUNCH(LENGTH)
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(1X,'*** YOUR REPLY WAS TYPED INCORRECTLY ; CHECK THE
*QUESTION AND REPLY AGAIN ***")
GO TO 10

EXECUTE SAMPLE PROGRAM

C

15 WRITE(6,107)
107 FORMAT(1X,'TO EXECUTE THE SAMPLE PROGRAM TYPE IN THE BASIC
**THE STATEMENTS APPEAR AND DON'T 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
**YOUR NEXT INPUT. WHEN THE END STATEMENT IS INPUT THE PROGRAM
**IS EXECUTED."*/
READ(5,101)END=308ICARD

308 WRITE(6,108)
108 FORMAT(1X,'HOWEVER, IF YOU MAKE AN ERROR THE COMPUTER WILL
**TELL YOU THE ERROR, AND EXPECT A CORRECTION. DON'T WORRY ABOUT
**THE ERROR. CHECK YOUR INPUT AGAINST THE SAMPLE PROGRAM AND
**INPUT THE CORRECT STATEMENT. IF YOU MAKE A TYPING MISTAKE
**INPUT FOUR ASTERIX (****). HIT RETURN AND THEN INPUT THE
**CORRECT STATEMENT. */10X,' YOU MAY USE ANY PAIR OF INTEGER OR DECILES00820
*HALF NUMBERS I* FOR INPUT DATA YOU MAY OMIT THE REM STATEMENTS AND LES00830
* THE PRINT "H'* LED TRAVELING ETC"* STATEMENT IF YOU DONT WANTLES00840
* TO TYPE A LOT OF STATEMENTS I** // STATEMENT IF YOU DONT WANTLES00850
CALL TESTI
LES00860
25 WRITE('111')
LES00870
111 FORMAT('15X,'B* PROGRAM FORMAT* // 5X,' A BASIC PROGRAM CONSISTS
* OF A SEQUENCE OF BASIC STATEMENTS * ONE* STATEMENT PER INPUT LILES00890
**NE ' FOLLORED BY AN END STATEMENT* // BECAUSE NO BASIC STATEMENTSLES00930
* MAY BE LONGER THAN ONE INPUT LINE (80 SPACES) I** // THERE IS NO PLES00910
* DEFAULT FOR CONTINUING STATEMENTS FROM ONE LINE I" TO THE NEXT I HLES00920
*GNEVER : YOU MAY SPACE THE INPUT LINE I" AS DESIRED FOR READABILI LILES00930
*TY SINCE THE COMPUTER IGNORES BLANKS IN BASIC */* //LES00940
READ(5,101,END=309)CARD
LES00950
309 WRITE('112')
LES00960
112 FORMAT('15X,'I0X,' STATEMENT NUMBERS
* EACH STATEMENT MAY HAVE AN OPTIONAL */ LES00980
* STATEMENT NUMBER PRECEDING IT FOR IDENTIFICATION PURPOSES */ LES00990
* THIS STATEMENT NUMBER MUST BE AN INTEGER BETWEEN 1 -> 9999 */ LES01000
5X, FOR EXAMPLE : 12 READ M,G */ LES01100
*I0X,' 2 KEY WORDS */ THE KEY WORDS THAT MAKE UP A LES01020
*BASIC STATEMENT (REM ,READ ,LET ,ETC),I* ARE SPECIAL TERMINAL SYM LES01030
*BCLS THAT ARE RECOGNIZED BY THE COMPUTER I* AND FOR THIS REASON THES01040
*EY MUST BE SPELLED CORRECTLY AND ONLY*/* USED IN BASIC STATEMENTS LES01050
*//19X,' THE END STATEMENT INDICATES THE INPUT PROGRAM IS*/ LES01060
*COMPLETED AND THAT PROGRAM EXECUTION IS TO BEGIN I THE **END** I* LES01070
*STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM */ LES01080
READ(5,101,END=313)CARD
LES01090
313 WRITE('113')
LES01100
113 FORMAT('10X,' YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT*/ LES01110
* WHAT YOU HAVE JUST LEARNED */ LES01120
WRITE(5,114)
LES01130
114 FORMAT('10X, ' IF THIS IS A LEGAL BASIC PROGRAM(REPLY : YES OR LES01140
*NO) */ 5X , REM ONE LINE DO NOTHING PROGRAM I5X,' END I*/ LES01150
30 READ(5,101,END=315)CARD
CALL CRUNCH(LENGTHI)
IF(CARDP(1).EQ.ASTRSK) GO TO 30
IF(CARDP(1).EQ.ASCII14.0R.CARDP(1).EQ.ASCII251) GO TO 31
315 WRITE(5,106)
LES01200
31 GO TO 30
30 WRITE(5,115)
LES01220
315 FORMAT('10X,' THE SIMPLEST BASIC PROGRAM CONSISTS OF JUST AN I** END** STATEMENT, I* LES01230
WRITE(5,116)
LES01240
116 FORMAT('10X, ' WHICH OF THE FOLLOWING BASIC STATEMENTS IS*/ LES01250
* IN THE PROPER FORMAT :I/15X,' A=15 LET T=M/G I15X,' B=I LET T=M/G I15X,' C=0 LET T=M/G I15X,' I**ALL** I* LES01270
32 READ(5,101,END=317)CARD
LES01280
LES01290
LES01300
CALL CRUNCH(ILNGTH)
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(1X) ALL OF THE ABOVE BASIC STATEMENTS ARE CORRECT BECAUSE
* * * SPACES ARE DISREGARDED. NOTE THAT STATEMENT B. * * *
* CORRECT IS CONFUSING TO READ. BLANKS AND INDENTATIONS MAKE A
* PROGRAM EASY TO READ. */
WRITE(6,218)
218 FORMAT(1X,5X,'C. ALPHANUMERIC CHARACTERS // 5X,')
* ALPHANUMERIC CHARACTERS ARE THE LEGAL CHARACTERS, Digits, AND
* 1. 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 ** * / + - ( ) = $ */
* STRING/* A STRING IS ANY LIST OF ALPHANUMERIC CHARACTERS*]
* ENCLOSED IN **SINGLE** QUOTES. FOR EXAMPLE */5X,* "THIS IS ALESO1520"
* STRING */
READ(5,101,END=318)CARD
88
318 WRITE(6,118) 118 FORMAT(1X,6X,'D. VARIABLES //
* 10X,1 IN BASIC THERE ARE THREE TYPES OF VARIABLES: SIMPLE, ALPHALESSO1580"
* AND SUBSCRIPTED. SUBSCRIPTED VARIABLES WILL BE COVERED IN */
* LESSON 5. */10X,1 SIMPLE VARIABLES
* SIMPLE VARIABLES ARE IDENTIFIED BY A SINGLE LETTER*
* OR A SINGLE LETTER FOLLOWED BY A DIGIT BETWEEN 0→9
* 5X,1 FOR EXAMPLE: A, A3, Z0, AND X ARE LEGAL VARIABLES; BUT
* 20X,1, 128, 192, ABC, AND XYZ ARE ILLEGAL VARIABLES. */
* THEREFORE
* USE: YOU HAVE SIMPLE VARIABLES FOR USE IN YOUR PROGRAMS. */
READ(5,101,END=319)CARD
319 WRITE(6,119) 119 FORMAT(1X,10X,2.2. ALPHAVARIABLES
* ALPHAVARIABLES ARE USED FOR ALPHANUMERIC MANIPULATION
* LATIONS/* IN WHICH A GROUP OF ALPHANUMERIC CHARACTERS, CALLED
* A STRING, ARE REPRESENTED BY AN ALPHAVARIABLE. THE ALPHALESSO1700"
* VARIABLE CONSISTS OF A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN
* THE MAXIMUM LENGTH OF THE ALPHANUMERIC STRING ASSIGNED
* TO THE ALPHAVARIABLE IS 16 CHARACTERS. */
* FOR EXAMPLE: A$, X$, Z$ ARE LEGAL ALPHAVARIABLES*]
* A SIMPLE SELECTION OF ALPHAVARIABLES FOLLOWS: */5X,* READ A$, B$, *]
* PRINT A$, B$, 5X,* DATA "MONDAY","21 JUNE","5X,* END */
* PROGRAM WILL PRODUCE THE OUTPUT: */10X,"MONDAY 21 JUNE"/}
LES01780
READ(5,101,END=320)CARD
320 WRITE(6,120)   LES01790
120 FORMAT(*'5X,'E NUMBERS
*10X,'* NUMBERS MAY BE EXPRESSED AS INTEGERS OR AS REALS
*AND A'*'//LES01820
*NUMBER WHETHER INTEGER OR REAL IS LIMITED TO 9 DIGITS, NOT INCLUDES
*ING*/'** DECIMAL POINT** A NUMBER IS ASSUMED POSITIVE UNLESS IT */'//LES01840
*IS PRECEDED BY A - SIGN
*10X,'*1. INTEGERS
*RS ARE NUMBERS WITH NO FRACTIONAL PART, IE: 3, 15
*10X,'* 2. REALS
* REAL (FIXED-POINT) NUMBERS HAVE A DECIMAL POINT AND A FRACTIONAL
*/' PART, IE: 3.0, 15.31, 729.1, 0.0'//)
*READ=101 END=321)CARD
321 WRITE(6,121)   LES01900
121 FORMAT(*'5X,'E EXPRESSIONS
*AN EXPRESSION MAY BE A SINGLE CONSTANT OR VARIABLE, OR AN */'//LES01930
*ARITHMETIC EXPRESSION. ARITHMETIC EXPRESSIONS ARE FORMED BY */'//LES01940
*USING OPERATORS AND PARENTHESIS
*10X,'* NUMBERS AND SIMPLE VARIABLES MAY BE COMBINED INTO ARITHMETIC
*10X,'* EXPRESSIONS BY USING ONE OR MORE OF THE ARITHMETIC OPERATORS */'//LES01970
*10X,'* EXPONETIATION*/5X,'* MULTIPLICATION*/5X,'* DIVISION
*/'//LES01990
*ON*/5X,'* ADDITION*/5X,'* SUBTRACTION
*/'//LES02000
*IN WRITING AN EXPRESSION, EACH ARITHMETIC OPERATION MUST BE SHOWN
*/' FOR EXAMPLE: 2+2, A/B
*8€6 */2 AND T-3.0 ARE LEGAL EXPRESSIONS
*/'//LES02030
*READ=101 END=322)CARD
322 WRITE(6,122)   LES02040
122 FORMAT(*'5X,'E PARENTHESIS
* PARENTHESIS MAY BE USED TO GROUP EXPRESSIONS AND */'//LES02070
* TO CONTROL THE ORDER IN WHICH AN ARITHMETIC EXPRESSION IS EVALUATED
* THE NORMAL HIERARCHY OF OPERATORS IS: */'//LES02090
* 2. */'//LES02100
* 1. */'//LES02120
*IF PARENTHESIS ARE USED TO GROUP EXPRESSIONS, EXPRESSIONS MUST BE SHOWN */'//LES02130
* IN ORDER TO EVALUATE FIRST. PARENTHESIS IN EXPRESSIONS MUST ALWAYS
* OCCUR IN PAIRS, IE: 5 * (4 + 3), (5*4+3) ARE EQUIVALENT
* EXPRESSIONS
*/'//LES02140
*OPERATIONS ARE EVALUATED ACCORDING TO THE OPERATORS HIERARCHY
*/'//LES02160
*IF TWO OPERATORS OF THE SAME HIERARCHY OCCUR IN AN EXPRESSION
*/'//LES02170
*EVALUATION IS FROM LEFT TO RIGHT, IE: 2*4+6/2
*IN THIS EXPRESSION 2*4 WILL BE EVALUATED FIRST, THEN 6/2
* AND THEN THE TWO RESULTS WILL BE ADDED
*/'//LES02190
*READ=101 END=323)CARD
323 WRITE(6,113)   LES02200
113 WRITE(6,123)   LES02210
123 FORMAT(*'5X,'E 4 + 6 / 2 HAS THE VALUE (REPLY WITH VALUE)
*/'//LES02240
*"*/LES02250
34 READ(5,101,END=324)CARD

