DOCUMENT RESUME

ED 081 193 EM 011 339

TITLE	Teaching Guide and Problem Supplement. A Publication of the Exemplary Project Problem Solving Computer Style 1969-1970.
INSTITUTION	New Orleans Public Schools, La.
SPONS AGENCY	Bureau of Elementary and Secondary Education (DHEW/OE), Washington, D.C.
REPORT NO	DPSC-67-3834
PUB DATE	70
GRANT	OEG-3-7-70384-4813
NOTE	203p.
EDRS PRICE	MF-$0.65 HC-$9.87
DESCRIPTORS	Algebra; Chemistry; *Computer Assisted Instruction; Computers; *Computer Science Education; Mathematics; Physics; *Problem Solving; *Programing; Secondary Grades; Secondary School Students; *Teaching Guides; Trigonometry
IDENTIFIERS	Elementary Secondary Education Act Title III; ESEA Title III; Fortran IV; IBM 1130 Computer

ABSTRACT

Secondary school teachers incorporating the use of a computer in algebra, trigonometry, advanced mathematics, chemistry, or physics classes are the individuals for whom this book is intended. The content included in it is designed to aid the learning of programing techniques and basic scientific or mathematical principles, and to offer some solutions to illustrative problems. Eight units are devoted to a step-by-step explanation of the FORTRAN IV language and programing, with material presented in a manner which assists the teacher in developing lectures and other forms of instruction which facilitate student imitation and encourage early operation of his initial attempt at programing. Following this introduction to FORTRAN and to programing is a problem supplement with five sections, one each devoted to algebra, trigonometry, advanced math, chemistry, and physics. A short bibliography is also included. (Author/LB)

Teaching Guide
and
Problem Supplement

A Publication of the Exemplary Project
Problem Solving Computer Style
1969—1970
funded by
United States Office of Education
Grant OEG 3-7-70384-4813
under Title III
of the
Elementary and Secondary Education Act
of 1965

New Orleans Public Schools
Dr. Alton W. Cowan, Superintendent
703 Carondelet Street
New Orleans, Louisiana 70130

The work presented or reported herein was performed pursuant to a Grant from the U.S. Office of Education, Department of Health, Education, and Welfare. However, the opinions expressed herein do not necessarily reflect the position or policy of the U.S. Office of Education, and no official endorsement by the U.S. Office of Education should be inferred.

PROBLEM SOLVING COMPUTER STYLE PROJECT TEAM

Under the Direction of:

Dr. Malcolm F. Rosenberg Jr., Assistant Superintendent for Instruction
Olympia E. Boucree, Acting Supervisor of Mathematics
Solange G. Petersen, Coordinator

Teachers:

Michael A. Achary
Gradell L. Armand
Harold J. Contreary
Raymond P. Cougle
Jean I. Cullen
Edmond C. Drouet
June B. Fey
Milton Roos

1970

For information write:

Solange G. Petersen, Coordinator
Problem Solving Computer Style
731 St. Charles Avenue
New Orleans, Louisiana 70130

FOREWORD

The purpose of this book is to serve as a guide for those secondary school teachers incorporating the use of a computer in algebra, trigonometry, advanced mathematics, chemistry, or physics classes. While experimenting with the learning of programming, teachers in the ESEA Title III Project, Problem Solving-Computer Style, revised these outlines. Detailed notes, including many examples and teacher-made exercises, were distributed to the students in lieu of a text. The programs included here could serve as demonstrations or assignments. Selected to aid the learning of particular programming techniques and basic scientific or mathematical principles, these methods represent some possible — but not necessarily most efficient — solutions to given problems.

The language used to program these problems, FORTRAN IV for the IBM 1130, can be easily translated into a language suitable for other computers. In the early stages of programming, with the basic arithmetic statements of the FORTRAN language learned, a minimum of input and output instruction is given for two reasons: 1) to facilitate student imitation, and 2) to insure early operation of the student's first program. Having the student run his first program early and efficiently in the course provides motivation to pursue more difficult problems, often on the student's own initiative.

Arranging the material so that it both increases gradually in programming difficulty and remains consistent with the subject matter being learned as the computer-oriented discipline is often quite difficult. Some techniques must be presented with the least possible detail; complete coverage of these techniques must be deferred until sufficient knowledge can be gained in the regular course work. Most students are eager to experiment with each new technique as it is presented, and many seek out applications from topics currently being learned or those previously mastered. The more adventurous student tries to learn new topics on his own.

Since there are few textbooks on programming available at the secondary level, and since the manuals provided by the manufacturer are more suitable as reference books for those already knowledgeable in programming, the teacher must develop his own lectures and notes, particularly for the early stages of instruction. With the foundation established, the student should be capable of comprehending the numerous texts and manuals available for use at the college or the professional level. If computer time is made available to the capable, interested student, then only encouragement is needed to insure rapid progress.

Perhaps this book of outlines and illustrative problems will be helpful to teachers who are seeking a way to enrich their courses to prepare students for the "computer revolution".

TABLE OF CONTENTS

FORTRAN LANGUAGE

Unit I 3

What is FORTRAN?, 3. Basic Arithmetic Operations, 3. Arithmetic Statements, 4. Simple Input/Output Statements with Format, 4. STOP Statement, 5. END Statement, 5. CALL EXIT, 5. First FORTRAN Program for Solution of a Mathematical Formula, 5. Data Cards, 6. Control Cards, 9.

Unit II 13

Statement Numbers, 13. GO TO Statement, 13. Comment Statements, 13. Sample Continuous Loop Program, 13. IF Statement, 17. Basic Flowcharting Symbols, 17. Introduction to Flowcharting, 18.

Unit III 25

Counters, 25. A Problem With a Counter, 25. Column Heading, 27. DO Statements, 27. CONTINUE Statement, 28. A Problem Using a DO-loop, 28. Subscripted Variables, 32. DIMENSION Statement, 32. Problem Using Subscripts, 32.

Unit IV 35

Library Functions, 35. A Problem Using Library Functions, 36. Arithmetic Statement Functions, 38. A Problem Using Arithmetic Statement Functions, 38. Computed GO TO, 40. A Problem Using a Computed GO TO, 41.

Unit V 45

READ Statement, 45. FORMAT Statements, 47. Complete Input Operation, 48. WRITE Statements, 50. FORMAT Statements, 50.

Unit VI 55

More About DO-loops, 55. A Problem Using Nested DO-loops, 55. Type Statements, 58. Problem Using a Type Statement, 58. Problem Using DO-loops, 60. A Problem Using Alphabetic Information, 67. Another Version of the Same Program, 69. A Problem Having a Special Exit from a Loop, 71.

Unit VII 73

Function Subprogram, 73. Writing a Function Subprogram, 73. Problem Using a Function Subprogram, 74. How to Store a Function Subprogram, 78. Deleting a Program from the Disk, 78. How to Stack the Deck of a Program Having Subprograms, 78. Subroutine Subprograms, 79. Writing a Subroutine Subprogram, 80. Problem Using a Subroutine, 80.

Unit VIII 85

Double Subscripts, 85. Matrices, 85. Double Subscripted I/O Statements, 85. A Problem Using Double Subscripted Variables, 86. More DO-loop Problems, 89. Special FORTRAN Supplied Subroutines, 100.

BIBLIOGRAPHY 103

PROBLEM SUPPLEMENT

Algebra Section I 107

Distance Between Two Points, 107. The Square Root of Complex Numbers, 108. Division of Complex Numbers, 110. Iterative Demonstration, 112. Sorting Numbers, 114. Distance Form a Point to a Line, 116. Solution of Quadratic Equations, 118. Location of Real Roots, 120. Graph of a Quadratic, 122. Test, 126.

Trigonometry Section II 129

Changing From Polar to Rectangular Coordinates, 129. Listing Cosines of Angles, 130. Listing Tangents of Angles, 132. Law of Cosines, 134. Changing From Rectangular to Polar Coordinates, 136. Finding the Missing Force, 139. Solving Right Triangles, 141. The Ambiguous Case, 150. The Resultant of Two Forces, 153.

ADVANCED MATHEMATICS SECTION III 159

Limit of Sin X/X, 159. Area Under a Curve, 160. Ellipse, 162. Inverse of Any Matrix, 166.

CHEMISTRY SECTION IV 175

Corrected Volume, 175. C to F to K, 176. PH to Hydrogen Ion Concentrations, 178. Solving Einstein's Equation, 180. Universal Gas Constant, 181. Half-Life, 183. Gas Analysis, 185.

PHYSICS SECTION V 191

Free Fall, 191. Einstein, 192. Rocket, 194. Light Intensity, 196. Index of Refraction, 198. Parabolic Mirror, 202. Acceleration, 204.

EXPANDED TEACHING GUIDE

Fortran Language Unit I

A. *WHAT IS FORTRAN?*

FORTRAN is a symbolic language, similar in many ways to regular algebra, which is particularly adapted to the programming of scientific and mathematical problems. Made up of 22 different types of statements, FORTRAN was first designed by IBM and some of its customers for the IBM 704, for solution of engineering and scientific problems. The name FORTRAN is from *FOR*mula *TRAN*slation

B. *BASIC ARITHMETIC OPERATIONS*

1. *Arithmetic And FORTRAN Symbols*

	Arithmetic	*Fortran*
Addition	+	+
Subtraction	-	-
Multiplication	X	* (asterisk)
Division	÷	/ (slash)
Exponentiation (small number elevated to right)	X^2	**

2. *Order Of Operations*

a. Exponentiation is performed first.
b. Multiplication or division is done next moving from left to right.
c. Addition or subtraction is done last also moving from left to right.
d. All operations at each level of "hierarchy" must be completed before going on to the next level.

3. *Use Of Parentheses*

a. Computing begins within the innermost pair of parentheses when sets are contained within one another.
b. All parenthesized expressions must be completely executed before other operations are executed.
c. Within each pair of parentheses the usual rules of hierarchy of operations still apply.
d. The result of operations within the parentheses becomes the quantity used in the overall expression.

4. *Constants*

a. Numbers which are used to specify known quantities in mathematical expressions are called constants. Their values do not change through the problem.
b. Those constants which contain a decimal point are in the *real* mode. Real constants may be greater than 10^{-38}, but less than 10^{+38}, and seven digits are significant. They may be written in the normal decimal notation or in exponential form, which will be discussed in detail later.
c. Constants which do not contain a decimal point are in the *integer* mode. An integer constant must lie between -32,767 and +32,767.

5. *Variables*

a. Variables are symbols which represent quantities whose values are specified elsewhere in the program.
b. A variable name is used in the FORTRAN expression.
   (1) Names may have 1 to 5 alphabetic or numeric characters.
   (2) The first character must be alphabetic and is significant to the mode of the variable.
   (3) No imbedded blanks are permitted.
   (4) No operation signs are permitted.
   (5) All letters must be capitals.

6. *Arithmetic Modes*

a. The mode of an arithmetic expression is determined by the constants and variables in the expression.
   (1) Real mode
      (a) Constants contain decimal points.
      (b) Variable names begin with a letter other than I, J, K, L, M, or N.
   (2) Integer mode
      (a) Constants contain no decimal points.
      (b) Variable names begin with I, J, K, L, M, or N.

b. If variables and constants are not of the same mode, a mixed-mode expression results.

7. *FORTRAN Expressions*

Any single constant or variable, or any combination of constants, variables, and operation signs is known as a FORTRAN expression. All quantities in an expression should be of the same *mode*. The exception to this is that the mode of an exponent need not necessarily agree with the remainder of the arithmetic expression.

C. *FORTRAN ARITHMETIC STATEMENTS*

1. A FORTRAN arithmetic statement is made up of a variable name, followed by the equal sign and then followed by a FORTRAN expression.
2. The execution of an arithmetic statement is as follows; the expression on the right of the = sign is computed according to the rules of hierarchy and the result is *assigned* to the variable whose name appears on the left of the equal sign.

D. *SIMPLE INPUT/OUTPUT STATEMENTS WITH FORMAT*

1. Input statements begin with the word *READ* followed by an open parenthesis, 2 numerals separated by a comma, a closed parenthesis, and a list of variable names separated by commas. The first numeral within the parentheses is the code number for the input device to be used, the second is the number of an associated statement called a FORMAT, which describes the form in which the data will be arranged on the input device. 2 is the code for the card or card reader and 6 is the code for data to be entered on the console keyboard.

   READ (2,4) ALT, BASE

2. Output statements have almost the same form as input statements except that the opening word is WRITE and the code numbers are 1 for the console typewriter and 3 for the printer.

   WRITE (3,8) ALT, BASE, AREA

3. Format statements consist of a statement number, punched in columns 1 through 5, the word FORMAT, an open parenthesis, format codes, a closed parenthesis. A FORMAT defines the layout of data on a card, or a printed line.
   a. 4 FORMAT (2F8.2)
      (1) The FORMAT statement above, associated with the READ statement illustrated in Section D1 causes two real numbers (F code), each punched in 8 card columns and each having two decimal places to be read into memory. The first number will be stored as ALT and the second as BASE.
      (2) CODES
         F Real Numbers
         I Integers
         E Exponentials
         A Alphanumeric Characters
         H Alphanumeric Characters (literal translation)
         X Blanks
   b. 8 FORMAT (2X, 'ALTITUDE = ',F8.2,5X,'BASE = ',F8.2,5X, 'AREA = ',F8.2)
      This FORMAT is associated with the WRITE statement illustrated above.
      2X — allows 2 spaces to be skipped, one for carriage control and one for neatness.
      'ALTITUDE = ' — everything within the apostrophes is copied by the computer.

F8.2 — allows space for the value of the altitude to be printed. 8 is the field width. .2 specifies 2 decimal places to be printed.

5X — allows spacing for neatness.

E. *STOP STATEMENT*

1. The STOP statement consists of only the word STOP.

2. This statement brings the computer to a halt and it may not be restarted without returning to the beginning of the program.

F. *END STATEMENT*

1. The word END is the complete statement. It must be the final step of every program.

2. The END statement does nothing but tell the computer during the translation stage that the program that it has been receiving is complete. Every FORTRAN program must have END as its last statement.

G. *CALL EXIT*

1. The word CALL is the standard word in use for using a FORTRAN subroutine. EXIT is the name of a subroutine supplied by the manufacturer.

2. CALL EXIT may be used instead of the STOP statement to stop the execution of a program.

H. *FIRST FORTRAN PROGRAM FOR SOLUTION OF A MATHEMATICAL FORMULA.*

1. Coding Sheet

PROGRAM ALTR1
PROGRAMMER JOHN DOE JR

	FORTRAN STATEMENT
	READ (2,4)ALT,BASE
4	FORMAT (2F8.2)
	AREA = 1./2.*ALT*BASE
	WRITE (3,8)ALT,BASE,AREA
8	FORMAT (2X,'ALTITUDE =',F8.2,5X,'BASE =',F8.2,5X,'AREA =',F8.2)
	CALL EXIT
	END

2. Data Card

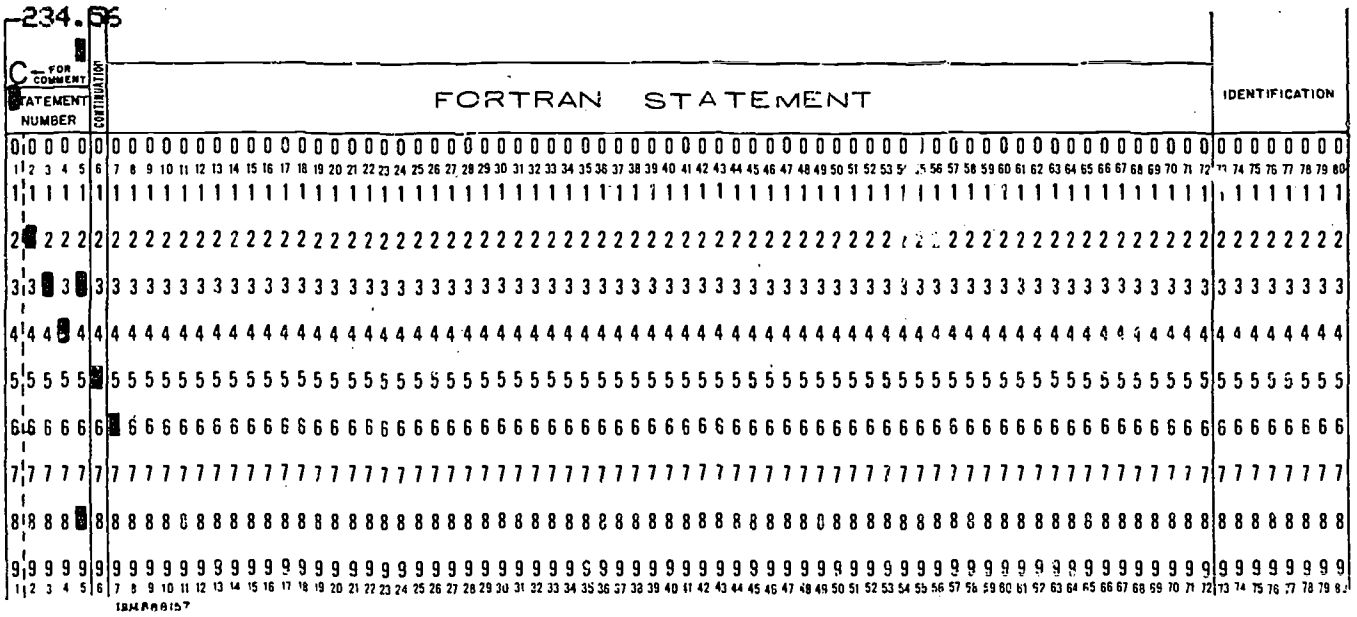
2.00 5.00

C — FOR COMMENT | STATEMENT NUMBER | CONTINUATION | FORTRAN STATEMENT | IDENTIFICATION

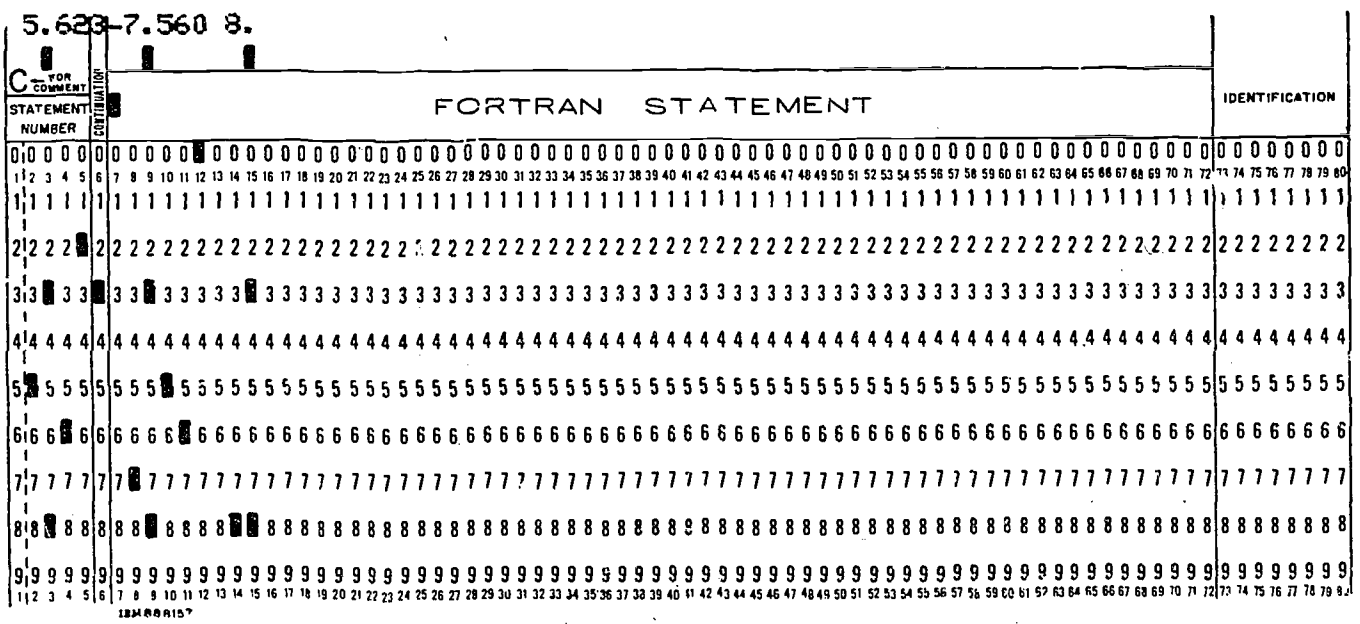
I. DATA CARDS

1. Agreement with FORMATS
It is a good practice to have the card punched with data and the FORMAT describing the data agree. The following are examples of data cards and agreeing FORMATS.