35
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 34
IF(CARDP(1).EQ.DIGIT(8)) GO TO 37
DO 35 I=1,10
IF(CARDP(1).EQ.DIGIT(1)) GO TO 36
35 CONTINUE
324 WRITE(6,106)
GO TO 34
124 FORMAT('0**1** YOUR ANSWER IS INCORRECT. THE EXPRESSION IS EVALUATE
*D AS FOLLOWS :')
36 WRITE(6,124)
WRITE(6,125)
125 FORMAT(10X,'4+6/2 -> 4+3 -> 7')
37 WRITE(6,126)
126 FORMAT(10X,'(4+6)/2 HAS THE VALUE (REPLY WITH VALUE)')
38 READ(5,101,END=397)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 38
IF(CARDP(1).EQ.DIGIT(6)) GO TO 41
DO 39 I=1,16
IF(CARDP(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(10X,'(4+6)/2 -> 10/2 -> 5')
41 WRITE(6,128)
128 FORMAT(10X,'(4+6)/2**2 HAS THE VALUE (REPLY : VALUE)')
42 READ(5,101,END=46)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 42
IF(CARDP(1).EQ.DIGIT(3).AND.CARDP(2).EQ.DIGIT(6)) GO TO 48
DO 43 I=1,10
IF(CARDP(1).EQ.DIGIT(1)) GO TO 44
43 CONTINUE
GO TO 46
44 CONTINUE
DO 45 I=1,10
IF(CARDP(2).EQ.DIGIT(1)) GO TO 47
45 CONTINUE
46 WRITE(6,106)
GO TO 42
47 WRITE(6,124)
WRITE(6,129)
48 WRITE(6,130)
129 FORMAT(10X,'(4+6)/2**2 -> (10/2)**2 -> (5)**2 -> 25')
48 WRITE(6,130)
130 FORMAT('**', 5X, 'F. SUMMARY
** THIS CONCLUDES THE INSTRUCTION SET FOR LESSON 1.**LES02750
** YOU WILL NOW BE GIVEN A SIMPLE BASIC PROGRAM AND ASKED TO FIND**LES02760
** THE MISTAKES IN IT CONCERNING WHAT YOU HAVE LEARNED IN THIS**LES02780
** LESSON. **/5X, '1 REM REVIEW PROGRAM */5X,' 2 READ A1,B2,Z1;LES02790
**Y*/5X,' 3 LET A1=B2+A1+Z1 */5X,' 4 LET B2=A1+2;LES02790
**Y*/5X,' 5 LET Z1=3,5,41952763 */5X,' 6 PRINT A1,B2,Z1 */5X,' 7 DATA 50,0.3;LES02810
**1416, **FINIT**/ 'AFTER LOOKING AT THE SAMPLE PROBLEM, YOU WILL BLE502820
** E ASKED QUESTIONS**/ ' ABOUT EACH STATEMENT;**/LES02830
** READ(5,101,END=33)CARD
** WRITE(6,131)
131 FORMAT('**', 10X, '1.) LINE 2 CONTAINS A READ STATEMENT FOLLOWED*/LES02860
** BY A LIST OF VARIABLES. ANY ERRORS (REPLY: YES OR NO)?**/LES02870
** READ(5,101,END=33)CARD
** CALL CRUNCH(LIGHT)
** IF(CARDP(1).EQ.ASTRSK) GO TO 50
** IF(CARDP(1).EQ.ALPHA(14)) GO TO 52
** IF(CARDP(1).EQ.ALPHA(25)) GO TO 51
** WRITE(6,106)
** GO TO 50
51 WRITE(6,132)
132 FORMAT('**', 10X, '2.) INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY.'/LES02950
** READ(5,101,END=52)CARD
** CALL CRUNCH(LIGHT)
** IF(CARDP(1).EQ.ASTRSK) GO TO 510
** IF(CARDP(1).EQ.ALPHA(14)) GO TO 52
** IF(CARDP(1).EQ.ALPHA(25)) GO TO 51
** WRITE(6,133)
** 133 FORMAT('**', 10X, '3.) THE ILLEGAL VARIABLE IS SY. SIMPLE VARIABLES ARE */LES03020
** A LETTER OR A LETTER FOLLOWED BY A SINGLE DIGIT. ALPHA VARIABLES**LES03030
**$*/ ARE A LETTER FOLLOWED BY A DOLLAR SIGN. $, IE. Y$.*1)
** WRITE(6,144)
134 FORMAT('**', 10X, '4.) LINE 3 CONTAINS A LET STATEMENT FOLLOWED*/LES03050
** BY A1 = EXPRESSION. ANY ERRORS (REPLY: YES OR NO)?**/LES03050
** READ(5,101,END=343)CARD
** CALL CRUNCH(LIGHT)
** IF(CARDP(1).EQ.ASTRSK) GO TO 54
** IF(CARDP(1).EQ.ALPHA(14)) GO TO 56
** IF(CARDP(1).EQ.ALPHA(25)) GO TO 55
** WRITE(6,106)
** GO TO 54
54 WRITE(6,134)
134 FORMAT('**', 10X, '5.) INPUT CORRECTION TO ILLEGAL EXPRESSION EVERYTHING*/LES03150
** AFTER = SIGN */1)
56 READ(5,101,END=58)CARD
** CALL CRUNCH(LIGHT)
** IF(CARDP(1).EQ.ASTRSK) GO TO 56
** IF(CARDP(1).EQ.ALPHA(12), AND CARDP(2).EQ.DIGIT(3)) GO TO 57
** IF(CARDP(1).EQ.ASTRSK) GO TO 56
** IF(CARDP(1).EQ.ALPHA(12), AND CARDP(2).EQ.DIGIT(3)) GO TO 57
** IF(CARDP(1).EQ.ASTRSK) GO TO 56
** IF(CARDP(1).EQ.ALPHA(12), AND CARDP(2).EQ.DIGIT(3)) GO TO 57
57 WRITE(6,134)
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. PARR(1)) GO TO 59
58 WRITE(6,135) LES03230
135 FORMAT('0', * 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) LES03290
136 FORMAT('0', *CX, ' 3,) LINE 4 IS SIMILAR TO LINE 3, ANY ERRORS? */ * REPLY , YES OR NO) ? */)
60 READ(5,101,END=361) CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1),EQ.ASTRSK) 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) LES03360
64 WRITE(6,137) LES03390
137 FORMAT('0', * THE EXPRESSION IS CORRECT */)
65 WRITE(6,138) LES03310
138 FORMAT('0', *CX, ' 4;) LINE 5 CONTAINS A LET STATEMENT FOLLOWED/*/ * BY A NUMBER; ANY ERRORS (REPLY : YES OR NO) ? */)
70 READ(5,101,END=361) CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 70
IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 71
371 WRITE(6,106) LES03360
71 WRITE(6,139) LES03320
139 FORMAT('0', * THE NUMBER CONTAINS 10 DIGITS AND THE MAXIMUM ALLOWED */ * IS 9 DIGITS. */)
72 WRITE(6,140) LES03350
140 FORMAT('0', *CX, ' 5;) 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(LNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 72
IF(CARDP(1).EQ.ALPHA(14)) GO TO 74
IF(CARDP(1).EQ.ALPHA(25)) GO TO 73
372 WRITE(6,106) LES03360
73 WRITE(6,132) LES03350
730 READ(5,101,END=74) CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 730
IF(CARDP(1).EQ.ALPHA(1).AND.CARDP(2).EQ.DIGIT(2)) GO TO 75
74 WRITE(6,141) LES03380
141 FORMAT('0', * THE ILLEGAL VARIABLE IS 1A; SIMPLE VARIABLES ARE A */ LES03320
```plaintext
* LETTER OR A LETTER FOLLOWED BY A SINGLE DIGIT ,IE. A1 *)
75 WRITE(C,142)
142 FORMAT(*'10X: THERE ARE NO ERRORS IN LINE 7: IS THE PROGRAM'/* (REPLY: Y)
* READY TO EXECUTE (ASSUMING ABOVE ERRORS CORRECTED)'/')
* S OR NO'*/)
750 READ(5,101,END=80)CARD
CALL CRUNCHLENGTH
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(*'10,X,' THE PROGRAM FORMAT REQUIRES THAT AN END STATEMENT 
*'/')
*/'/ BE THE LAST STATEMENT OF THE PROGRAM - THIS SAMPLE PROGRAM* '/')
* M'/') WOULD NOT EXECUTE */'*' THIS CONCLUDES THE REVIEW QUESTIONS*'/'
* FROM LESSON 1 */'/')
CALL EXIT
END

COMMON
STACK(100), PROC(200), CARD(80), CARDP(80), ALPHA(48),
- IAPTR, INPTR, IADATA(500), XDATA(500), STRING(5),
- DIGIT(10), IPRINT(10), LISTS(100), ISTRS(100),
- PRT(2500), NERRS, INST, NSTL, DEBUG, DOLGEN, QUOTE,
- EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
- PARLET, ASTRSK, BLANK
COMMON INTERP, IXERR
REAL*8 LES2, LESEN, '
CALL ALOAD(LES2, 1)
WRITE(6,10C)
100 FORMAT(*'10,X,' *** LESSON 2. ***
* THIS INSTRUCTION SEQUENCE WILL INTRODUCE YOU TO THE BASIC LANGUAGE*'/)
* All STATEMENTS REM, PRINT, READ, AND DATA, USING THESE STATEMENTS* '/
* ENS '*/ YOU WILL BE ABLE TO CONSTRUCT AND EXECUTE ELEMENTARY PROGRAMS*'/
* AMS '*/ 1X THE FORM FOR EACH BASIC STATEMENT WILL INCLUDE /*'/
* X,'** KEY WORD' ** ELEMTS OR LST ELEMTS SEPARATED BY COMMAS**'/
* $'
* THE CARET SYMBOLS '<< >>' DELINATE THE LEGAL ITEMS THAT MAY*/
* FOLLW THE KEYWORD AND MAKE UP THE BASIC STATEMENT */'/
READ(5,101,END=3C2)CARD
101 FORMAT(80A1)
302 WRITE(6,102)
102 FORMAT(*'10*X, ' A REM
* /*5X,' THE BASIC STATEMENT WHICH ALLOWS YOU TO INSERT */LES00280
* REMARKS INTO YOUR PROGRAM IS IDENTIFIED BY THE KEY WORD 'REM'**'/
* '* REM IS A NON-EXECUTING STATEMENT WHICH MAY BE USED OPTIONALLY*'/LES00350
```
* FACILITY TO EXECUTE YOUR FIRST PROGRAMS. FOR EXAMPLE: /*10X;* LE500790
*REM PROGRAM TO COMPUTE THE SQUARE OF A NUMBER */10X;* LE500800
*PRINT'5 SQUARE=' 5**2/10X; END */5X;* LE500810
*PRODUCE SQUARE RESULT:/'5X;* 5 SQUARED = 25 */5X;* LE500820
*YOU WILL NOW BE GIVEN 10 CUBIC PROBLEMS TO SOLVE. YOU WILL ENTER /* LE500830
*THE EXECUTION PHASE OF CAIBASIC WHERE YOU CAN RUN YOUR PROBLEMS' /* LE500840
*AND THEN YOU WILL RETURN TO FINISH THE LESSON. /*5X;*'1) FIND THE LE500850
*SQUARE ROOT OF 5 SQUARED MINUS + TIMES 2 TIMES 2 */5X;*'2) FINDLE500860
*THE VALUE OF 3.141628 CUBED)1/12/* WHERE B=2.50, H=3.03 */77)
309 WRITE(6,109)
109 FORMAT('01, IF YOU WISH TO SKIP THESE PROBLEMS REPLY: YES */
*AND THE LESSON WILL CONTINUE.*/)
5 READ15,101,END=309)
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.*ASTR)*GO TO 5
IF(CARDP(1).EQ.*ALPHA(25)*GO TO 10
CALL TEST
310 WRITE(6,110)
110 FORMAT('01, REPLY WITH THE ANSWER TO THE FIRST PROBLEM */
6 READ(5,101,END=310)
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.*ASTR)*GO TO 5
IF(CARDP(1).EQ.*DIGIT(4)*GO TO 7
WRITE(6,111)
111 FORMAT('01, YOUR ANSWER IS WRONG. YOUR PRINT STATEMENT SHOULD */
*HAVE BEEN PRINT ((5**2)-(4**2*2))**5/*
7 WRITE(6,112)
112 FORMAT('01, REPLY WITH THE ANSWER TO THE SECOND PROBLEM */
8 READ(5,101,END=7)
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.*DIGIT(2)*AND.CARDP(2).EQ.*DIGIT(3)*GO TO 9
IF(CARDP(1).EQ.*DIGIT(4)*GO TO 10
9 WRITE(6,113)
113 FORMAT('01, 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('01, /* LE501150
*THE READ STATEMENT IS THE METHOD WHICH PROVIDES INPUT TO THE /* LE501160
*PROGRAM. THE FORM OF THE READ STATEMENT IS: /*5X;* LE501170
*READ '/* VARIABLE,'/* '/* VARIABLE */ LE501180
*FOR EVERY VARIABLE IN THE READ LIST THERE MUST BE A CORRESPONDING */ LE501190
*/ ELEMENT IN A DATA STATEMENT. THE READ AND DATA STATEMENTS ARE LE501200
*/ USED TOGETHER TO ASSIGN INPUT VALUES TO PROGRAM VARIABLES. */ 2X
*/ WHEN THE READ STATEMENT IS EXECUTED, EACH VARIABLE IS ASSIGNED LE501220
*/ SUCCESSIVE NUMBERS FROM A STACK OF NUMERIC DATA OR SUCCESSIVE LE501230
*" STRINGS" FROM A STACK OF ALPHANUMERIC DATA. AS EACH VARIABLE LE501240
*IS READ, IT TAKES THE TOP ELEMENT OF THE APPROPRIATE DATA. */ LE501250
*STACK */77)
READ(5,101,END=315)CARD

115 WRITE(6,115)LES01270
*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."

READ(5,101,END=316)CARD

136 WRITE(6,136)LES01430
116 FORMAT(58,10X, WHEN THE FIRST DATA STATEMENT IS INTERPRETED */ LES01410
* BY THE CBASIC COMPILER A FIRST IN, FIRST OUT STACK IS FORMED.*/ LES01420
* FOR NUMERIC AND ALPHA-NUMERIC DATA, AS EACH NUMBER OR STRING*/ LES01430
* IN A DATA LIST IS INTERPRETED, IT IS PLACED ON THE BOTTOM OF THE*/ LES01440
* ITS RESPECTIVE DATA STACK, AS OTHER DATA STATEMENTS ARE LOCATED*/ LES01450
* IN THE PROGRAM, ITS ELEMENTS ARE PLACED ON THE BOTTOM OF THE"*/ LES01460
* PROPER STACK. **T0X, ** EXAMPLE: */ "5X," DATA 10.0, 13.33," J. E. S"*/ LES01470
* MITH** 18"/5X," DATA "2", DDE"'/5X,"LES01480
* PRODUCES THE FOLLOWING DATA STACKS: /*"5X,"NUMERIC",10X," ALPHANUMERIC"*/ LES01490
* 18"/5X," 19"*/ LES01520

READ(5,101,END=317)CARD

137 WRITE(6,137)LES01530
117 FORMAT(58,10X, DURING EXECUTION OF READ STATEMENTS, AS THE */ LES01540
* VARIABLES IN THE TOP OF THE DATA STACK ARE ASSIGNED VALUES FROM THE */ LES01550
* THE APPROPRIATE DATA STACK. THE DATA STACK IS DECREMENTED AND*/ LES01560
* THE NEXT ELEMENT POPS UP **T0X, ** EXAMPLE: */ "5X," READ A,B1**: LES01570
* ** ASSUMING THAT THE DATA FROM EXAMPLE 1.1 IS AVAILABLE, THE */ LES01580
* ** VARIABLES IN THE READ LIST ARE ASSIGNED VALUES AS FOLLOWS: */ "5X, LES01590
* ** THE RESULTING DATA STACKS ARE AS FOLLOWS: */ "5X," NUMERIC",10X," ALPHANUMERIC"*/ LES01610
* PHA"/5X," 18",15X," Z", DDE'/5X," 19"*/) LES01620