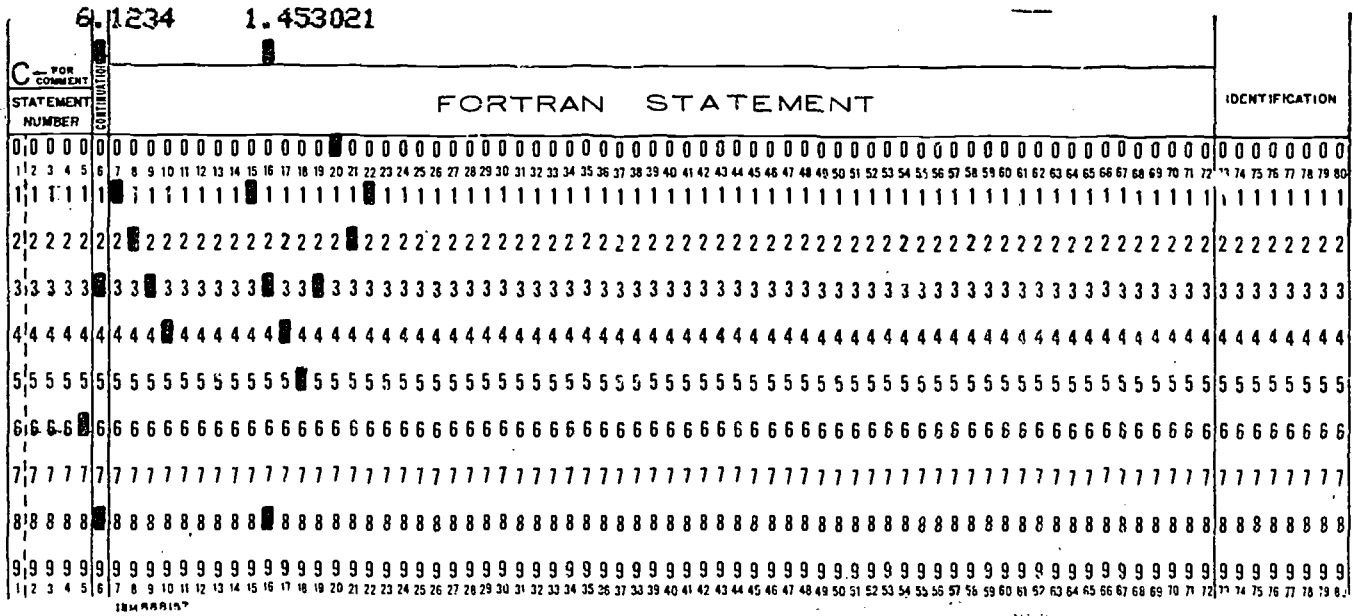
a. 1 FORMAT (F7.2)



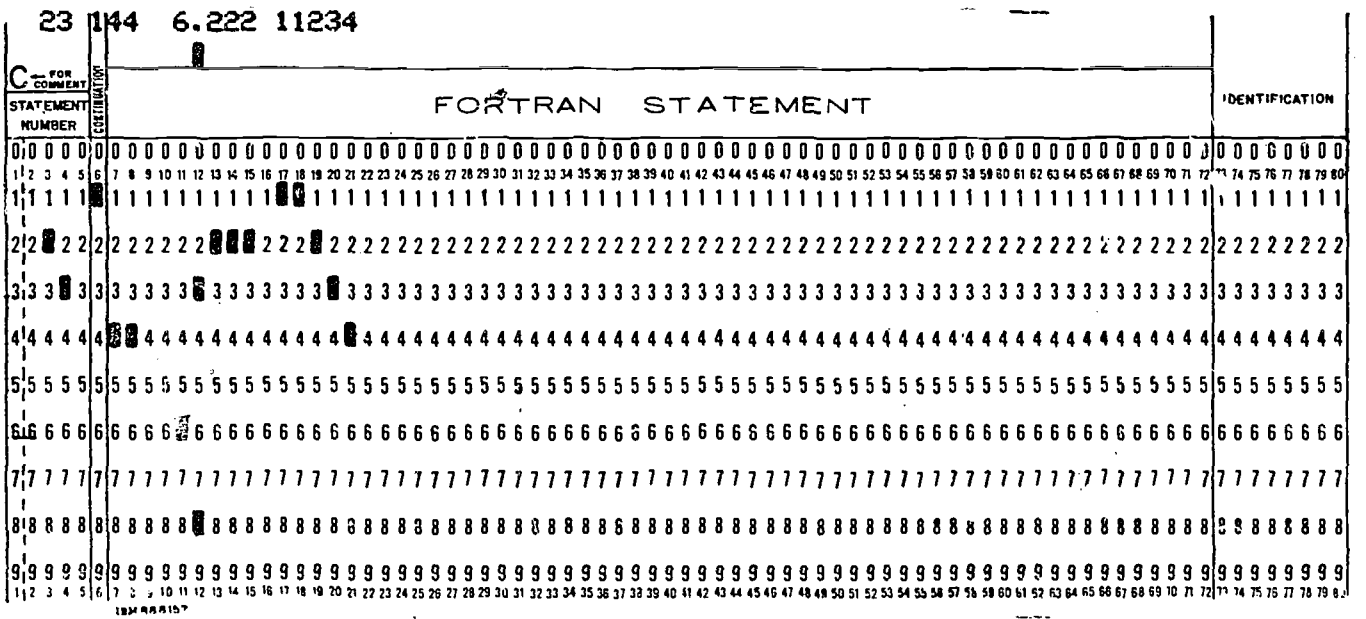
b. 2 FORMAT (3F6.3)



c. 3 FORMAT (F10.4,F12.6)



d. 4 FORMAT (2I4,F73,I6)



2. Data cards are placed behind the rest of the program after the // XEQ.

3. A blank card must be placed after the data cards.

J. CONTROL CARDS

1. Monitor Control Cards

The computer disk is loaded with a Supervisor or Monitor, which is a program to coordinate the operations of a computing system. The Supervisor controls the transition from job to job. The Monitor Control Cards needed at this time are:

// JOB	Start a new job
// FOR	Start a compilation (translation from FORTRAN to machine language)
// XEQ	Load and execute program

2. FORTRAN Control Cards

These cards may be placed in any sequence after the // FOR. FORTRAN control cards needed at this time are:

*IOCS (CARD,DISK,1132 PRINTER, TYPEWRITER,KEYBOARD)

Identifies all logical I/O devices to be used during execution of the program

*LIST SOURCE PROGRAM	List all program cards
**JOHN DOE JR.	For program identification: will print on top of every page

IBM FORTRAN Coding Form

CONTROL CARDS

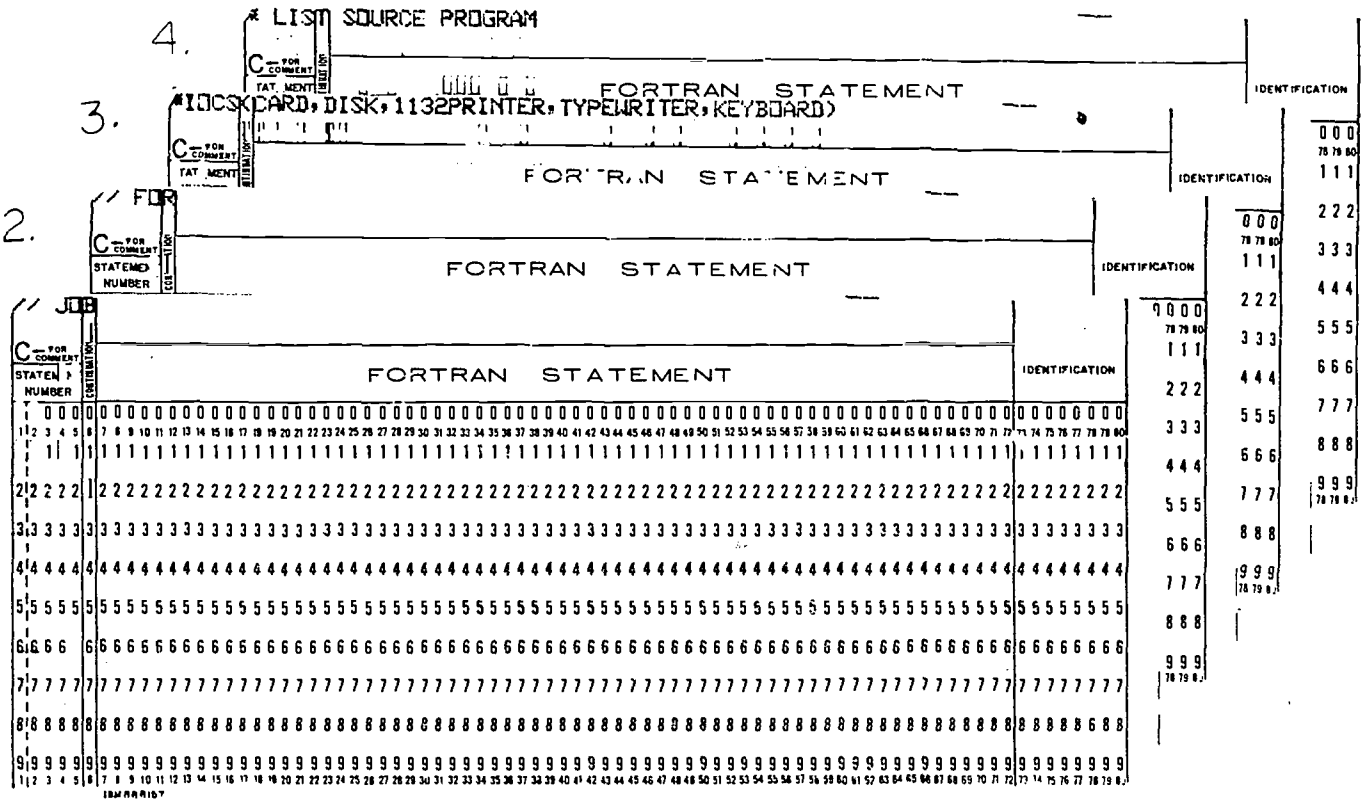
FORTRAN STATEMENT

```
// JOB
// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
**JOHN DOE JR.
// XEQ
```