READ(5,101,END=318)CARD

310 WRITE(6,310)LES01630
118 FORMAT(58,10X, E. RESTORE , RESTORES */ LES01640
* THE RESTORE BRAND STATEMENTS ARE USED TO RETURN THE NUMERIC"*/ LES01650
* AND ALPHA-NUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO */ LES01660
* THAT THE DATA MAY BE USED AGAIN, THE FORM OF THE RESTORE STATEMENTS*/ LES01670
* NT" IS: */ "10X," RESTORE ( RESTORES NUMERIC DATA ) */ "10X," RESLES01680
* TORES ( RESTORES ALPHA-NUMERIC DATA ) */ LES01690
* YOU WILL NEVER BE ASKED SOME QUESTIONS ABOUT THE READ , DATA AND*/ LES01700
* RESTORE STATEMENTS. */ LES01710
* PART OF A BASIC PROGRAM: /*"5X," DATA 2,5,7,16,"ANS"'/5X," DATA LES01720
* "CORRECT",25,"WRONG",0,"5"," READ A,B3,1S,C1,D"/5X," READ J"LES01740
WHAT IS THE VALUE OF C1 ?? REPLY WITH VALUE 1

*WHEN THE READ STATEMENTS ARE EXECUTED, THE VARIABLES ARE ASSESSED AS FOLLOWS:
A <-- 5
Z4 <-- 2
C1 <-- 7
K$ <-- 0.0

*SUMMARY
* TO INPUT VALUES INTO YOUR PROGRAM, AND HOW THE PRINT STATEMENT WORKS TOGETHER.
* IS USED TO OUTPUT AND LABEL RESULTS YOU HAVE THE FACILITY
* TO WRITE SIMPLE PROGRAMS USING INPUT DATA. FOR EXAMPLE:
**COULD BE WRITTEN IN SYMBOLOGIC FORM AND THE DATA
**COULD BE READ IN AS FOLLOWS:
**IN THE NEXT LESSON YOU WILL BE SHOWN HOW TO USE ASSIGNMENT STATEMENTS.
* YOU GREATER FLEXIBILITY IN WRITING EXPRESSIONS.
* AND WILL ALLOW YOU TO DO ASSIGNMENTS SUCH AS:
**
** AND Z=(A**2+3)/A, THEN BY SAYING PRINT A,Z
** EXPRESSIONS WOULD BE DISPLAYED.
**
* YOU WILL NOW BE GIVEN SOME REPRESENTATIVE PROBLEMS.
* TO GIVE YOU A CHANCE TO EXERCISE YOUR NEW PROGRAMMING TOOLS.
* / 10 *'1.5 * WRITE A PROGRAM TO SOLVE THE EQUATION: X**2+10Y-24 = / LESO2240
* WHERE THE INPUT DATA IS X=10, Y=3 */ / 10 * WRITE A PROGRAM LESO2250
* TO SOLVE THE QUADRATIC EQUATION: X / (-B+ S*P2-B**2) / 4A */ 10 * WRITE AND EXECUTE LESO2270
* THESE PROGRAMS NOW */ REPLY: YES */ AND YOU WILL ENTER THE CAIBALESO2280
* SIN COMPILER: OTHERWISE REPLY: NO */ AND YES WILL GO ON TO THE NLESO2290
* EXT LESSON */ */
35 READ(5,101,END=325)CARD
36 CRUNCH(L!NGTH)
37 IF(CARDP(1).EQ.ASTRSK) GO TO 35
38 IF(CARDP(1).EQ.ALPHA(11)) GO TO 45
39 IF(CARDP(1).EQ.ALPHA(14)) GO TO 40
40 WRITE(6,126)
41 WRITE(6,125) ** YOUR REPLY IS INCORRECTLY TYPED: REPLY AS IN ***
42 FORMAT(1,E,10x) ** */ */
43 WRITE(6,126) IF YOU DECIDE TO RUN THESE PROBLEMS LATER THE ANSWERSLESO2400
44 */ WILL NOT BE GIVEN SO YOU MAY CHECK YOUR RESULTS: */ -5X, X
45 CALL EXIT
46 WRITE(6,127)
47 100 FORMAT(10,10X) ** REPLY WITH YOUR ANSWER TO PROBLEM 1. */ */
48 100 READ(5,101,END=328)CARD
49 IF(CARDP(1).EQ.ASTRSK) GO TO 50
50 IF(CARDP(1).EQ.DIGIT(1).AND.CARDP(3).EQ.DIGIT(11)) GO TO 55
51 WRITE(6,128)
52 FORMAT(10,10x) ** THE CORRECT ANSWER IS 106 AND THE PROGRAM SHOULD */ */
53 100 FORMAT(10,10x) ** HAVE BEEN SIMILAR TO */ -5X, X READ X, Y*/5X, X PRINT**ANSWER** X=LESO2520
54 100 FORMAT(10,10x) */ X*/5X, X DATA 0,1,3/*5X, X END //
55 WRITE(6,129)
56 WRITE(6,126) IF YOU DECIDE TO RUN THESE PROBLEMS LATER THE ANSWERSLESO2530
57 CALL EXIT
58 FORMAT(10,10x) ** REPLY WITH YOUR ANSWER TO PROBLEM 2. */ */
59 READ(5,101,END=62)CARD
60 CRUNCH(L!NGTH)
61 IF(CARDP(1).EQ.ASTRSK) GO TO 60
62 IF(CARDP(1).NE.MINUS. AND. CARDP(2).NE.DECIMAL) GO TO 62
63 WRITE(6,130)
64 FORMAT(10,10x) ** THE CORRECT ANSWER IS -50 AND THE PROGRAM SHOULD LESO2540
65 */ LOOK SIMILAR TO */ -5X, X READ A, B, C*/5X, X PRINT**ANSWER**(-BLESO2560
66 */ + (B**2 - 4*A*C)**.5) / 2*A*/5X, X DATA 2.5,2*/5X, X END //
67 CALL EXIT
68 END

C
C LESSON 3 PRESENTS THE 'LET' STATEMENT AND BUILTIN FUNCTIONS

COMMON
- STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48), LESO4010
- LESO4020
100 FORMAT('**4', 5X, '*** LESSON 3, ***
** 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) 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('**4', 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 IS
*ASSIGNED TO THE VARIABLE 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, B=3*/10X, LET B
*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('**4', 10X, 'THE ONLY RESTRICTION ON THE USE OF VARIABLES **
*IS THAT SIMPLE AND SUBSCRIPTED VARIABLES CAN ONLY BE ASSIGNED**
*Numerical values and alpha variables can only be assigned alpha-**
*Numerical strings. For example : **10X, LET A=D4*X(5) [3]**2 -
*6)/14*/10X, LET A=X5='HELP'**10X, LET Y=(B - 4*A*C)**5*/10X
*LET OS='ANSWER'** THE FOLLOWING ASSIGNMENT IS ILLEGAL : **10X
*LET A=D8=3+4**
READ(5, 101, END=305) CARD
305 WRITE(6, 105)
105 FORMAT('**4', 10X, 'YOU WILL NOW BE ASKED QUESTIONS CONCERNING **
*WHAT YOU HAVE JUST LEARNED **/
WRITE(6,106)
106 FORMAT('0',10X,': 1.) REPLY WITH VALUE OF X IN BELOW PROGRAM *///
*0X' LET A=*,/10X', LET Z=0+4,+/10X', LET X=A**2 - 4*B
* + 3*C/10X', PRINT'"ANSWER =**,X'/10X', END*///
5 READ(5,101,END=306) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 5
IF(CARDP(1).EQ.DIGIT(1)) GO TO 10
IF(CARDP(1).EQ.PLUS.AND.CARDP(2).EQ.DIGIT(2)) GO TO 10
306 WRITE(6,107)
107 FORMAT('0',': 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('0',10X,': 2.) REPLY WITH VALUE OF Z IN BELOW PROGRAM *///
*0X', DATA 1,2,3,4,6/10X', DATA'RIGHT ON',7,8/10X', READ A,B,X
*+/10X', READ G5,Y,C0, 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(1)) GO TO 20
309 WRITE(6,109)
109 FORMAT('0',': YOUR RESPONSE WAS INCORRECT • THE VARIABLES ARE */
* ASSIGNED VALUES AS FOLLOWS::*/5X', 'NUMERIC'10X', 'ALPHA'5X', 'AC
* == 1,10X', 'X-- RIGHT ON*/5X', 'BC == 2*/5X', 'X<-- 3*/5X', 'Y<-- 4*/
*5X', 'C-- 6*/5X', D<-- 7*/5X', Z <-- X+Y ; Z <-- 7//)
20 WRITE(6,110)
110 FORMAT('0',10X,': LET R = A+B/(C-0)// WHICH FORMULA DOES THIS */
* STATEMENT REPRESENT /*/10X', 'REPLY WITH CORRECT LETTER'1/10X', 'A
*) R=(A+B)/(C-D)/10X', 'B) R=A*(B/C)-D/1//
25 READ(5,101,END=311) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 25
IF(CARDP(1).EQ.ALPHA(1)) GO TO 30
311 WRITE(6,111)
111 FORMAT('0',': 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('0',10X,': B. BUILT-IN FUNCTIONS */5X',
* BUILT-IN FUNCTIONS ARE COMMONLY USED PROGRAMS ALREADY WRITTEN */
* AND STORED IN THE CAILASIC COMPILER FOR YOUR USE. THERE ARE */
* FUNCTIONS TO FIND SQUARE ROOTS. LOGARITHMS. ABSOLUTE VALUES • */
* AND TRIGONOMETRIC VALUES. THE FORM FOR THE BUILT-IN FUNCTION IS */
* ==//10X', 'FUNCTION NAME << (EXPRESSION) >>' WHERE THE EXPRESSION
*N IS ENCLOSED IN PARENTHESES */
READ(5,101,END=313)CARD
313 WRITE(6,113)
113 FORMAT('01,10X,' THE BUILT-IN FUNCTIONS AND DEFINITIONS ARE: '*/5X
   *, SQR(X) --- SQUARE ROOT OF ARGUMENT (MUST BE POSITIVE) '*/5X,'*
   ABS(X) --- ABSOLUTE VALUE OF ARGUMENT '*/5X,' LOG(X) ---
   NATURAL LOGARITHM OF ARGUMENT '*/5X,' EXP(X) --- EXPONENTIAL FUNCTION
   , VALUE OF 2.71828 ** X '*/5X,' INT(X) --- INTEGER PART
   OF ARGUMENT IS RETURNED '*/5X,' SIN(X) ---
   SINE OF THE ARGUMENT '*/5X,' COS(X) --- COSINE OF ARGUMENT '*/5X,' TAN(X) --- TANGENT OF ARGUMENT '*/5X,' ARC
   TANGENT IN RADIAN'S OF ARGUMENT '*/5X,' IN ALL THE ABOVE BUILT-IN FUNCTION
   IONS THE ARGUMENT IS ANY LEGAL EXPRESSION '*/' AND AS NOTED THE SQ
   R FUNCTION ARGUMENTS REQUIRED A POSITIVE ARGUMENT '*/' THE TRIGONOMETRIC F
   NTIONS REQUIRE AN ARGUMENT VALUE IN RADIANS '*/'/
READ(5,101,END=314)CARD
314 WRITE(6,114)
114 FORMAT('01,10X,' 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 '*/'
   FCR EXAMPLE: '*/5X,' LET Z=SQR(ABS(-5)) IS A CORRECT USE OF BUI
   LT IN FUNCTIONS '*' /
READ(5,101,END=315)CARD
315 WRITE(6,115)
115 FORMAT('01,10X,' THE FOLLOWING PROGRAM IS A EXAMPLE OF HOW TO '*/'
   USE BUILT-IN FUNCTIONS '*/5X,' REM PROGRAM TO COMPUTE SQUARE ROOT
   AND LOGARITHMS '*/5X,' READ A'*/5X,' PRINT 'SQUARE ROOT='*/5X,' Y'*/5X,' LOG='*/5X,' Z'*/5X,' DATA 4.8,7.5
   XEND'*/' PRODUCE RESULT '*/5X,' SQUARE ROOT= 2.16,10X,' LOG = 1.386'/
*/)
READ(5,101,END=316)CARD
316 WRITE(6,105)
317 WRITE(6,117)
117 FORMAT('01,10X,' YOUR REPLY WAS INCORRECT. MULTIPLE FUNCTIONS CAN'*/'
   BE USED IN AN EXPRESSION AS LONG AS THE EXPRESSION IS WELL FORMED'*/')
40 WRITE(6,118)
118 FORMAT('01,10X,' 2. IS THE FOLLOWING SEQUENCE OF PROGRAM STATEMENT
   $5 LEGAL '*/' REPLY: YES OR NO '*/5X,' LET B=-9/5X,' LET X=SQR(B)**1
   *0X,' **/10X,' **/16X,' **/}