DATA CARDS

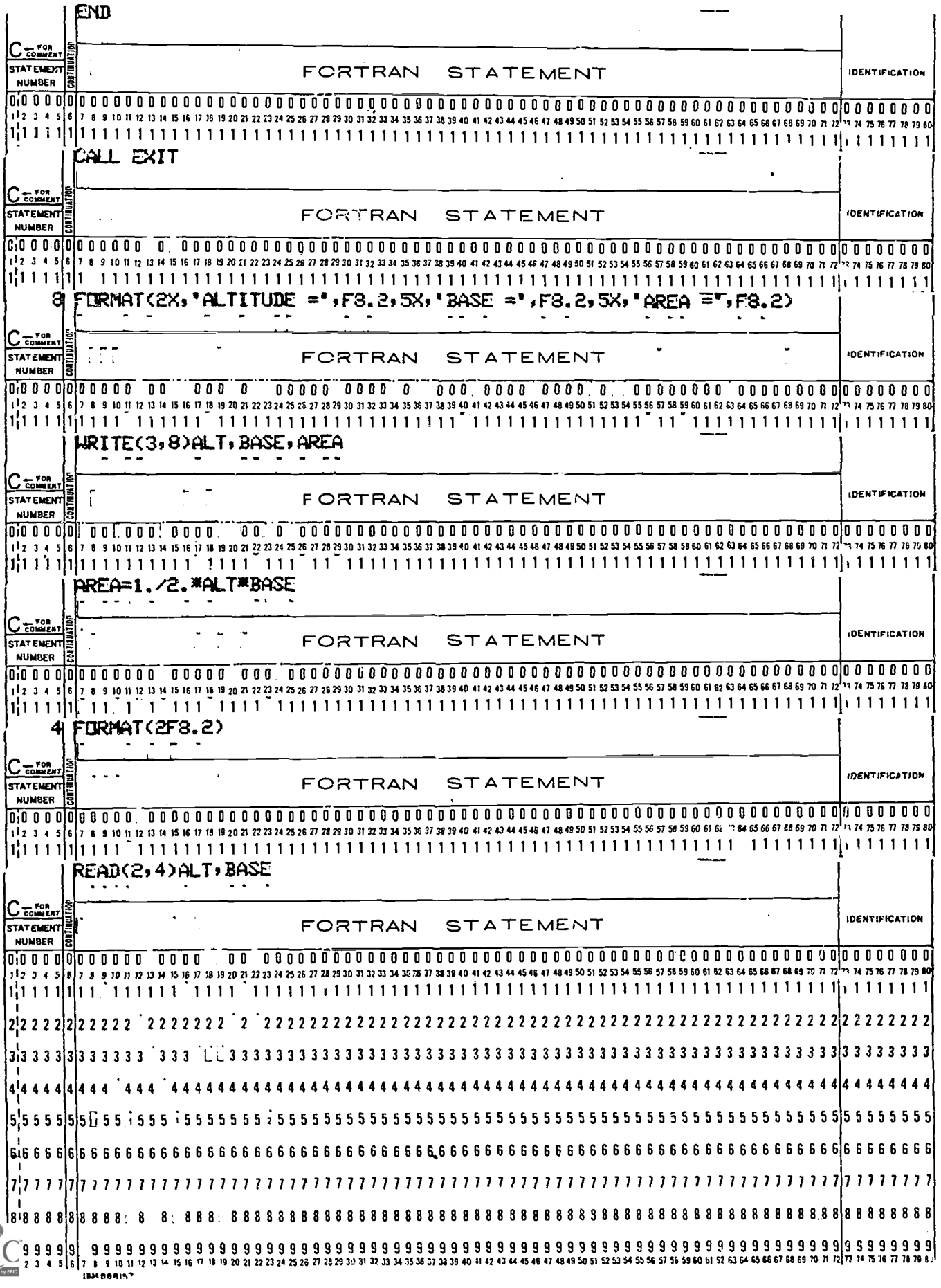
AFTER 'END' STATEMENT

PROGRAM



CONTROL CARDS

3. Program Cards



4. Printout

```
PAGE    1
// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0002         0002         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER, TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
      READ(2,4)ALT,BASE
    4 FORMAT(2F8.2)
      AREA=1./2.*ALT*BASE
      WRITE(3,8)ALT,BASE,AREA
    8 FORMAT(2X,'ALTITUDE =',F8.2,5X,'BASE =',F8.2,5X,'AREA =',F8.2)
      CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES       6  PROGRAM      86

 END OF COMPILATION

// XEQ
 ALTITUDE =     2.00      BASE =     5.00      AREA =     5.00
```

FORTRAN LANGUAGE

Unit II

A. *STATEMENT NUMBERS*

1. Statement numbers are assigned to FORTRAN statements by the programmer.

2. They are used to serve as an identification when the statement is referred to by another statement in the program.

3. They are placed to the left of the statement in columns 1 through 5.

4. Restrictions
   a. No two statements may have the same statement number.
   b. The number may not exceed 5 digits.

B. *GO TO STATEMENT*

1. The GO TO statement is used to change the sequential order of execution of program steps. It is one of the FORTRAN control statements and is sometimes called an unconditional branch.

2. This statement consists of the words GO TO followed by a statement number, for example:

   GO TO 30

3. The statement following any GO TO statement must be numbered.

C. *COMMENT STATEMENTS*

1. Comment statements allow the programmer to insert comments or directions. These will not appear in the output unless a special control card is used. A well documented program will use many such statements.

2. C is placed in column 1 of the FORTRAN statement card. The comment may be entered in any manner on the balance of the card.

D. *SAMPLE CONTINUOUS LOOP PROGRAM*

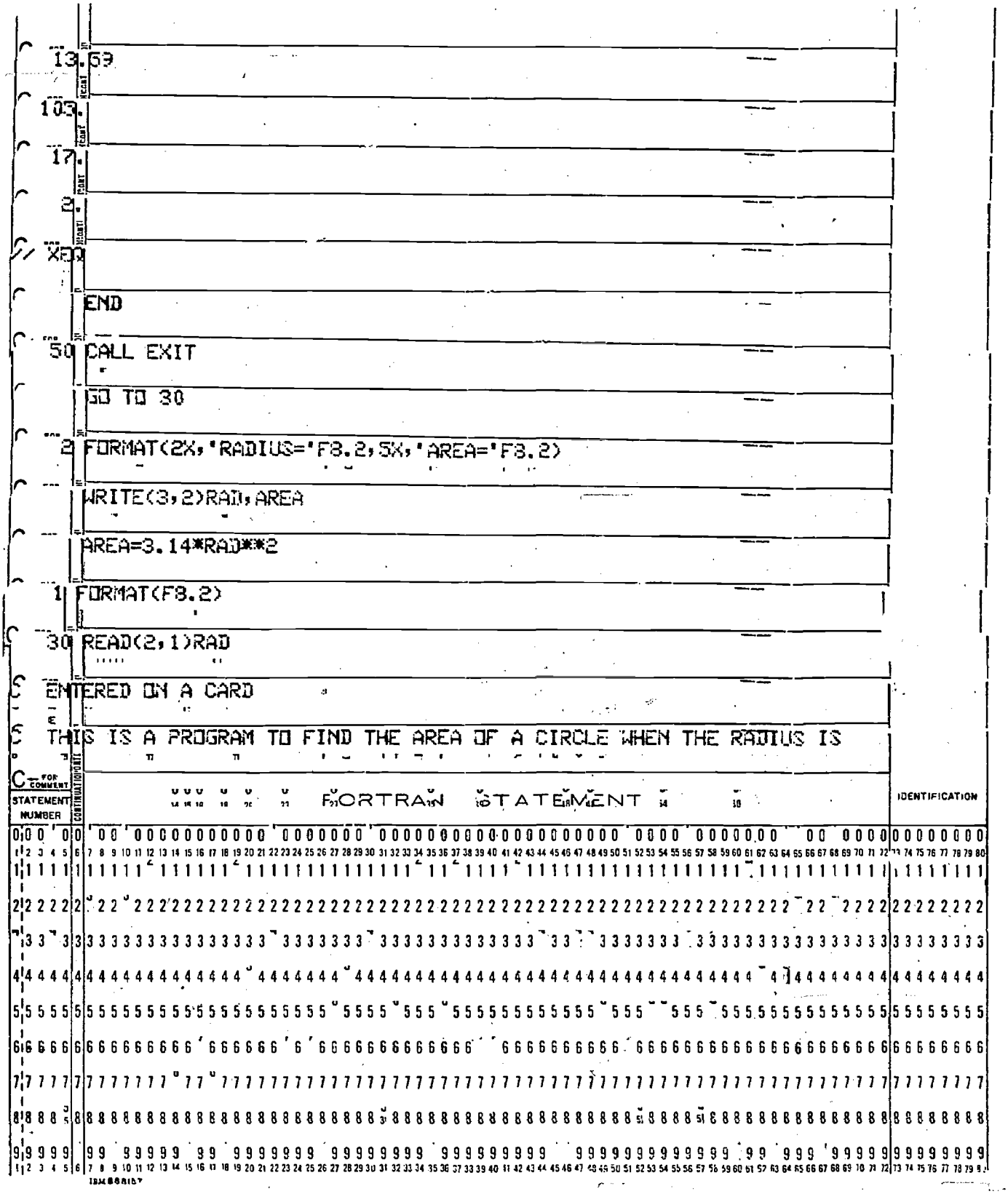
```
C THIS IS PROGRAM TO FIND THE AREA OF A CIRCLE WHEN THE RADIUS IS
C ENTERED ON A CARD.
 30 READ(2,1) RAD
  1 FORMAT(F8.2)
    AREA= 3.14*RAD**2
    WRITE(3,2) RAD,AREA
  2 FORMAT(2X,'RADIUS=',F8.2,5X,'AREA =',F8.2)
    GO TO 30
 50 CALL EXIT
    END
```

IBM FORTRAN Coding Form

ARCIR
JOHN DOE JR.

FORTRAN STATEMENT

```
C THIS IS A PROGRAM TO FIND THE AREA OF A CIRCLE WHEN THE RADIUS IS
C ENTERED ON A CARD
   30 READ (2,1)RAD
    1 FORMAT (F8.2)
      AREA = 3.14*RAD**2
      WRITE (3,2)RAD,AREA
    2 FORMAT (2X,'RADIUS =',F8.2,5X,'AREA =',F8.2)
      GO TO 30
   50 CALL EXIT
      END
// XEQ
  2.
 1.7.
 105.
 13.59
```



PROGRAM CARDS AND DATA CARDS
AREA OF A CIRCLE

```
PAGE    1

// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
   0000         0002         0002         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
C   THIS IS A PROGRAM TO FIND THE AREA OF A CIRCLE WHEN THE RADIUS IS
C   ENTERED ON A CARD
    30 READ(2,1)RAD
     1 FORMAT(F8.2)
       AREA=3.14*RAD**2
       WRITE(3,2)RAD,AREA
     2 FORMAT(2X,'RADIUS='F8.2,5X,'AREA='F8.2)
       GO TO 30
    50 CALL EXIT
       END

UNREFERENCED STATEMENTS
 50

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES       4  PROGRAM      72

 END OF COMPILATION

// XEQ

 RADIUS=     2.00      AREA=    12.56
 RADIUS=    17.00      AREA=   907.46
 RADIUS=   105.00      AREA=34618.50
 RADIUS=    13.59      AREA=   579.92
```

OUTPUT - THE AREA OF A CIRCLE

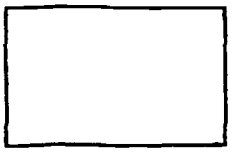
E. *IF STATEMENT*

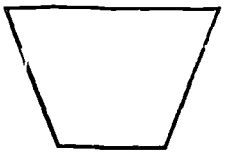
1. IF statements are used to cause the computer to branch to one of three possible statements, depending on the value of a FORTRAN expression. This is known as a "decision."

2. The form of the IF statement is the word IF, followed by a pair of parentheses containing a FORTRAN expression, followed by exactly 3 statement numbers separated by commas.
   IF (X-Y) 60,20,40

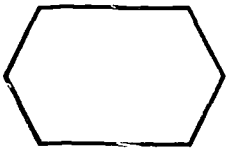
3. Execution — The expression within the parentheses is evaluated. If the value is negative the next statement to be executed would be statement 60, if the value is zero, statement 20 will be executed next, and if the value is positive, statement 40 will be executed next.

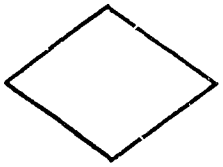
F. BASIC FLOWCHARTING SYMBOLS

1. 
processing

2. 
output

3. 
input

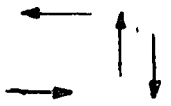
4. 
subroutine

5. 
decision

6. 
terminal (start or end)

7. 

N=1,2...
connector

8. 
flow direction

9. 

N=1,2...
offpage connector

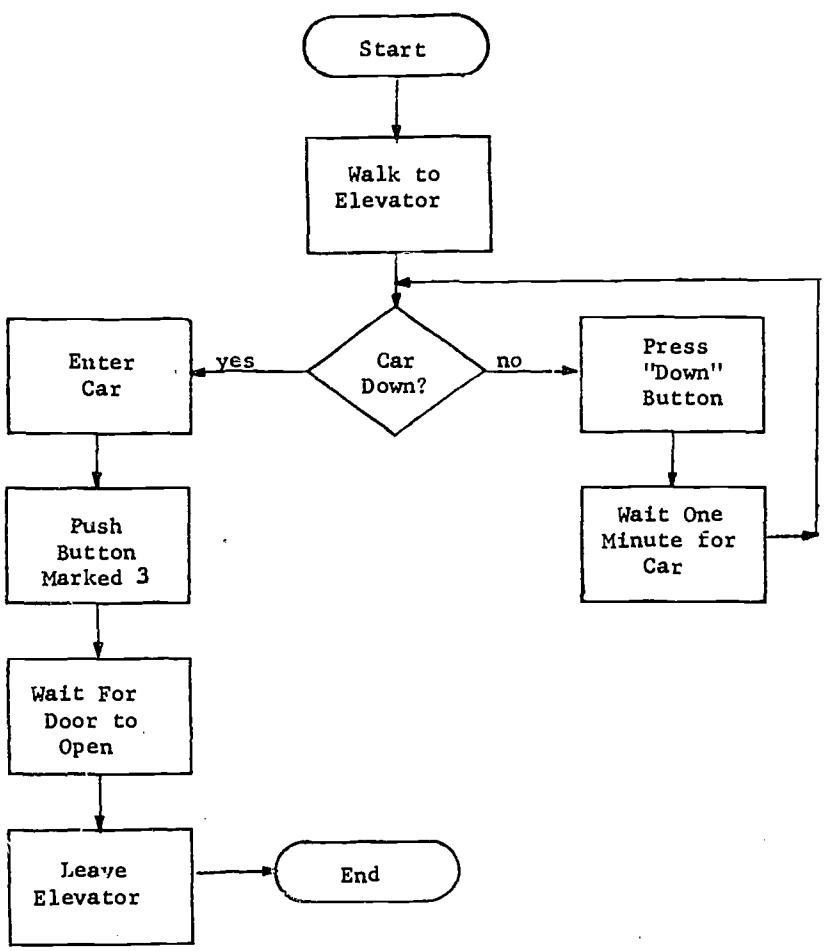
H. *INTRODUCTION TO FLOWCHARTING*

1. A flowchart is defined as a graphic representation of a sequence of instructions to accomplish a given task.

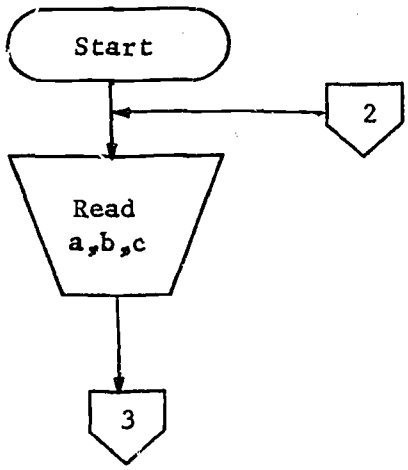
2. Used as a basic step to problem solving, flowcharting becomes essential as the difficulty of the problem presented increases.

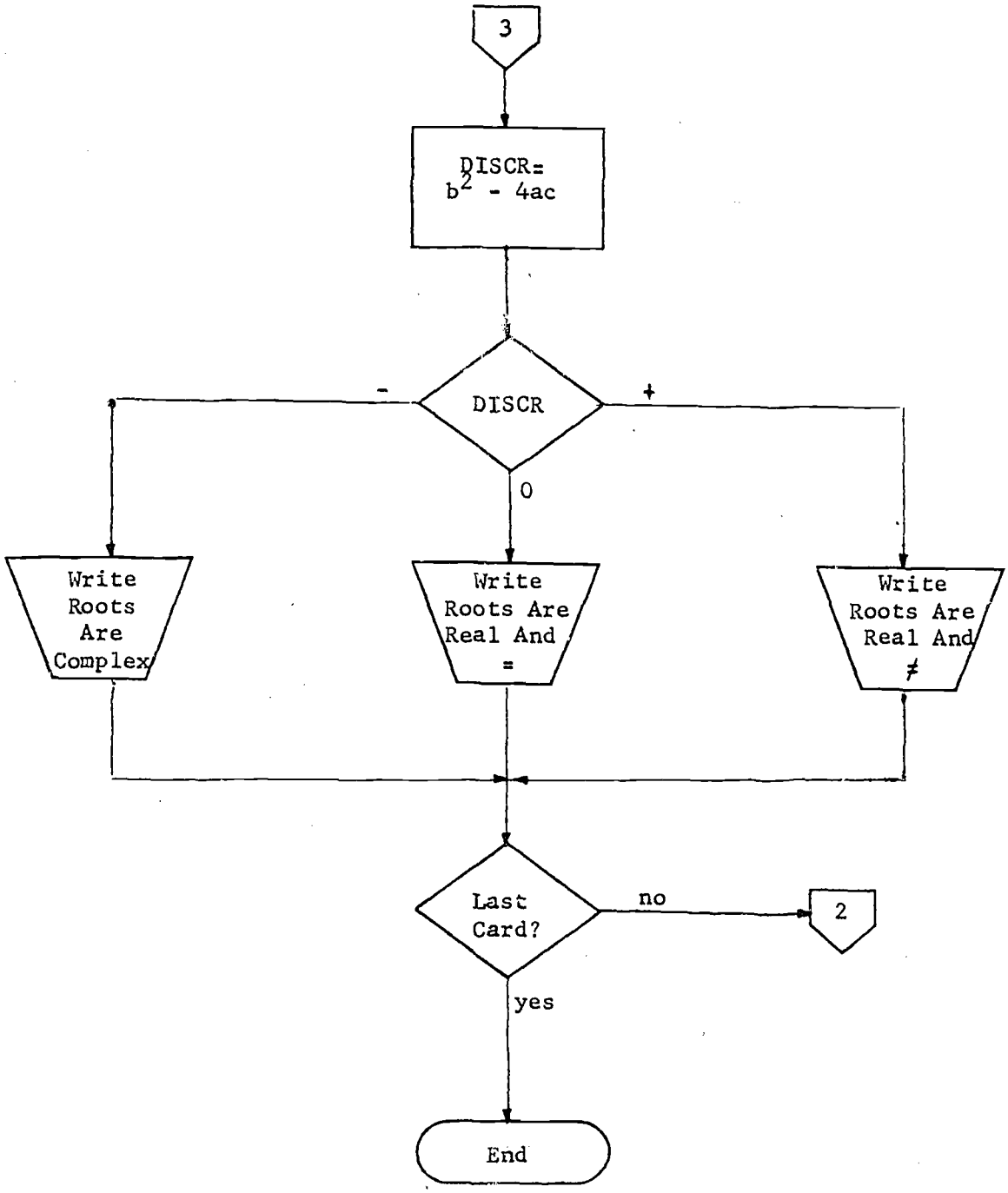
3. Flowcharting a Non-Mathematical Problem
   a. Problem — Go to the 3rd floor of a building from the lobby, using an elevator.