50 READ(5,101)END=319,13CARO
IF(CARDP(1),EQ,ASTRKS)GO TO 50
119 FORMAT(*'NO. THE SQUARE ROOT OF A NEGATIVE NUMBER IS AN */
*UNDEFINED OPERATION.* IN THE NEXT LESSON YOU WILL SEE HOW */
*BASIC STATEMENT FOR TESTING AND BRANCHING TO ANOTHER SEGMENT */
*IF THE PROGRAM IF THE TEST IS TRUE. FOR EXAMPLE THE ABOVE */
*PROGRAM SEQUENCE MIGHT BE ALTERED AS FOLLOWS: */5X, LET B=-5*/5X
*/5, IF B LT 0 THEN X=SQR(X)*1/2X, ELSE */10X, */5
*/10X, */5, LET B=ABS(B)*5X, LET B=ABS(B)*5X
*/, GO TO 5C:*5X, */10X*/5 */ THIS PROGRAM SEQUENCE TESTS */
*FOR A NEGATIVE ARGUMENT. IF B IS */5 IT BRANCHED TO STATEMENT NUMB*/
*ER 100, MAKES THE ARGUMENT POSITIVE* */ AND BRANCHED BACK TO STATE*/
*MENT 5C TO COMPLETE THE PROGRAM */*/
READ(5,101)END=320,13CARO
320 WRITE(6,123)
123 FORMAT(*'1.0X,' 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 REPETITIVE 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 WAIT FOR SOME TEST CONDITION* */
READ(5,101)END=324,13CARO
324 WRITE(6,124)
124 FORMAT(*'1.6X,' THE FOLLOWING REVIEW PROBLEMS WILL EXERCISE */
*YOUR PROGRAMMING SKILLS TO DATE */1.0X,' 1. WRITE A PROGRAM TO COM*/
*PUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME NUMBER OF YEA*/
*RS HENCE. THE FORMULA IS */1.0X,' P=S*(1/(1+1)**N))/* WHERE P */IS THE PRESENT WORTH OF AN INVESTMENT HENCE** AT AN */*INTEREST RATE OF I. FOR DATA USE I=.08 S=5000 N=20 */5X, */2.1 WRITE A PROGRAM TO FIND SIDES A, B AND C OF A TRIANGLE USING*/
*THE LAWS OF SINES FORMULA: */1.0X,' A/SIN(A) = B/SIN(B) = 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 */1.0X,' 1 DEG=PI(3.1416/180) RADIANS. THE DATA FOR THE PROGR*/AM IS */1.5X,' 5C, 55.0, 55.2. ANGLE A=98.71 DEG. */5X, ANG*/LE B=49.97 DEG. */5X, ANGLE C=31.32 DEG. */7. IF YOU WISH TO RUN THE*/CALL CRUNCH(LNGTH) */IF(CARDP(1),EQ,ASTRKS)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(*'YOUR REPLY IS INCORRECT, REPLOY AGAIN */")
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* $1072.74*/5X,'2
*') SIDE A=45.06 IN. AND SIDE C=23.69 IN. */7/)
CALL EXIT
70 CALL TEST1
71 WRITE(6,127)
127 FORMAT(0,10X, * REPLY WITH ANSWER TO QUESTION 1. ) ?? */7/)
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.DECMAL. AND CARDP(6).EQ.DIGIT(8)) GO TO 90
80 WRITE(6,128)
128 FORMAT('0',10X, * THE CORRECT ANSWER IS $1075.74 AND THE PROGRAM*/
* SHOULD HAVE BEEN SIMILAR TO : */5X, ' HAVE READ S.I.N*/5X,' LET P=S*(1/
*(1+1)**(N))/5X, ' PRINT**PRESENT WORTH=**,P*/5X DATA 50G0,.08,20*/
*5X, ' END */7/)
90 WRITE(6,129)
129 FORMAT(0,10X, * REPLY WITH ANSWER TO SIDE A FOR QUESTION 2. ) */7/)
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.DECMAL. AND CARDP(4).EQ.DIGIT(11)) GO TO 97
95 WRITE(6,130)
130 FORMAT('0',10X, * THE CORRECT ANSWER IS SIDE A=45.06 IN. AND */
* SIDE C=23.69 IN. THE PROGRAM SHOULD HAVE BEEN SIMILAR TO : */5X,
* READ A=*.01*5X, * LET A1=B*SIN(A*(3.1417/180))/SIN(B*3.141/
*6/180))/5X, ' PRINT**SIDE A=**A1,**SIDE C=**C1*/5X, ' DATA 98.71,49.97,*/
*31.32,34.91*/5X, END */7/)
97 CALL EXIT
END

THIS LESSON INTRODUCES BRANCHING, BOTH CONDITIONAL AND
UNCONDITIONAL.

COMMON
- STACK(100), PRUG(250), CARD(80), CARDP(80), ALPHA(48),
- IAPTR, INPTR, IDATA(500), XDATA(150), STRING(5),
- DIGIT(10), IPRTB(10), LIST(100), LIST(100),
- PR(2500), NERRS, INST, NSTLST, DEBUG, ODLSON, QUOTE,
- PARLT, ASTRSK, BLANK
COMMON INTERP, IXERR
REAL*8 LESS*4/LESON*4/ CALL ALOAD(LESS*4,N1)
WRITE(6,100)
100 FORMAT(0,5X,*** LESSON 4 *** /*5X, LESS*4,N1)
WRITE(6,100)
*THIS INSTRUCTION SEQUENCE WILL COVER BRANCHES AS YOU HAVE /* LES00130
*SEEN FROM PREVIOUS PROGRAMS. A PROGRAM'S EXECUTION USUALLY /* LES00140
*TAKES A DIFFERENT ROUTE FROM THE FIRST TO THE LAST STATEMENT /* LES00150
*THE CONDITION THAT ALLOWS DETOURS TO OCCUR IN PROGRAMS IS CALLED /* LES00160
*BRANCHING. THERE ARE TWO TYPES OF BRANCHING: UNCONDITIONAL /* LES00170
*AND CONDITIONAL /*)
READ(5,101)END=3021CARD

101 FORMAT(80A1)
102 WRITE(6,102) /* THAT IS /* GO / 5X; /* AN UNCONDITIONAL BRANCH IS AN IMPERATIVE TRANSFER OF CONTROL /* LES02350
*FROM ONE POINT TO ANOTHER. THERE ARE TWO FORMS /* LES0240
*OF THE UNCONDITIONAL BRANCH: 'GO TO' AND THE 'COMPUTED' /* LES0250
*GO TO///IXX', I.GO TO '<< STATEMENT NUMBER >>' /* THIS COMMAND LES0250
*TRANSFERS PROGRAM CONTROL DIRECTLY TO THE STATEMENT NUMBER /* AND LES0270
*CONTINUES EXECUTION FROM THAT POINT. THE 'GO TO' IS USED /* FOR LES0280
*FORMING LOOPS IN A PROGRAM. /* /
READ(5,101)END=3031CARD

303 WRITE(6,103) /* A SAMPLE LOOP FOLLOWS: /// 5X; /* REM PROGRAM TO COLES0320
*PUT HERE YOUR WORTH/*5X; /* REM P=INVESTMENT, S=PRINCIPAL, I=INTEREST LES0330
*RATE, N=NR.YEARS/*5X; /* PRINT 'INVESTMENT', 'PRINCIPAL', 'INTEREST' LES0340
*/* **NR.YEARS /// 5X; /* READ S,N, P=17/(1+I)**N) /* LES0350
*6X; /* PRINT P,S,N, P, GO TO 10 /* DATA 5000, 0.02, 10; 5000, 0.06, 20; 5000, 0.08, 30; /* LES0360
*500, 0.06, 10, 5X; /* END /// PRODUCES THE OUTPUT /// / /* INVESTMENT, 10 LES0370
*X, /* PRINCIPAL, 10X; /* INTEREST, 10X; /* NR.YEARS /// 2X; /* 1072.74 /* LES0380
*5000, 0.06, 10, 2X; /* 1072.74, 1072.74, 1072.74, 1072.74, 1072.74 /* LES0390
*X, /* 1072.74 /* 2X, /* 1072.74 /* LES0390
*UO; /* YOU TRIED TO READ MORE NUMERIC DATA THAN YOU PUT IN /// **/ /* LES0410
READ(5,101)END=3041CARD

304 WRITE(6,104) /* THE ERROR OCCURS BECAUSE YOU RUN OUT OF DATA DURING /* LES0420
*/* THE EXECUTION OF THE LOOP SET-UP BY THE UNCONDITIONAL TRANSFER /* LES0430
*/* IF THE READ STATEMENT WERE NOT IN THE LOOP TO CAUSE THE /* LES0450
*PROGRAM TO STOP, THEN YOU WOULD BE IN AN 'INFINITE LOOP' /// /* LES0470
*A CONDITION IN WHICH THERE IS NO WAY TO STOP. YOU MUST ALWAYS /* LES0480
*CHECK FOR THE 'INFINITE LOOP' CONDITION BY MAKING SURE THAT /* LES0490
*YOUR PROGRAM HAS AN EXIT /// /* LES0500
READ(5,101)END=3051CARD

305 WRITE(6,105) /* ON '<< EXPRESSION >>' GO TO '<< STATEMENT NUMBER /* LES0520
*/* THIS SPECIAL FORM OF THE 'GO TO' /* LES0540
*COMMAND /// IS CALLED THE 'COMPUTED GO TO', THE EXPRESSION /* LES0550
*IN THE FORM OF THE STATEMENT MUST EVALUATE TO AN INTEGER /* LES0560
*BETWEEN 1-32769. IF IT IS NOT AN INTEGER, OR OUTSIDE THIS /* LES0570
*RANGE AN ERROR WILL OCCUR /// / 5X; /* WHEN THE 'COMPUTED GO TO' IS LES0580
*EXECUTED THE EXPRESSION IS EVALUATED /// AND PROGRAM CONTROL TRANSLES0590
*FERS TO THE N-TH STATEMENT NUMBER /// WHERE N-TH REPRESENTS THE LES0600
*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=306)CARD

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

307 WRITE(6,107)
107 FORMAT(0" CONDITIONAL BRANCHING */5X,* I
*THE CONDITIONAL BRANCH TRANSFERS PROGRAM CONTROL ONLY IF CERTAIN */1
*RELATIONS ARE TRUE. IF THE TEST OF RELATIONS IS TRUE THEN*/1.
*TRANSFER OF CONTROL OCCURS; OTHERWISE PROGRAM CONTROL CONTINUES*/1.
*WITH THE NEXT STATEMENT. THE FORM OF THE CONDITIONAL BRANCH IS*/1.
*/1. IF << EXPRESSION >> << RELATION >> << EXPRESSION >> THEN
<< STATEMENT NUMBER >> */1. NOTE THAT ALPHA VARIABLES ARE NOT ALLOWED*/1.
*/1. AS AN EXPRESSION IN A/* CONDITIONAL BRANCH. */1)
READ(5,101,END=306)CARD

308 WRITE(6,108)
108 FORMAT(0" THE RELATIONS ARE: */5X,* I
*SYMBOLS: %2X,' EXAMPLE *5X,' MEANING */1
* B, 5X, A GREATER THAN OR EQUAL TO B */2X, LT, 5X, A LT B*/1.
* A, LE, B, 5X, A LESS THAN OR EQUAL TO B, 5X*/1.
* B, 5X, LE, B, 5X, A NOT EQUAL TO B */3X, A = B*/1.
*/1. 6X, A EQUAL B //)
READ(5,101,END=309)CARD

309 WRITE(6,109)
109 FORMAT(0' WHEN THE CONDITIONAL STATEMENT IS EXECUTED /*5930
* THE EXPRESS iON RELATION EXPRESSION ) IS TESTED AND IF THE RELATION/*5940
* IS TRUE, THEN THE STATEMENT NUMBER $GO TO THE NEXT SEQUENTIAL /*5950
* OTHERWISE PROGRAM CONTROL CONTINUES TO THE NEXT SEQUENTIAL /*5960
* STATEMENT. FOR EXAMPLE: */5X,* I READ A*/5X,* I */5X,* /5X,*/1.
* IF, A = SQR(A)*/5X,* /5X,* */5X,* /5X,*/1.
* PRINT "ILLEGAL ARGUMENT"*/5X,* GO TO */5X,*/1. THE ONLY TLE*/5990
* THAT THE CONDITIONAL BRANCH IS EXECUTED IS WHEN /* A IS LESS*/1.
* MAN 0. */1.
READ(5,101,END=310)CARD

310 WRITE(6,110)
110 FORMAT(01, "YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT BRANCHING */1.
* NO,"215X,* 1 GO TO END */1.
5 READ(5,101,END=311)CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.0) GO TO 5
IF(CARDP(1).EQ.1) GO TO 10
311 WRITE(6,111)
111 FORMAT(*'YOUR ANSWER IS INCORRECT. THE ONLY THING THAT IS*'
!"ALLOWED AFTER A **GO TO** IS A STATEMENT NUMBER BETWEEN 1--9999')
10 WRITE(6,112)
112 FORMAT(*'15X, IF X LT A$ THEN 10*')
15 READ(5,101)END=313)CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.0)go to 15
IF(CARDP(1).EQ.1) GO TO 20
313 WRITE(6,113)
113 FORMAT(*'YOUR ANSWER IS INCORRECT. ALPHA VARIABLE MAY NOT*'
!BE USE FOR TESTING PURPOSES IN A CONDITIONAL STATEMENT **')
20 WRITE(6,114)
114 FORMAT(*'5X, IF(X**3-4) GE 10 THEN GO TO 100*'
!READ(5,101)END=313)CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.0)GO TO 25
IF(CARDP(1).EQ.1) GO TO 30
315 WRITE(6,115)
115 FORMAT(*'YOUR ANSWER IS INCORRECT. THE ONLY THING ALLOWED**'
!AFTER THEN IS A STATEMENT NUMBER **')
30 WRITE(6,116)
116 FORMAT(*'5X, LET Y=6,**15X,* ON Y GO TO 1,3,5,7,999**')
35 READ(5,101)END=317)CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.0)GO TO 35
IF(CARDP(1).EQ.1) GO TO 40
317 WRITE(6,117)
117 FORMAT(*'YOUR ANSWER IS INCORRECT. THERE ARE NOT ENOUGH****'
!STATEMENT NUMBERS IN THE **COMPUTED GO TO** **')
40 WRITE(6,118)
118 FORMAT(*'10X, 2,* CONSIDER THIS PROGRAM SEGMENT; ANY ERRORS (READ 1430
!EPLY: YES OR NO),**5X,* 1 READ A,B,**5X,* 3 LET Z=A*B**5X,* IF Z L**1440
!7 GO TO 999**5X,* **5X,* **5X,* **5X,* GO TO 3**5X,* DATA 3,5,7,**
!5X,**5X,* 99% END*')
READ(5,101)END=319)CARD
319 WRITE(6,119)
119 FORMAT(*'THERE IS AN **INFINITE** LOOP IN THIS PROGRAM ** IT''**
!COULD BE CORRECTED BY CHANGING THE GO TO 3, TO GO TO 1, **5X,*
!WHEN YOU WRITE PROGRAMS HAVING BRANCHES OR LOOPS YOU MUST ALWAY*
!** 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')
LES01090
LES01100
LES01110
LES01120
LES01130
LES01140
LES01150
LES01160
LES01170
LES01180
LES01190
LES01200
LES01210
LES01220
LES01230
LES01240
LES01250
LES01260
LES01270
LES01280
LES01290
LES01300
LES01310
LES01320
LES01330
LES01340
LES01350
LES01360
LES01370
LES01380
LES01390
LES01400
LES01410
LES01420
LES01430
LES01440
LES01450
LES01460
LES01470
LES01480
LES01490
LES01500
LES01510
LES01520
LES01530
LES01540
LES01550
LES01560
* LOOPS AND ARE AS FOLLOWS : ///