   b. Step by Step Solution
      (1) Walk to elevator.
      (2) Is car at your level?
      (3) If yes:
         (a) Enter car.
         (b) Push button marked 3.
         (c) Wait for car to stop at 3.
   c. Flowcharted Solution
         (d) Wait for door to open.
         (e) Leave elevator.
      (4) If no:
         (a) Press *down* button.
         (b) Wait one minute for car.
         (c) Go to step 2.



4. Flowcharting a Mathematical Problem
   a. Problem —
      (1) Read the 3 coefficients of a quadratic equation from each card.
      (2) Determine the nature of the roots of each equation.
      (3) Print out an answer.
   b. Step by Step Solution
      (1) Read coefficients.
      (2) Compute the discriminant, $b^2 - 4ac$.
      (3) Is the discriminant,-,0,or +?
      (4) If negative, write, "Roots are complex".
      (5) Is this the last problem?
         (a) If yes, end program.
         (b) If no, read another card.
      (6) If positive, write, "Roots are real and unequal".
      (7) Go to step 5.
      (8) If zero, write, "Roots are real and equal".
      (9) Go to step 5.
   c. Flowcharted Solution




3
DISCR=
b2 - 4ac
DISCR
-
+
0
Write
Roots
Are
Complex
Write
Roots Are
Real And
=
Write
Roots Are
Real And
≠
Last
Card?
no
2
yes
End

d. Coding Form

IBM FORTRAN Coding Form

NATRO
JOHN DOE JR.

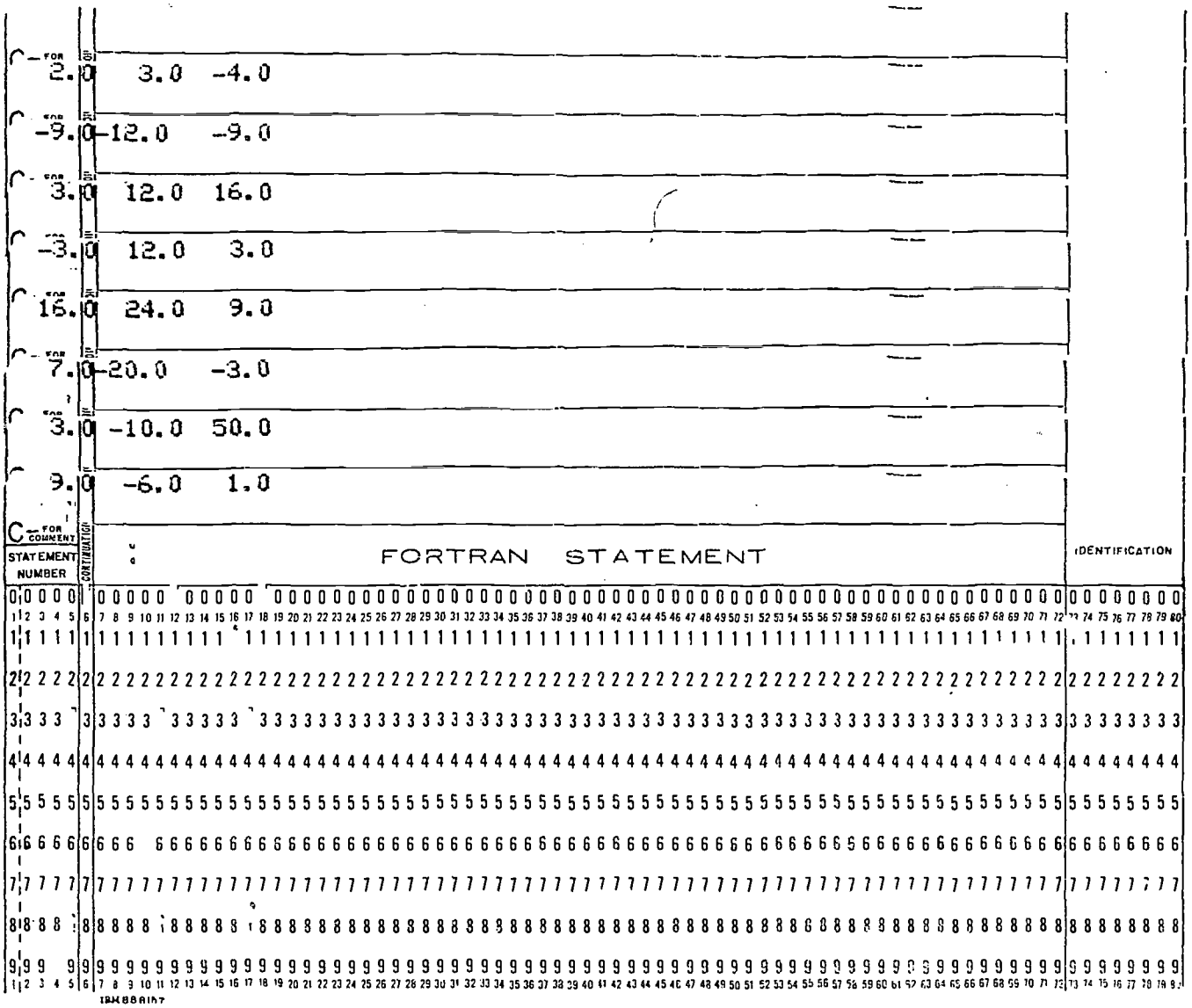
FORTRAN STATEMENT

```
C THIS PROGRAM WILL FIND THE NATURE OF THE ROOTS OF A QUADRATIC EQUATION
  21 READ (2,1)A,B,C
   1 FORMAT (3F6.1)
     DISCR = B**2-4.*A*C
     IF (DISCR) 4,6,8
   4 WRITE (3,5)A,B,C
   5 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE COMPL
    1EX')
     GO TO 21
   8 WRITE (3,9)A,B,C
   9 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE REAL
    1AND UNEQUAL')
     GO TO 21
   6 WRITE (3,7)A,B,C
   7 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE REAL
    1AND EQUAL')
     GO TO 21
  15 CALL EXIT
     END
```

```
      END
   13 CALL EXIT
      GO TO 21
     1AND EQUAL')
    7 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE REAL
    6 WRITE(3,7)A,B,C
      GO TO 21
C     1AND UNEQUAL')
    9 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE REAL
    8 WRITE(3,9)A,B,C
      GO TO 21
     1EX')
    5 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE COMPL
    4 WRITE(3,5)A,B,C
      IF(DISCR)4,6,8
      DISCR= B**2-4.*A*C
    1 FORMAT(3F6.1)
   21 READ(2,1)A,B,C
C  THIS PROGRAM WILL FIND THE NATURE OF THE ROOTS OF A QUADRATIC EQUATION
```

STATEMENT NUMBER | CONTINUATION | FORTRAN STATEMENT | IDENTIFICATION

PROGRAM CARDS - NATURE OF THE ROOTS OF A QUADRATIC EQUATION



DATA CARDS - NATURE OF THE ROOTS OF A QUADRATIC EQUATION

c. Printout

```
PAGE    1
// JOB
LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0002         0002         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
C  THIS PROGRAM WILL FIND THE NATURE OF THE ROOTS OF A QUADRATIC EQUATION
    21 READ(2,1)A,B,C
     1 FORMAT(3F6.1)
       DISCR= B**2-4.*A*C
       IF(DISCR)4,6,8
     4 WRITE(3,5)A,B,C
     5 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE COMPL
      1EX')
       GO TO 21
     8 WRITE(3,9)A,B,C
     9 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE REAL
      1AND UNEQUAL')
       GO TO 21
     6 WRITE(3,7)A,B,C
     7 FORMAT (2X,'A=',F6.1,2X,'B=',F6.1,2X,'C=',F6.1,2X,'ROOTS ARE REAL
      1AND EQUAL')
       GO TO 21
    15 CALL EXIT
       END

UNREFERENCED STATEMENTS
 15

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES      10  PROGRAM     182

 END OF COMPILATION

// XEQ

 A=    9.0   B=   -6.0   C=    1.0   ROOTS ARE REAL AND EQUAL
 A=    3.0   B=  -10.0   C=   50.0   ROOTS ARE COMPLEX
 A=    7.0   B=  -20.0   C=   -3.0   ROOTS ARE REAL AND UNEQUAL
 A=   16.0   B=   24.0   C=    9.0   ROOTS ARE REAL AND EQUAL
 A=   -3.0   B=   12.0   C=    3.0   ROOTS ARE REAL AND UNEQUAL
 A=    3.0   B=   12.0   C=   16.0   ROOTS ARE COMPLEX
 A=   -9.0   B=  -12.0   C=   -9.0   ROOTS ARE COMPLEX
 A=    2.0   B=    3.0   C=   -4.0   ROOTS ARE REAL AND UNEQUAL
```

OUTPUT - NATURE OF ROOTS OF A QUADRATIC EQUATION

FORTRAN LANGUAGE

Unit III

A. *COUNTERS*

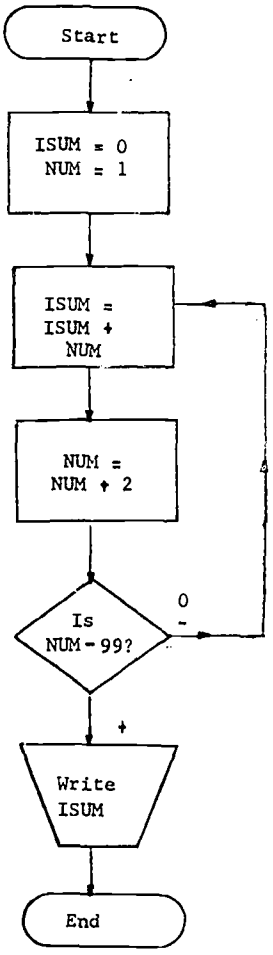
1. Counters are used to cause a program or a part of a program to repeat itself a designated number of times, or to produce an output which records the number of items which have been processed.