READ(5,101)END=320) CARD

320 WRITE(6,120)  LESO1590

120 FORMAT('0,i5',1*COUNT AND TEST METHOD, IN WHICH A COUNTER */
              IS INCREMEntED IN THE LOOP AND WHEN THE COUNTER REACHES A */
              CERTAIN VALUE IT BRANCH OUT OF THE LOOP *EXAMPLE PROGRAM */
              REM COUNT AND TEST METHOD, REM N IS COUNTER, INITIALIZED TO 0 */
              0, AND COUNTS FROM 1->10/10X, LET N=Z=0/10X, READ X/10X, 1.LES01640
* U LET Z=Z+X/10X, LET N=N+1/10X, IF N GT 10 THEN 100/10X, GOLES01650
* TO 10/10X, 100 PRINT*SUM=Z,100/10X, DATA 10/10X, END //// RELEASES01660

45 READ(5,101)END=50) CARD

CALL CRUNCH((LENGTH))

40 IF(CARDP(1).EQ.ASTRSK) GO TO 45

45 IF(CARDP(1).NE.DIGIT(1)) OR CARDP(2).NE.DIGIT(2)) GO TO 50

50 WRITE(6,121)  LESO1700

121 FORMAT('0',1*YOUR ANSWER IS INCORRECT - THE ONLY WAY TO BE */
              SURF OF VALUES IN A LOOP IS TO KEEP A TAB LE OF THE VARIABLES */
              AND KEEP TRACK OF THE VALUES IN THE LOOP : 5/X X 5/N 5X */
              4/X 10/X 5X 0/X 5X 5X 20/10X 13/X 5X LESO1770

122 FORMAT('0',1*READ AND TEST METHOD, IN WHICH A VALUE IS */
              READ IN AND TESTED FOR THE END OF LOOP CONDITION */
              REM READ AND TEST DEMONSTRATION, 1/X, 1/X IF 9999 THEN 9999,1/X */
              LET Z=100/10X, IF X=9999 THEN 9999,1/X LET Z=100/10X */
              * LE 10/X, 100/10X, PRINT*SUM=Z,100/10X, END /// REPLY WITH VALUE OF Z ///

65 READ(5,101)END=320) CARD

CALL CRUNCH((LENGTH))

60 IF(CARDP(1).EQ.ASTRSK) GO TO 65

65 IF(CARDP(1).EQ.DIGIT(1)) AND CARDP(2).EQ.DIGIT(2)) GO TO 70

123 FORMAT('0',1*YOUR ANSWER IS INCORRECT - VARIABLE VALUES IN THE */

70 WRITE(6,124)  LESO1950

124 FORMAT('0',1*SUMMARY, 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 CF LOOPS SO THAT */

70 WRITE(6,124)  LESO1950

124 FORMAT('0',1*CONTROL A LOOP BY THE INCREMENT AND TEST METHOD */
              THE FOLLOWING/* LESO2040
*G PROBLEMS WILL TEST YOUR NEW SKILLS :/**/
READ(5,101)END=325
325 WRITE(6,128)/*WRITE A PROGRAM TO COUNT THE NUMBERS**/
125 FORMAT(10I15,1X) /*WRITE BETWEEN 50 AND 60, AND ALSO PRINT THEM OUT. THE INPUT DATA**/LES02090
*IS 10,50,35,75,62,60,54,5*//*5X*,2*/*WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT**/
*FOR SOME YEARS HENCE AT VARYING INTEREST RATES THE FORMULA IS : P = S(1/(1+I)**N)**LES02120
*N**/*WHERE P = PRESENT WORTH, S = PRINCIPAL, I = INTEREST, AND N = NR. OF YEARS**/
*ARSS**/*FOR DATA USE S=5000, AND I = .04--->8, IN INCREMENTS OF .01***/
*AND N=20**/*YES ; OTH
ERWISE REPLY : NO**/*/
80 READ(5,101)END=326)CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 80
IF(CARDP(1).EQ.ALPHA(251))GO TO 85
326 CALL EXIT
85 CALL TEST1
WRITE(6,127)**I F 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(LNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 90
IF(CARDP(1).EQ.ALPHA(141)CALL EXIT
328 WRITE(6,128)/*FORMAT('G',IOX, 'PROBLEM 1.1/5X,' REM SORT AND COUNT PROBLEM/5XLES02330
*1')REM C.COUNTER, N=NUMBER/5X,PRINT NUMBERS IN RANGE 50-260**LES02540
*5X, IF N = 0 THEN 999 THEN 9999/5X,IFN=2350**LES02630
*N LE 60 THEN 1/5X, IF N LT 50 THEN 1/5X, SET C=C+1**/5X, PRINTLES02360
*T N/5X,GO TO 1/5X,/*DATA DC,50,35,75,56,62,60,54**5X,9999 PRLELES02370
*INT C,0,NUMBERS IN RANGE 5C-260**/5X, END**//10X, **PROBLEM 2.1**LES02380
*5X,'RECCOMPUTE PRESENT WORTH OF AN INVESTMENT/5X,**REM = PRESEN**LES02390
*T WORTH S=PRINCIPAL,N=NR.YEARS,I=VARIABLE INTEREST/5X,**PRINT**INTLES02430
**REST**/*5X,**PRINCIPAL**,5X,**YEARS**/5X,**READLES02410
* S=N/5X,LET I=1.0/5X,/*IF I GT .08 THEN 20C,,/5X, LET P=S*(1/I)**LES02420
* I+I)**N)/5X,**PRINT I,P,S,N/5X,LET I=I+G1*/5X,/*GO TO 1/5XLES02430
*5X,**DATA 5000,20/5X,20C END**/
CALL EXIT
END

THIS LESSON INTRODUCES ITERATION, SUBSCRIPTED VARIABLES, AND LISTS (VECTORS) AND TABLES (MATRICES)

COMMON
- STACK(100), PRG(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: **/LES00100
* A. ITERATION (LOOPING) **/LES00110
* IN THE LAST LESSON YOU WERE SHOWN HOW TO USE CONDITIONAL AND 
* UNCONDITIONAL BRANCHES TO CONTROL LOOPING. THE COUNT AND TEST **/LES00150
* ITERATIVE LOOP OCCURS SO FREQUENTLY THAT AN ABBREVIATED BASIC **/LES00160
* STATEMENT HAS BEEN DEVISED TO CONTROL LOOPING. THE ITERATIVE **/LES00170
* LOOP HAS THE FOLLOWING FORM: **/LES00180
* FER << SIMPLE VARIABLE >> = << EXPRESSION >> TO << EXPRESSION >> **/LES00190
* NEXT << SIMPLE VARIABLE >> **/**
101 READ (5,101) END=3C2 CARD
302 WRITE (6,102)
102 FORMAT (6,5x) FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT: **/5x:
* FOR I=1 TO 10 STEP 2/5X, LET X=X+1/5X, NEXT I **/5X: **/LES00200
* THE SIMPLE VARIABLE FOLLOWING "FOR" IS THE LOOP INDEX. WHEN **/LES00210
* THE FOR/NEXT PAIR IS EXECUTED, THE LOOP INDEX IS GIVEN THE **/LES00220
* VALUE (INITIALIZED) OF THE FIRST EXPRESSION (I=1 IN EXAMPLE). **/LES00230
* 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 **/LES00240
* "NEXT." OTHERWISE THE REMAINING STATEMENTS WITHIN THE LOOP **
* (FOR/NEXT) ARE EXECUTED SEQUENTIALLY UNTIL THE "NEXT" STATEMENT **/LES00250
* IS REACHED. **/**
303 WRITE (6,103) END=3C3 CARD
103 FORMAT (6,5x) WHEN THE "NEXT" STATEMENT IS REACHED, THE LOOP **/LES00300
* INDEX IS INCREASED/INCREMENTED) BY THE AMOUNT OF THE EXPRESSION /LES00310
* FOLLOWING "STEP", AND CONTROL IS TRANSFERRED BACK TO THE "FOR" **/LES00320
* STATEMENT WHERE THE LOOP CONTINUES UNTIL THE INDEX VALUE IS **/LES00330
* GREATER THAN THE FINAL VALUE. FOR EXAMPLE: **/5X: **/LES00340
* REM DEMO LOOP SUM THE NUMBERS 1 TO 5X, FOR I=1 TO 10 STEP 1/5X, LET C=I/5X, **/LES00350
* NEXT I/5X, PRINT "SUM=", SUM **/LES00360
* C**/**/5X: **/LES0420
304 FORMAT (6,104) END=3O4 CARD
104 FORMAT (6,5x) YOU WILL NOTICE THAT THE SIMPLE VARIABLE FOLLOWING**/LES0460
* **NEXT** IS THE SAME AS THE SIMPLE VARIABLE FOLLOWING** "FOR**; */5X* LES0050
* AND THAT THE** "NEXT" STATEMENT MARKS THE END OF THE LOOP */5X* LES00510
* BECAUSE THE INCREMENT VALUE OF A LOOP IS CCPITONELY ONE(1), THE */5X* LES00510
* **STEP** MODIFIER AND ITS EXPRESSION MAY BE OMITTED, AND THE */5X* LES00520
* INCREMENT VALUE WILL BE ASSUMED TO BE ONE(+1). FOR EXAMPLE THE */5X* LES00530
* ABOVE **"FOR** STATEMENT COULD BE WRITTEN : */5X* LES00540
* **FOR** I=1 TO 10*/5X*, THE INCREMENT VALUE AFTER**STEP** MAY BE */5X* LES00550
* **POSITIVE OR NEGATIVE ALLOWING THE FLEXIBILITY OF LOOPTING FORWARD */5X* LES00560
* OR BACKWARD. FOR A NEGATIVE**STEP** VALUE THE TEST BECOMES */5X* LES00570
* **LESS THAN** FOR EXAMPLE THE FOLLOWING*"**STATEMENTS ARE */5X* LES00580
* **EQUIVALENT : */5X*, FOR I=1 TO 10*/5X*, FOR I=10 TO 1 STEP -1 */5X* LES00590
* READ(I,101) END=3051C0D LES00590

305 WRITE(10,1) ANOTHER USEFUL TECHNIQUE OF LOOPING IS **NESTING**/* LES00600
**"** NESTING REFERS TO PLACING ONE LOOP INSIDE ANOTHER LOOP. **"** LES00630
**"** THE INNER LOOP **"** SPINS**"**AROUND AS MANY TIMES AS THE OUTER LOOP **"** LES00640
**"** IS INCREMENTED. FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT **"** LES00650
**"** FOR I=1 TO 10*/5X*, FOR J=1 TO 20 */5X*, **"** LES06600
**"** NEXT J, \&X, NEXT I /* LES06600
**"** REPEATED 10 TIMES, AND THE INNER LOOP(II) WOULD BE REPEATED 20 TIMES FOR EACH INC** LES06600
**"** TIMES FOR THE OUTSIDE LOOP, OR 200 REPETITIONS **"** LES06690
**"** LOOPS MAY BE NESTED UP TO A MAXIMUM OF 20; HOWEVER, THEY **"** LES06700
**"** CANNOT OVERLAP THE INNERMOST LOOP MUST BE CLOSED WITH ITS **"** LES06710
**"** **"** STATEMENT BEFORE ENCOUNTERING THE NEXT OUTER LOOP**"* **"** LES06720
**"** **"** STATEMENT FOR EXAMPLE : **"*/5X*, LES00730
**"** FOR X=1 TO 100, STEP 1*/5X*, FOR Y=2 TO 50*/5X*, FOR Z=3 TO 50*/5X*, **"** LES06740
**"** NEXT Z, NEXT Y, NEXT X, **"** LES06750
* READ(I,101) END=3061CARD LES00760