2. Several statements are required to construct a counter.
   a. The value of an integer is initialized at the beginning of a program.
      N=1
   b. After the FORTRAN statements which solve the problem have been completed, the counter is increased by
      N=N+1
   c. The current value of N is tested against its limit; if it reaches its limit, the program progresses to a new set of procedures.
      IF (N-100)5,5,10

B. *A PROBLEM WITH A COUNTER*

1. Problem: Find the sum of the odd integers from 1 to 99.

2. Solution:
   a. Decide on variable names.
      (1) ISUM - sum of the integers
      (2) NUM - numbers to be added
   b. Flowchart the problem.

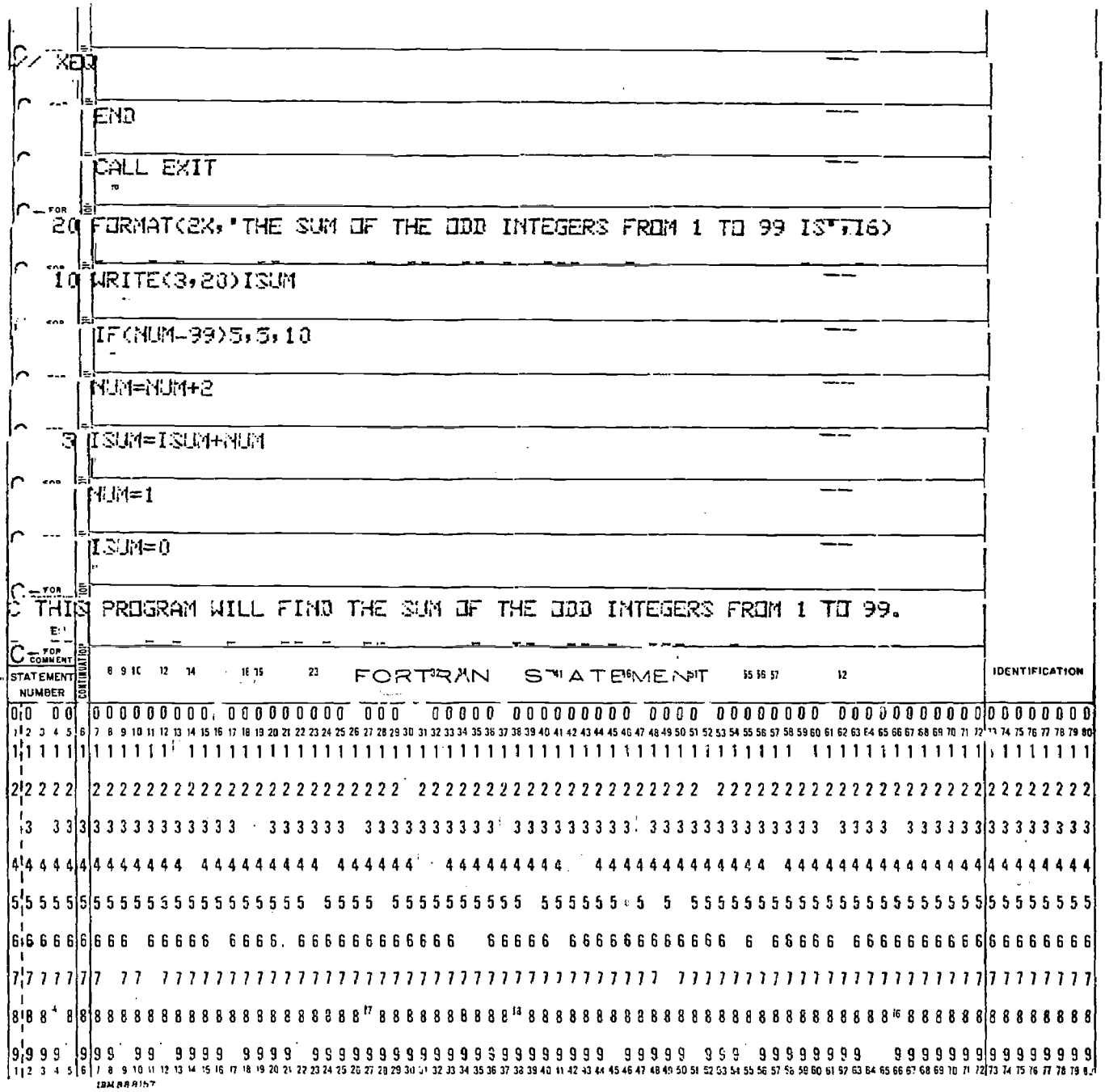


c. Coding Form

IBM

```
C.  THIS PROGRAM WILL FIND THE SUM OF THE ODD INTEGERS FROM 1 TO 99.
      ISUM=0
      NUM=1
    5 ISUM=ISUM+NUM
      NUM=NUM+2
      IF(NUM-99)5,5,10
   10 WRITE(3,20)ISUM
   20 FORMAT(2X,'THE SUM OF THE ODD INTEGERS FROM 1 TO 99 IS',I6)
      CALL EXIT
      END
```

d. Punched Cards



PUNCHED CARDS FOR THE SUM OF ODD INTEGERS

e. Printout

```
PAGE    1
// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0001         0001         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
C   THIS PROGRAM WILL FIND THE SUM OF THE ODD INTEGERS FROM 1 TO 99.
      ISUM=0
      NUM=1
    5 ISUM=ISUM+NUM
      NUM=NUM+2
      IF(NUM-99)5,5,10
   10 WRITE(3,20)ISUM
   20 FORMAT(2X,'THE SUM OF THE ODD INTEGERS FROM 1 TO 99 IS',I6)
      CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES      4  PROGRAM      94

END OF COMPILATION

// XEQ

 THE SUM OF THE ODD INTEGERS FROM 1 TO 99 IS  2500
```

C. *COLUMN HEADINGS*

1. Instead of identifying each term in a set of answers, it may be convenient to list answers under one set of column headings.

2. Procedure
   a. The WRITE statement, without a variable list is used.
      WRITE(3,24)
   b. The FORMAT statement used in combination with the WRITE contains the headings. Care must be taken to allow for spacing between headings so that columns will be neatly spaced and headings either centered above or right-adjusted.
      24 FORMAT(2X,`RADIUS`,5X,`AREA`)

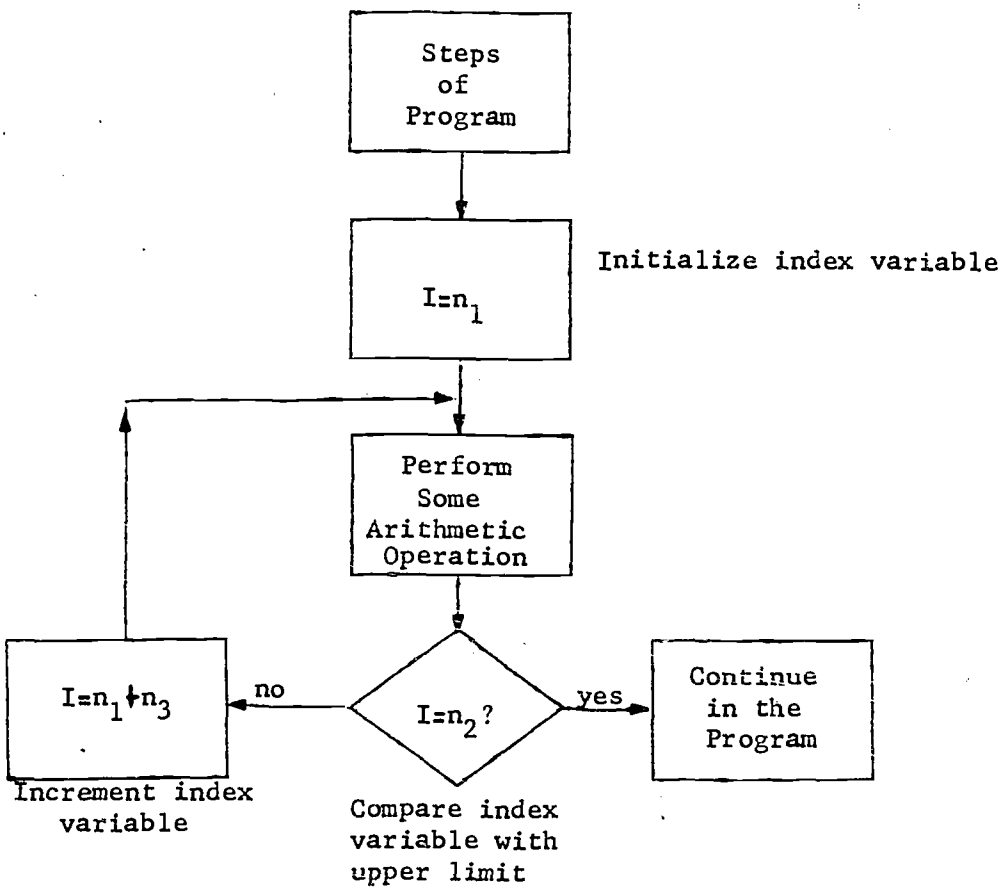
D. *DO STATEMENTS*

1. The DO statement is a single statement which combines the 2 operations which control a loop, the counter and the IF.

2. The statement consists of the word DO, a statement number, and an index definition.
   DO 20 $I=n_1,n_2,n_3$

3. The index definition consists of a non-subscripted integer variable, an equal sign, and 2 or 3 integer quantities (constant or variable), separated by commas. The first integer is the lower limit of the loop, the second is the upper limit, and the third is an optional increment. If the increment is not specified, it is assumed to be 1.