306 WRITE(10,2) **"** FOR/*/5X*, **"** LES00770
**"** WITHIN A FOR/NEXT LOOP CONDITIONAL AND UNCONDITIONAL **"** LES00780
**"** **"** NAL**"** BRANCHES MAY BE USED TO TRANSFER CONTROL OUT OF A LOOP**"** LES00790
**"** OR WITHIN LIMITS OF THE SAME LOOP. HOWEVER IT IS NOT POSSIBLE**"** LES00800
**"** TO BRANCH INTO THE MIDDLE OF A FOR/NEXT LOOP. BECAUSE LOGIC **"** LES00810
**"** PROBLEMS OCCUR AND AN ERROR WILL RESULT. AN ADDITIONAL ITEM**"** LES00820
**"** TO BE CAREFUL ABOUT IS USING THE INDEX VARIABLE OF THE FOR/NEX**** LES00830
**"** LOOP IN COMPUTATIONS. IF YOU ALTER THE VALUE OF THE LOOP INDEX**"** LES00840
**"** **"** YOU WILL EFFECT THE ACTION OF THE LOOP. FOR EXAMPLE : **"*/5X*, LES00850
**"** FOR I=1 TO 50*/5X*, LET I=I+10*/5X*, NEXT I */5X*, **"** PRINT I */5X*, **"** LES00860
D**"** **"** WITH THE VALUE OF I THAT IS PRINTED **""** LES00870
* READ(I,101) END=3071CARD LES00880
CALL CRUNCH(1LGTH, **""** LES00890
IF CARDP(1)=EQ.DIGIT(2).AND.CARDP(2)=EQ.DIGIT(2)) GO TO 5
IF CARDP(1)=EQ.DIGIT(2) GO TO 10
307 WRITE(6,107) LES00920
10 WRITE(6,108) **"** LES00960
108 FORMAT('I0') YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT WHAT YOU HLE500970
*VE JUST LEARNED */7] HLE500980
WRITE(6,109) HLE500990

169 FORMAT('I0',16X,'1') SAMPLE PROGRAM */15X;' LET S=0'/15X;' FOR K=5/5LH501000
* TO S'/15X;' LET S=S+K'/15X;' NEXT K'/15X;' PRINT 'SUM='1';S'/15X;' HLE501010
* END'/ REPLY WITH VALUE OF S '/7] HLE501020
15 READ(5,101),END=310)CARD
CALL CRUNCH(LNGTH)
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) HLE501030

110 FORMAT('C') YOUR ANSWER IS INCORRECT * THE VALUES OF K AND S */15X;
* IN THE LOOP ARE: 5'/5X;' K'/5X;' S'/5X;' 5'/5X;'-5'/5X;'-4'/5X;'-5'/5X;'-4'/5X;'-5'/5X;'=0'/5X;' HLE501040
* 55'/5X;' 1'=5'/5X;'-1'=5'/5X;'-1'=5'/5X;'-5'/5X;'-0'/5X;' HLE501100
155'/5X;' 1'=5'/5X;'-1'=5'/5X;'-1'=5'/5X;'-5'/5X;' HLE501120
* 5'/5X;' S'/5X;' 0'/5X;' 6'/5X;' S'/0'/15X;' HLE501100
WRITE(6,111) HLE501130

111 FORMAT('C',16X,'2') SAMPLE PROGRAM */15X;' FOR I=100 TO 1 STEP 2 I=15X;
* 15X;' LET Z=I**2'/15X;' LET Y=2+10'/15X;' NEXT I'/15X;' PRINT I*/15X;' HLE501140
* 15X;' END'/ REPLY WITH VALUE OF I'/7] HLE501160
25 READ(5,110),END=312)CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 25
IF(CARDP(1).EQ.DIGIT(2).AND.CARDP(3).EQ.DIGIT(1)) GO TO 30
WRITE(6,112) HLE501170

312 FORMAT('O') YOUR ANSWER IS INCORRECT * THE INDEX VALUE OF THE */15X;
* LOOP 'I' GOO IS GREATER THAN THE FINAL VALUE('I') IN THE FIRST */15X;
* TEST * THIS IS I=100 * IF THE LOOPS WERE DECREMENTED IN STEPS OF -2 */15X;
* THEN THE LOOP WOULD BE EXECUTED AND THEN I=0 */15X;' HLE501220
WRITE(6,113) HLE501230

113 FORMAT('C',16X,'3') SAMPLE PROGRAM */15X;' FOR I=5
* TO 1 STEP -1'/15X;' FOR J=I TO 5'/15X;' IF I=J THEN 10'/15X;' GO */15X;
* TO 15'/15X;' 10 LET T=T+1'/15X;' NEXT I'/15X;' NEXT J'/15X;' PRINT I*/15X;' HLE501250
* NEXT J'/15X;' ENC'/ REPLY WITH VALUE OF T */15X;' HLE501300
35 READ(5,110),END=314)CARD
CALL CRUNCH(LNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 35
IF(CARDP(1).EQ.DIGIT(1).AND.CARDP(2).EQ.BLANK) GO TO 40
WRITE(6,114) HLE501310

314 FORMAT('O') YOUR ANSWER IS INCORRECT * THE OUTSIDE LOOP('I') */15X;
* IS REPEATED 5 TIMES('I'=5-->1) * AND FOR EACH VALUE OF I * THE */15X;
* INNER LOOP('J') IS REPEATED 5 TIMES('J'=1-->5) * THEREFORE THE */15X;
* THE CONDITIONAL BRANCH IS EXECUTED FIVE TIMES [:5'/15X;' I=5'; J=5'/5LH501370
* X*/5X;' J=4'/5X;' I=3'/5X;' J=2'/5X;' HLE501390
* 1'=1'; J=1'/7] * THUS T=5 */15X;' HLE501410
WRITE(6,115) HLE501420

315 FORMAT('O',16X,'8') SUBSCRIPTED VARIABLES/LISTS AND TABLES */5X;' HLE501430
* 1. IN LESSON 1 YOU LEARNED THAT SIMPLE VARIABLES WERE A LETTER */15X;' HLE501440
*ZERO IN EACH ELEMENT OF THE LIST YOU ARE ASSURED THAT THE LIST*/
*IS CLEANED UP BEFORE YOU USE IT.*/  
READ(5,101),END=321CARD  
LES01930  
321 WRITE(6,121)  
121 FORMAT('C5X, 5X, 3. IN ADDITION TO LISTS(VECTORS), BASIC ALLOWS*/
*YOU THE ABILITY TO USE TABLES, OR MATRICES, OR TWO DIMENSIONAL* LES01970  
** ARRAYS, * THE SUBSCRIPTS OF A TABLE REPRESENT THE ROWS AND *  
LES01980  
** COLUMNS OF THE TABLE, * THE FIRST SUBSCRIPT IS THE ROW AND THE */
** SECOND SUBSCRIPT IS THE COLUMN, * FOR EXAMPLE IN TABLE**D**':I'/5X,*  
LES01990  
D(3,4) REFERS TO THE VALUE OF THE ELEMENT IN ROW 3, COLUMN 4*  
LES0204C  
*OF TABLE**D**'///  
READ(5,101),END=3221CARD  
LES02090  
322 WRITE(6,122)  
122 FORMAT('C10*, AS WITH A LIST, ANY ELEMENT OF A TABLE MAY BE */
*REFERENCED BY DEFINING THE PAIR OF SUBSCRIPTS AS DESIRED *  
LES02050  
*RCK=COLUMN ORDER = IN A 3X3 TABLE(MATRIX) THE TABLE IS REFERENCED* LES02600  
** AS FOLLOWS : ///5X, 1,1,5X, 1,2,5X, 1,3,5X, 2,1,5X, 2,2,5X, 2,3///  
LES02700  
X, 2,3///5X, 3,1,5X, 3,2,5X, 3,3/// WHERE THE FIRST SUBSCLESO2800  
**RIPT IDENTIFIES THE ROW AND THE SECOND THE COLUMN *  
LES02900  
**USING THIS REFERENCE SYSTEM CONSIDER HOW TO FILL THE FOLLOWING TABLES  
LE ///5X, 1,1,5X, 2,1,5X, 3,3///  
LES02110  
* 4, 5X, 9/// END ///  
LES02120  
READ(5,101),END=3231CARD  
LES02130  
323 WRITE(6,123)  
123 FORMAT('C10*, THIS TABLE**X** COULD BE FILLED AS FOLLOWS : ///5X,*  
*DIMX(3,3)/6X, FOR I = 1 TO 3, FOR J = 1 to 3, X,J, READ XI,J, I ///  
*5X, 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=3241CARD  
CALL CRUNCH(11NTH)  
IF(CARDP(1),EQ.'ASTPSK') GO TO 55  
IF(CARDP(1),EQ.'DIGIT(17)') GO TO 60  
324 WRITE(6,124)  
124 FORMAT('C10*, 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 ///  
60 WRITE(6,125)  
125 FORMAT('C10*, NOW WITH TABLE**X** ASSIGNED VALUES CONSIDER THIS/**  
PROGRAM • ASSUMING TABLE**X** HAS BEEN FILLED BY THE ABOVE PROGRAM**LES02290  
**///5X, LET S = 0,5X, FOR I = 1 TO 3, 5X, FOR J = 1 TO 3, 5X, IF I = 00**LES02320  
**E J THEN 5/5X, LET S = S,X(1,1)/5X, 5 NEXT J/5X, NEXT I///5X,**  
**PRINT**SUM = S///5X, END /// REPLY WITH VALUE OF S///**  
65 READ(5,101),END=3261CARD  
CALL CRUNCH(11NTH)  
IF(CARDP(1),EQ.'ASTPSK') GO TO 65  
IF(CARDP(1),EQ.'DIGIT(2)') AND CARDP(2),EQ.'DIGIT(6)') GO TO 70  
326 WRITE(6,126)  
126 FORMAT('C10*, YOUR ANSWER IS INCORRECT. THIS PROGRAM SUMS THE ** LES02380
CALL CRUNCH(ILNGTH)

IF(CARDP(1).EQ.ASTRSK) GO TO 85
IF(CARDP(1).EQ.ALPHA(14)) GO TO 90

333 WRITE(6,133)
133 FORMAT('0', 'THE STATEMENT IS INCORRECT. (5,5,5) IS ILLEGAL', '/')
*ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.'/)

90 WRITE(6,134)
134 FORMAT('0', 'D. SUMMARY', 'IN THIS LESSON YOU HAVE LEARNED HOW TO USE THE FOR/NEXT STATEMENT', 'AND HOW TO MANIPULATE LISTS, VECTORS, AND TABLES, MATRICES.', 'YOU NOW KNOW ALL THE TOOLS TO BEGIN WRITING SOPHISTICATED PROGRAMS.', 'AND AS YOU WRITE MORE COMPLICATED PROGRAMS, YOU WILL FIND A NEED FOR SUBROUTINES. SUBROUTINES ARE SIMPLY 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 PROGRAM.', 'THE GOSUB STATEMENT ALLOWS SUBROUTINES IN BASIC, AND YOU WILL BE INTRODUCED TO IT IN THE NEXT LESSON.')

READ(5,1G1) END=335 /CARD

335 WRITE(6,135)
135 FORMAT('0', 'YOU WILL NOW BE GIVEN TWO OPTIONAL PROGRAMMING PROBLEMS', 'PROBLEMS TO EXERCISE YOUR NEW TOOLS.', 'REVERSE THE NUMBERS IN A 10-ELEMENT LIST.', '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 THE CHANGES.', 'WRITE A PROGRAM TO ARRANGE THE FOLLOWING LIST IN DECREASING ORDER: 10, 30, 5, 15, 40. ONE METHOD OF APPROACHING THIS PROBLEM IS TO CHECK THE FIRST ELEMENT OF THE LIST AGAINST THE SECOND ELEMENT.', '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 ORDERED.', 'ST IS IN ORDER.')

READ(5,1O1) END=336 /CARD

336 WRITE(6,136)
136 FORMAT('0', 'IF YOU WANT TO EXECUTE THESE PROBLEMS REPLY: YES'), ELSE NO', '/')

95 READ(5,1O1) END=337 /CARD

CALL CRUNCH(ILNGTH)

IF(CARDP(1).EQ.ASTRSK) GO TO 95
IF(CARDP(1).EQ.ALPHA(15)) CALL TEST1

337 WRITE(6,137)
137 FORMAT('0', 'IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEMS', 'REPLY: YES', 'OTHERWISE NO', '/')

96 READ(5,1O1) END=338

CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 96
IF(CARDP(1).EQ.APHA(14)) CALL EXIT

338 WRITE(6,138) LESO3340
138 FORMAT(10*','PROBLEM 1.),'//5X,' REM PROGRAM TO REVERSE ELEMENTS IN
*     A LIST.//5X,' REM Y IS A TEMPORARY LOCATION FOR THE ELEMENT BEING
*     SWAPPED//5X,' DIM X(10)//5X,' FOR I=1 TO 5//5X,' LET Y=X(I)//5X,' LESO3390
*     LET X(I)=X(11-I) //5X,' LET X(I)=Y//5X,' NEXT I//5X,' END.//' P
*     PROBLEM 2.),'//5X,' REM SORT LIST IN DECREASING ORDER//5X,' REM S; ILESO3410
*     S THE EXCHANGE COUNTER,' Y IS A TEMPORARY LOCATION//5X,' DIM D(5)LESO3420
*     //5X,' FOR I=1 TO 5//5X,' READ D(I)//5X,' NEXT I//5X,' LET S=L'//5X
*     FOR I=1 TO N-1//6X,' IF D(I) GE D(I+1) THEN 16//5X,' LET Y=D(I)LESO3440
*     //5X,' LET D(I)=D(I+1) //5X,' LET D(I+1)=Y//6X,' LET S=S'//6X
*     //5X,' IF S # NE C THEN 5//5X,' PRINT NUMBERS IN ORDER//5X,' FOLESO3460
*     IF I=1 TO 5//5X,' PRINT D(I)//5X,' NEXT I//5X,' END.//' P
CALL EXIT
END

THIS LESSON INTRODUCES SUBROUTINES AND RECURSION

COMMON
- STACK(100), PROG(200), CARD(60), CARDP(80), ALPHA(48),
- IAPTR, IMPTR, IADATA(500), Xndata(500), STRING(5), LESO260
- Digits(11), PRTB(10), LISTS(100), NLISTS(100),
- PRT(250), NERRS, INS, NSTLST, DEBUG, DOLGSN, QUOTE,
- EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
- PARLFT, ASTRSK, BLANK
CCMCN INTERP,EXERR
REAL*8 LES6,LES6N
CALL ALADD(LES6,NI)
WRITE(6,10C) LES0010
100 FORMAT(10*I,'*** LESSON 6 ***')
** THIS LESSON WILL INTRODUCE YOU TO SUBROUTINES AND THE PROGRAMMING
** TECHNIQUE CALLED RECURSION. */5X,' A SUBROUTINE */5X,' IN
** MANY PROGRAMS A BLOCK OF STATEMENTS MAY BE NEEDED ON MORE */5X,' THAN
** OCCASIONS, A TECHNIQUE TO ECONOMIZE ON CODING INSTRUCTIONS */5X
** CALLED A SUBROUTINE IS PRESENTED. */5X,' THE FORM OF THE SUBROUTINE LESO170
** IS */5X,' GOSUB //>5X,' STATEMENT NUMBER >>5X,' ***//5X,' ***5X
** ***//5X,' RETURN */5X,' LESO190
** READ(5,101) END=3C21CARD
101 FORMAT(10A1)
LES0220
302 WRITE(6,102) LES0220
102 FORMAT(10*A1,'*** EXECUTION OF THE GOSUB STATEMENT CAUSES THE '//' LESO230
** COMPUTER TO TRANSFER CONTROL TO THE STATEMENT NUMBER AFTER GOSUB*/5X,' LESO240
** **' WHEN CONTROL IS TRANSFERRED, STATEMENTS ARE EXECUTED SEQUENTIALLY,
** IALLY *** UNTIL A RETURN IS ENCOUNTERED. AT THAT TIME, CONTROLS LESO260
** L*** RETURN TO THE NEXT STATEMENT FOLLOWING THE GOSUB ** LESO270
** STATEMENT WHICH CALLED THE SUBROUTINE. FOR EXAMPLE CONSIDER */5X,' LESO280
*READ N'/5X,* IF N GT 0 THEN 20*/5X,* GO TO 50*/3X,* 20 PRINT N'/5X *
* GOSUB 100*/5X,* PRINT F'/5X,* GO TO 1G*/8X,* 100 REM ITERATIVE *
* SOLUTION */13X,* LET F=F*/13X,* FOR I=1 TO N'/14X,* LET F=F*/13X,* LES00790 *
* NEXT I/*10X,* REM RETURN FACTORIAL(N)*/8X,* RETURN*/3X,* 50 REM *
* FACTORIAL, F(N)= IF N=0 THEN 1/*5X,* REM OTHERWISE, F(N)= LES00810 *
* N*/(N-1)/5X,* RESTORE*/5X,* PRINT 'RECURSIVE SOLUTION */7X,* 6U R *
* READ N'/5X,* IF N GT 0 THEN 70*/5X,* GO TO 9999*/5X,* 7G PRINT N'/5 *
* X,* GO SUB 200*/5X,* PRINT F'/6X,* GO TO 50*/8X,* 70 REM RECURSIVE *
* SOLUTION */13X,* IF N GT C THEN 210*/13X,* LET F=F*/13X,* RETURN*/8LES00850 *
* X,* 210 LET N=N'-1*/13X,* REM RECURSIVE CALL */13X,* GOSUB 200*/13X,* LES00860 *
* LET N=N'/1*13X,* LET F=F*/N*/8X,* RETURN*/5X,* DATA 2,5,8,6,-1*9 LES00870 *
* 999 END */7X,* LES00880 READ(5,101,END=308) CARD 308 WRITE(6,108) *
* IN THE ITERATIVE METHOD THE FACTORIAL FUNCTION */7X,* LES00910 *
* N*/(N-1)/5X,* IF N=0 THEN 1/*5X,* REM OTHERWISE, F(N)= LES00920 *
* RECURSIVE SOLUTION THE FACTORIAL FUNCTION, F(N), IS DEFINED */7X,* LES00930 *
* IN TERMS OF ITS FINAL VALUES WHEN N=0, F(N)=1; OTHERWISE */7X,* LES00940 *
* F(N)=N*/(N-1)*/5X,* REM KEEP TRACK OF WHERE THE SUBROUTINE CALLS RETURN IN RECURSION */7X,* LES00950 *
* IT HELPS TO VISUALIZE A LAST-IN-FIRST-OUT (LIFO) STACK CONTAININ *
* G */ THE ADDRESS OF THE GOSUB STATEMENTS EVERY TIME A GOSUB */7X,* LES00960 *
* IS ENCAPTURED, PUT ITS ADDRESS ON TOP OF THE STACK (PUSHING */7X,* LES00970 *
* DOWN ANYTHING PREVIOUSLY ON THE STACK). THEN EVERY TIME A RETURN */7X,* LES01000 *
* IS ENCAPTURED, RETURN TO THE TOP ADDRESS ON THE STACK (AND */7X,* LES01010 *
* POP UP THE NEXT ADDRESS ON THE STACK).* LES01020 READ(5,101,END=309) CARD 309 WRITE(6,104) *
* TO FULLY UNDERSTAND THE CONCEPT OF RECURSION YOU */7X,* LES01050 *
* SHOULD STEP THROUGH THE FACTORIAL PROBLEM BY HAND USING THE */7X,* LES01060 *
* HELP OF THE STACK TO SEE HOW RECURSION WORKS. IF YOU UNDERSTAND */7X,* LES01070 *
* THE CONCEPT OF RECURSION YOU ARE READY TO SOLVE THIS PROBLEM. */7X,* LES01080 *
* 5X,* LET X=X'/1*13X,* LET X'/5X,* GO TO 9999*/8X,* 5G LES01090 *
* CU LET X=X'/1*13X,* IF X GT 3 THEN 150*/13X,* GOSUB 100*/8X,* 15G LES01095 *
* LET X=X'/1*8X,* RETURN*/9999 END */7X,* LES01100 IF CARDP(1).EQ. BLANK GO TO 10 *
* CALL CRUNCH(LNGTH) *
* IFCARDP(1).EQ. STR5X GO TO 5 *
* IFCARDP(1).EQ. DIG13X AND CARDP(2).EQ. BLANK GO TO 10 5 READ(5,101,END=301) CARD 310 WRITE(6,110) *
* YOUR ANSWER IS INCORRECT. CONSIDER THE FOLLOWING. */7X,* LES01170 *
* TABLE OF VALUES FOR X AND ITEMS IN THE STACK11-FIRST GOSUB 2-SELES01180 *
* COND GOSUB */6X,* X=1*10X,* STACK/5X,1/,5X,* X=2*10X,* STACK/5X,1/,5X,* LES01190 *
* X=2*12X,* STACK/21X,2/,7X,* X=7*12X,* STACK/21X,2/,7X,* 21X,1/,5X,* 10X,* STACK/21X,2/,21X,1/,11*7X,* X=6* LES01210 *
* X=2*21X,* STACK/721X,2/,21X,1/,11*7X,* 75X,* X=6* LES01220 *
* EXAMPLE UNTIL YOU UNDERSTAND THE RECURSION TECHNIQUE QUESTION. */7X,* LES01230 *
* KNOWING HOW TO USE RECURSION IS NOT A REQUIREMENT FOR KNOWING */7X,* LES01240
* HOW TO USE BASIC, IT IS ONLY A CLASSIC PROGRAMMING TECHNIQUE */ LES01250
*)

10 WRITE(6,111)  
111 FORMAT('5X,'C* SUMMARY')  
112 FORMAT('5X,' THE FOLLOWING PROBLEMS WILL TEST YOUR LATEST SKILLS')  
*7/ OF BASIC STATEMENTS * THEY SAVE BOTH PROGRAMMER TIME AND */ LES01120
*COMPUTER STORAGE SPACE, THE GOSUB/RETURN COMMAND AND THE OTHER* LES01300
* 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

312 WRITE(6,112)  
313 WRITE(6,113)  
314 WRITE(6,114)  

* IF YOU WANT TO EXECUTE THESE PROGRAMS NOW, THEN */
* REPLY: YES, OTHERWISE NO */

20 READ(5,101)END=3151CARD  
25 READ(5,101)END=3151CARD

CALL CRUNCH(LNTH)  
CALL TEST1  

WRITE(6,114)  

* IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEMS*/
* REPLY: YES */

25 READ(5,101)END=3151CARD  
315 WRITE(6,115)  

115 FORMA10T('10X,' PROBLEM 1 /*5X,' REM ES=EMPLYEEE, S=GROSS SALARY, R=RETIRE*/ LES01670
* REMENT, 5X,' REM T=TAX, P=NET PAY /*5X,' READ N /*5X,' FOR I=1 TO N */ LES01690
**/9X,' REM E,S,/, 5X,' GO SUB 20 /*9X,' GO SUB 30 /*9X,' P=S-R-T /*9X,' LES01700
**/9X,' REM E /*9X,' GO TO 9999 /*9X,' REM TAX CAL /*
* CULATION /*10X,' IF S 'LT 600 THEN 40 /*13X,' LET T=0.03 /*S /*13X,' GO
*TO 50\(^{0}/9\)X, *40 LET T=5\(^{0}/9\)X, *50 RETURN;*/9\)X, *30 REM RETIREMENT CALCULLESQ1730
*ATION*13X, *IF S GE 2500 THEN 60\^{0}/13X, *IF S GE 800 THEN 70\^{0}/13X, LESO1740
* LET R=S/13X, * GO TO 80\^{0}/9\)X, *60 LET R=20\^{0}/13X, * GO TO 80\^{0}/9\)X, *70LESO1750
* LET R=10\^{0}/9\)X, *80 RETURN */5\)X, * DATA **JONES** */5\)X
* DATA **SMITH** */5\)X, * DATA **DALE** */5\)X, * DATA **BERRY** */5\)X
*Y=10500*/5\)X, * PRINT SUM FROM X=--Y*/6\)X, * PRINT SUM FROM X=--Y*/6\)X
* GET=13X, *10 REM SUM NUMBERS RECURSIVELY */13X, LESO1810
* * IF X NE Y THEN 20\^{0}/13X, * RETURN */9\)X, *20 LET S=X*13X, LESO1820
* *13X, *30 SUB 10\^{0}/13X, * LET Y=S+13X, LESO1830
* *X=DATA 3, 6*/5\)X, * DATA 1, 10 */5\)X, * DATA 9999, 1*/9\)X, * RETURN */5\)X
CALL EXIT
END

THIS LESSON GIVES A COMPLETE SUMMARY OF BASIC LANGUAGE STATEMENTS

COMMON
- STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
- INPTR, INPRT, IADATA(500), XNDATA(500), STRING(5),
- DIGIT(10), PRITB(10), LISTST(100), ISTST(100),
- PRT(2500), NERRS, INST, NSLLST, DEBUG, DLSGN, QUOTE,
- EQUALS, PARRT, DECIMAL, PLUS, MINUS, SLASH, COMMA,
- PARLFT, ASTRK, BLANK

COMMON
- interp, err
- real*8 les7*lesn

CALL LOAD(LE57*N1)

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

FORMAT(10*, 5X, *) ** NUMBERS : (LIMITED TO 9 DIGITS) */5X, *
* A) INTEGRALS : DIGITS WITH NO FRACTIONAL PART, EG. 5, 7, 10 */5X, *
* B) REALS : DIGIT WITH A FRACTIONAL PART, EG. 5.0, 7.31, 16.0 */5X, *
* A NUMBER MAY BE PRECEDED BY A SIGN (+, ), BUT IS ASSUMED TO */5X, *
* BE POSITIVE IF NONE IS GIVEN */5X, *
* C) VARIABLES */5X, *
* A) SIMPLE VARIABLES : A SINGLE LETTER, OR A SINGLE LETTER */5X, *
*FOLLOWED BY A DIGIT:  EG. A1, B6, Z0

*B) ALPHABET: A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN ($)
*C) SUBSCRIPTED VARIABLE: [SINGLE LETTER]
*#) (SINGLE LETTER)
*#) SINGLE SUBSCRIPT: <<LETTER>> {<<EXPRESSION>>}
*#) DOUBLE SUBSCRIPT: <<LETTER>> {<<EXPRESSION>>}

READ(5,101,END=303)CARD

303 WRITE(6,103) 5. OPERATORS: (LISTED IN DECENDING HIERARCHY)

A) EXPONENTIATION **
B) MULTIPLICATION *
C) DIVISION /
D) ADDITION +
E) SUBTRACTION -
F) EXPRESSIONS :

G) SINGLE NUMBER OR VARIABLE:  EG. 10, -52, D, X(I,J)
H) BUILT IN FUNCTION:  EG. SQRT(10), TAN(75)
I) ARITHMETIC EXPRESSION:  EXPRESSIONS SEPARATED BY OPERATORS
J) AND GROUPED BY PARENTHESIS:  EG. 5+6, A**2(3*X-4)

READ(5,101,END=304)CARD

304 WRITE(6,104)

104 FORMAT(10)* 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

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

READ(5,101,END=305)CARD

305 WRITE(6,105)

105 FORMAT(10)* 4. LET << VARIABLE >> = << EXPRESSION >>

A) PRINT <<  *STRING* OR EXPRESSION......*STRING* OR EXPRESSION
* >> /5X*, SIMPLY PRINT
B) DATA <<  *STRING* OR NUMBER ...... *STRING* OR NUMBER
C) RESTORE OR RESTORE
D) IF << EXPRESSION >> << RELATION >> THEN << STATEMENT NUMBER >>
E) IF << EXPRESSION >> GO TO << STATEMENT NUMBER >>
F) GO TO << STATEMENT NUMBER >>
G) FOR << SIMPLE VARIABLE >> = << EXPRESSION >> TO << EXPRESSION >>
H) NEXT << SIMPLE VARIABLE >> = << EXPRESSION >>
I) DIM << LETTER >> ( << INTEGER EXPRESSION >> ) OR
J) DIM << LETTER >> ( << INTEGER EXPRESSION >> ) OR...
* 100:13. GOSUB << STATEMENT NUMBER >> 100 5X,*
* 5X:* 5X: 5X: 5X: 5X: 5X: 5X: \** RETURN \*/}
READ(5,101,END=306) CARD
366 WRITE(6,106)
106 FORMAT(''' 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=U SHOULD GET THE BASIC MANUAL*/*
*TECHNICAL NOTE NR. 3211-12 IN ROOM I-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*/*
*CARGAGE OUT )''*/)
CALL EXIT
END
END

LES02030
13.13.46 START
EXECUTION BEGINS...

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

1. WHEN ASKED FOR A RESPONSE, TYPE IN THE CORRECT REPLY AND
HIT THE CARTRIDGE 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 CARTRIDGE 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 CARTRIDGE RETURN, THEN HIT ATTN KEY AND TYPE LOGOUT.

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

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

CAIL-BASIC 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

70
WRITE PROGRAMS IN A MINIMAL AMOUNT OF TIME.

THE REFERENCE TEXTS RECOMMENDED FOR CAS BASIC ARE:

1. BASIC LANGUAGE MANUAL, TN # 0211-12 APRIL 1971
   (FREE UPON REQUEST IN 1-147)

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 LOOPSING 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 TELETYPING 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 USED   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 ASSIGN NUMBERS IN THE DATA STATEMENT TO THE VARIABLES M AND G
- LET ASSIGNED 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 PRECEDING 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)??
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.) ISLET=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.

C. 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'

GO
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, 9Z, ABC, AND XYZ 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$, X$, Z$ 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, IE. 3, 15
2. REALS

REAL (FIXED-POINT) NUMBERS HAVE A DECIMAL POINT AND A FRACTIONAL PART, I.E. 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, *6, X+Z, 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, I.E. 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.) \( 4 + 6 / 2 \) has the value (reply with value)

Your answer is incorrect. The expression is evaluated as follows:

\[ 4 + 6 / 2 \rightarrow 4 + 3 \rightarrow 7 \]

2.) \( (4 + 6) / 2 \) has the value (reply with value)

3.) \( ((4 + 6) / 2)^2 \) has the value (reply: value)

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, SY
3 LET A1=2*(A1+Z1)
4 LET B2=((A1+3)+Z1)**2
5 LET Z1=3.141592763
6 PRINT IA,B2,Z1
7 DATA 5.0,3,3.1416,'FINI'

After looking at the sample problem, you will be asked questions about each statement.

3.) 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 \( a1 = \text{expression} \). 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? (reply: yes or no)

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.

80

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.

80

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

80

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', 'STRING' >>

MULTIPLE ELEMENTS IN THE PRINT LIST ARE SEPARATED BY COMMAS.

80

1. 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, B1, S**2

ASSUMING THAT A=10.0 , B=13.5 WOULD PRINT :

1 10.0 13.5 25

2. PRINT «STRING» WILL PRINT OUT ALL THE
ALPHA-NUMERIC CHARACTERS OF THE STRING WITHIN 'SINGLET' 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 ZONES
EACH IS COLUMNS WIDE . PRINT ZONES CAN BE SKIPPED BY PUTTING A
BLANK IN THE PRINT LIST . FOR EXAMPLE :

PRINT A, Ü, ,X

OUTPUTS THE VALUE OF A IN THE FIRST ZONE , Ü 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'S SQUARED »1, S**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 SQUARED MINUS % TIMES 2 TIMES 2.

2) FIND THE VALUE OF 3.1416(B CUBED)/12 WHERE B=2.50, H=3.03.

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

YES

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 CAYKASIC 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', 19

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

DURING EXECUTION OF READ STATEMENTS, AS THE
VARIABLES IN THE READ LIST ARE ASSIGNED VALUES FROM THE TOP OF
THE APPROPRIATE DATA STACK, THE DATA STACK IS DECREMENTED AND
THE NEXT ELEMENT POPS UP.

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

E. RESTORE, RESTORES

THE RESTORE, RESTORES STATEMENTS ARE USED TO RETURN THE NUMERIC
AND ALPHA-NUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO
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', 'WRONG', 0.0
READ A, B, $, C, D
READ J$, E$, X
```

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

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

```
RESTORE
READ Z$, A
```

2.) What is the value of Z$? Reply with value.

The correct answer is Z$=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:

```
Z$ <-- 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:

```
READ A,B
PRINT'ANS=',(((A**2)**.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:

\[ A+B+2-A=A+C , \text{AND } Z=(A+2+3)/A , \]

THEN BY SAYING PRINT A,Z
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
\[ (-b+\sqrt{b**2-4ac})*.5)/2a \]
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 CAIBASIC 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 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.

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:

\[
\text{LET \langle \text{ VARIABLE } \rangle = \langle \text{ EXPRESSION } \rangle} \quad \text{OR} \quad \text{LET \langle \text{ VARIABLE } \rangle = \langle \text{ VARIABLE } \rangle \ldots \langle \text{ EXPRESSION } \rangle}
\]

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:

\[
\begin{align*}
\text{LET A=12.5} \\
\text{LET B=H} \\
\text{LET A=B-25}
\end{align*}
\]


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 ALPHANUMERIC STRINGS. FOR EXAMPLE:

\[
\begin{align*}
\text{LET A=Da} & =X(5) = (3+2-6)/4 \\
\text{LET A=S} & = 'HELP'
\end{align*}
\]
LET Y=(B - 4*A*C)**.5
LET US='ANSWER ='

THE FOLLOWING ASSIGNMENT IS ILLEGAL:

LET A=D*/(3+4)

So

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

1

2.) REPLY WITH VALUE OF Z IN BELOW PROGRAM

DATA 1,2,3,4,6
DATA 'RIGHT ON', 7,8
READ A,B,X
READ G5,Y,C,D
LET ?X+Y
END

6

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=(A+B)/(C-D)
B.) R= A*(B/C)-D

8. BUILT-IN FUNCTIONS

BUILT-IN FUNCTIONS ARE COMMONLY USED PROGRAMS ALREADY WRITTEN AND STORED IN THE CAl BASIC 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 << (EXPRESSION) >>
WHERE THE EXPRESSION IS ENCLOSED IN PARENTHESES.

80

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.

80

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

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

1. IS THE FOLLOWING STATEMENT LEGAL
REPLY: YES, OR NO.

```
LET Z1=8+SQR(3.0*EXP(9.3))
```

yes

2. IS THE FOLLOWING SEQUENCE OF PROGRAM STATEMENTS LEGAL
REPLY: YES OR NO.

```
LET B=9
LET X=SQR(B)
...
```  

no

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:

```
LET B=9
IF B LT 0 THEN 100
SO LET X=SQR(X)
```
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 READ IN 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:
\[ P = \frac{S}{\left(1 + \frac{I}{100}\right)^N} \]
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 = (\(\pi/180\)) RADIANS. THE DATA FOR THE PROGRAM IS:
SIDE a=34.91 IN. ANGLE A=98.71 DEG.
SIDE b=45.97 DEG. ANGLE C=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.) $1072.74
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'
10 READ S,I,N
LET P=S*(1/(((1+I)**N))
PRINT P,S,I,N
GO TO 10
DATA 5000,.08,20,5000,.08,10,5000,.06,10
END

PRODUCES THE OUTPUT:

<table>
<thead>
<tr>
<th>INVESTMENT</th>
<th>PRINCIPAL</th>
<th>INTEREST</th>
<th>NR.YEARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3072.74</td>
<td>5000</td>
<td>.08</td>
<td>20</td>
</tr>
<tr>
<td>2315.97</td>
<td>5000</td>
<td>.08</td>
<td>10</td>
</tr>
<tr>
<td>2791.99</td>
<td>5000</td>
<td>.06</td>
<td>10</td>
</tr>
</tbody>
</table>

*** 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.

THE 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>SYMBOLS</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:

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 X1 LT A1 THEN 10

NO
IF(X**5 - 4) GE 10 THEN GO TO 100

yes

YOUR ANS\(\text{WER 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

2.) 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 3,5,7,10
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:

no

1. COUNT AND TEST METHOD, IN WHICH A COUNTER
IS INCREMENTS 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, INITIALIZE TO 0, AND COUNTS FROM 1-->10

94
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 THERE 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>100</td>
</tr>
<tr>
<td>11</td>
<td>110</td>
<td></td>
</tr>
</tbody>
</table>

THUS Z=110

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

REM READ AND TEST DEMONSTRATION
REM 9999 TO 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, 62, 40, 54, 56.

2.) WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME YEARS HENCE, AT VARYING INTEREST RATES. THE FORMULA IS:

\[ P = \frac{S}{(1 + I)^N} \]

WHERE \( P \) = PRESENT WORTH, \( S \) = PRINCIPAL, \( I \) = INTEREST, AND \( N \) = Nr. OF YEARS. FOR DATA USE \( S = 5000 \), AND \( I = .04 \), IN INCREMENTS OF .01, AND \( N = 20 \)

IF YOU DESIRE TO EXECUTE THESE PROGRAMS NOW

REPLY: YES; OTHERWISE REPLY: 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 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+I
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 TO 10.
LET C=0
FOR I=1 TO 10 STEP 1
LET C=C+I
NEXT I
PRINT 'SUM=',C
END
```

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
```
NEXT I
PRINT I
END

REPLY WITH VALUE OF I

1

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

REPLY WITH VALUE OF T.

5

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 VARIABLES.
BUT A1(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 L=1 TO 10
LET N(L)=0
NEXT L
READ K
FOR J=1 TO K
READ N(J)
NEXT J
LET L=N(1)
FOR J=2 TO K
IF N(J)<L THEN 10
LET L=N(J)
```

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

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 "A":
A(3,4) REFERS TO THE VALUE OF THE ELEMENT IN ROW 3, COLUMN 4
OF TABLE "A".

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:

<table>
<thead>
<tr>
<th>1,1</th>
<th>1,2</th>
<th>1,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,1</td>
<td>2,2</td>
<td>2,3</td>
</tr>
<tr>
<td>3,1</td>
<td>3,2</td>
<td>3,3</td>
</tr>
</tbody>
</table>

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

THIS TABLE "X" COULD BE FILLED AS FOLLOWS:

DIM X(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
END

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

6

NO WITHTABLE 'X' ASSIGNED VALUES CONSIDER THIS
PROGRAM (ASSUMING TABLE 'X' HAS BEEN FILLED BY THE ABOVE PROGRAM)

LET \( s = 0 \)
FOR \( i = 1 \) TO 3
FOR \( j = 1 \) TO 3
IF \( i \neq j \) THEN 5
LET \( s = s + x(i,j) \)
5 NEXT J
NEXT I
PRINT 'SUM = ', s
END

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:

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

\[
\text{DIM } \langle \text{TABLE VARIABLE} \rangle ( \langle \text{SIZE}, \text{SIZE} \rangle )
\]
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 REFERECE 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 CAI-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 PARENTHESIS.

```
DIM X(500),B(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) USING SUBSCRIPTS. YOU NOW HAVE ALL THE TOOLS TO BEGIN WRITING SOPHISTICATED 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 SUBROUTINE CAN BE 'CALLED' FROM ANYWHERE IN YOUR PROGRAM. 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;

no
IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEMS 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 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 BURDENOSO, A TECHNIQUE TO ECONOMIZE ON CODING INSTRUCTIONS CALLED A SUBROUTINE, IS PRESENTED. THE FORM OF THE SUBROUTINE IS:

GOSUB << STATEMENT NUMBER >>

***

***

RETURN

50

EXECUTION OF 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

***
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 ACCIDENTLY 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
GO TO 20
200 LET Q=INT(X/Y)
LET R=X-Q*Y
IF R=0 THEN 300
LET X=Y
LET Y=R
GO TO 200
300 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>
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,
GO SUB IOC
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
GOSUB 200
LET N=N+1
LET F=F*N
RETURN
DATA 2,5,8,6,-1
999 END

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*F(FACTORIAL OF N-1).

TO KEEP TRACK OF WHERE THE SUBROUTINE CALLS RETURN IN RECURSION IT HELPS TO EVISIONING A LAST-IN-FIRST-OUT (LIFO) STACK CONTAINING THE ADDRESS OF THE GOSUB STATEMENTS. EVERY 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>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td></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 <$2000, R = $0.0
   IF SALARY <$2500, R = $10
   IF SALARY >$2500, R = $20

B) TAX
   IF SALARY <$600, T = $5
   OTHERWIE T = .05(SALARY)

THE DATA IS:

EMPLOYEE GROSS SALARY
2. Write a recursive program to sum the numbers from x to y. Read in your own data and test your program.

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 CII-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. 5.0,7.31,-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 FOLLOWED BY A DIGIT. EG. A,A1,80,20
   - B) ALPHA VARIABLES: A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN ($) EG. A$V
   - C) SUBSCRIPTED VARIABLE: (SINGLE LETTER)
     (1) SINGLE SUBSCRIPT: <<LETTER>> (<<EXPRESSION>>)
     (2) DOUBLE SUBSCRIPT: <<LETTER>> (<<EXPRESSION,EXPRESSION>>)
\[ A(5), Z(5, 10), X(A+B, X**2) \]

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

B. 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 >>
                   ....
   NEXT << SIMPLE VARIABLE >>
12. DIM << LETTER >> ( << INTEGER EXPRESSION >> ) OR
    DIM << LETTER >> ( << INTEGER EXPRESSION, INTEGER EXPRESSION >> )
13. GOSUB << STATEMENT NUMBER >>
                   ....
RETURN

\textit{go}

\textit{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 1-147 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).}

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

\textit{no}

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

\textit{yes}

\textit{**** CAIBASIC COMPILER ****}

\textit{THE CAIBASIC COMPILER IS A LINE BY LINE INTERPRETER. THE BASIC}
INTERPRETER ACCEPTS STANDARD BASIC STATEMENTS, AND IT ANALYZES EACH BASIC STATEMENT AS IT IS INPUT. AN ADDED FEATURE OF THE CIA BASIC 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.)

**** CIA BASIC 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 gt 3 then 150
gosub 100
   150 return
9999 end

DO YOU WANT TO EDIT YOUR PROGRAM
REPLY: YES OR NO.

yes
**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 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 ADD followed by the basic statement. All blanks following the letters ADD 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 GO TO 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,ADD,...)

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 :

no

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
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