4. Execution
   a. The computer sets the index variable equal to the lower limit.
   b. All statements up to and including the statement number in the DO statement itself are executed.
   c. The index variable is increased by the increment, or by 1, if no increment is specified.
   d. The current value of the index is checked against the upper limit. When the index reaches its greatest possible value without exceeding the upper limit, drop out of the loop.
   e. Flowchart of a DO loop.



E. *CONTINUE STATEMENT*

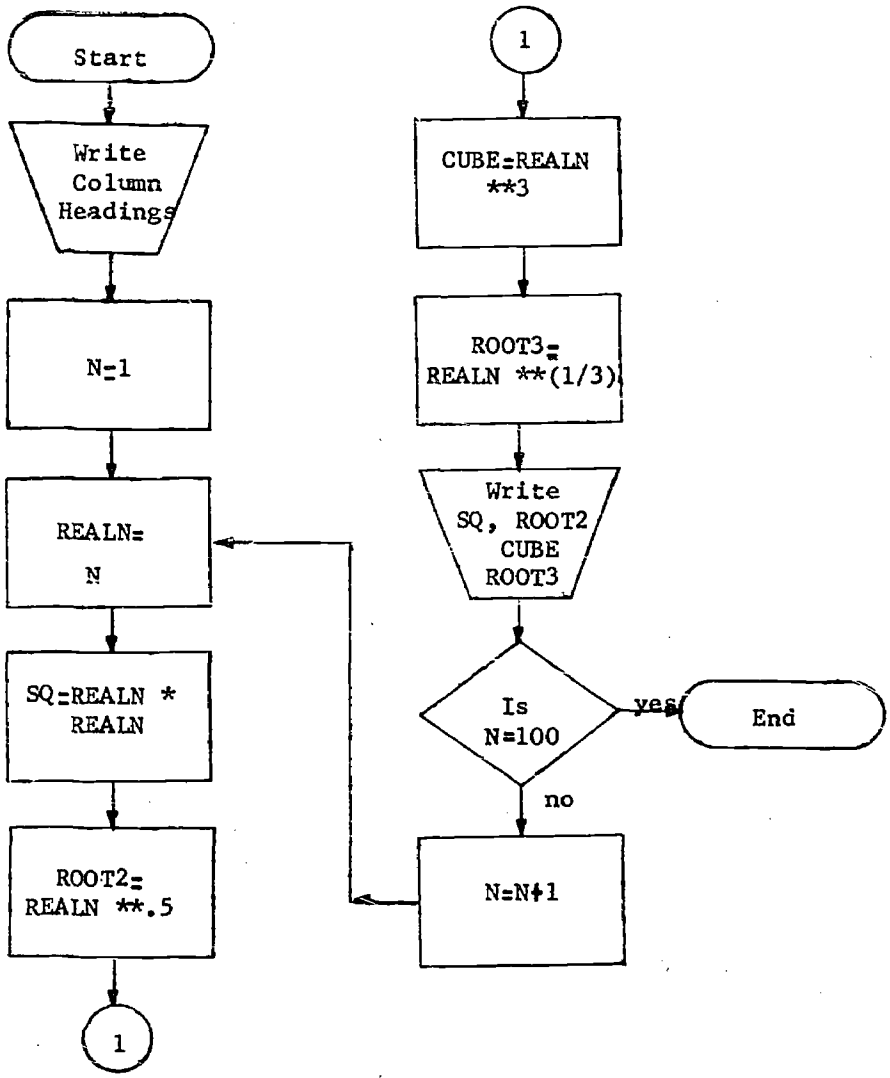
1. This statement provides the DO statement with a reference number in a loop which might otherwise end with a control statement, e.g. IF or GO TO.

2. It consists of only the word CONTINUE.

F. *A PROBLEM USING A DO-LOOP*

1. Problem — Find the squares, square-roots, cubes, and cube-roots of the integers from 1 to 100.

2. Flowchart



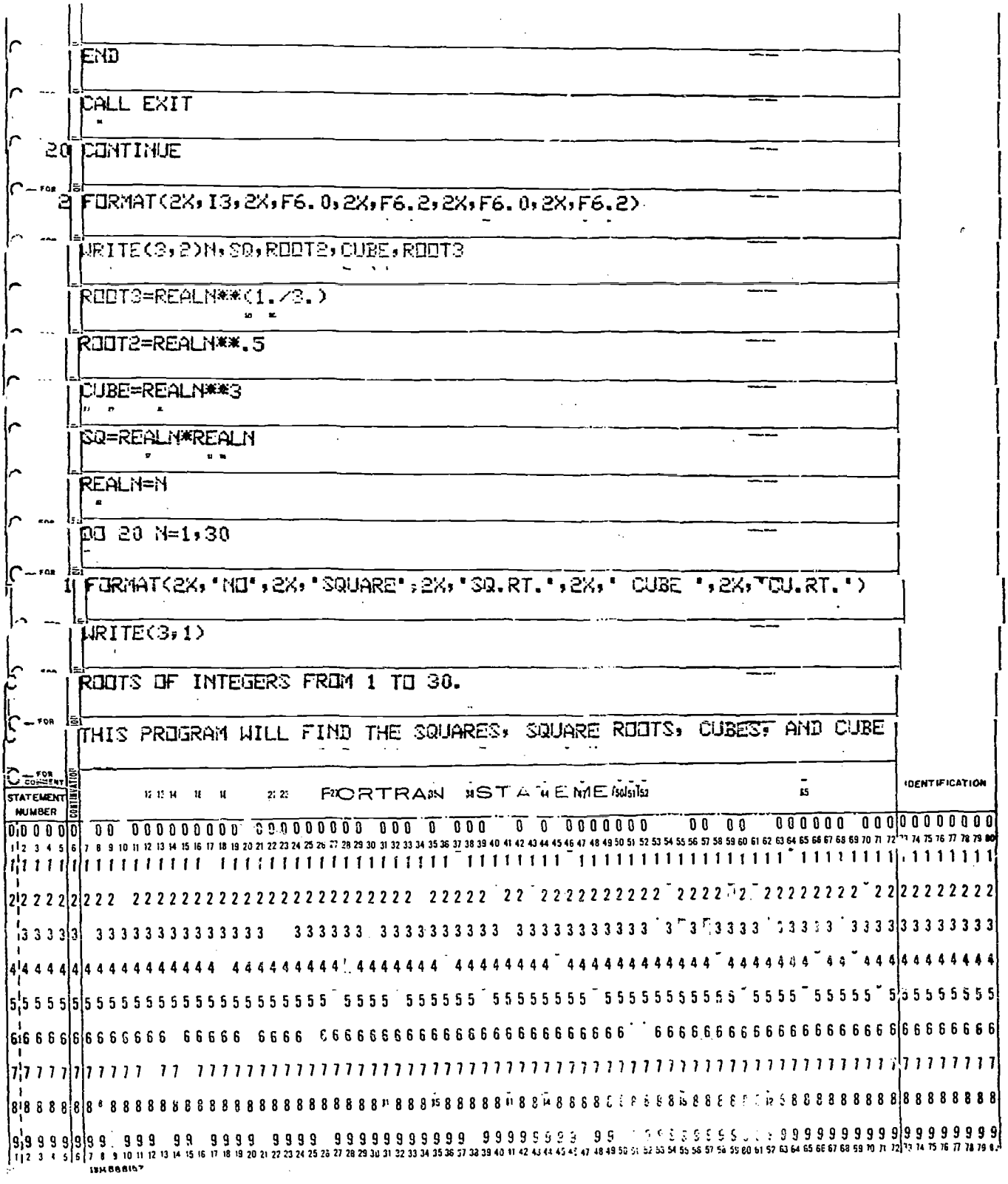
3. Coding Form

IBM FORTRAN Coding Form

FORTRAN STATEMENT

```
C       THIS PROGRAM WILL FIND THE SQUARES, SQUARE ROOTS, CUBES, AND CUBE
C       ROOTS OF INTEGERS FROM 1 TO 30.
        WRITE(3, 1)
    1   FORMAT(2X,'NO', 2X,'SQUARE', 2X,'SQ.RT.', 2X,' CUBE ', 2X,'CU.RT.')
        DO 20 N=1,30
        REALN=N
        SQ=REALN*REALN
        CUBE=REALN**3
        ROOT2=REALN**.5
        ROOT3=REALN**(1./3.)
        WRITE(3, 2)N, SQ, ROOT2, CUBE, ROOT3
    2   FORMAT(2X, I3, 2X, F6.0, 2X, F6.2, 2X, F6.0, 2X, F6.2)
   20   CONTINUE
        CALL EXIT
        END
```

4. Program Cards



A PROBLEM USING A DO-LOOP
SQUARES, SQUARE ROOTS, CUBES, CUBE ROOTS

5. Printout

```
PAGE    1
// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0002         0002         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST  SOURCE PROGRAM
C       THIS PROGRAM WILL FIND THE SQUARES, SQUARE ROOTS, CUBES, AND CUBE
C       ROOTS OF INTEGERS FROM 1 TO 30.
        WRITE(3,1)
     1  FORMAT(2X,'NO',2X,'SQUARE',2X,'SQ.RT.',2X,' CUBE ',2X,'CU.RT.')
        DO 20 N=1,30
        REALN=N
        SQ=REALN*REALN
        CUBE=REALN**3
        ROOT2=REALN**.5
        ROOT3=REALN**(1./3.)
        WRITE(3,2)N,SQ,ROOT2,CUBE,ROOT3
     2  FORMAT(2X,I3,2X,F6.0,2X,F6.2,2X,F6.0,2X,F6.2)
    20  CONTINUE
        CALL EXIT
        END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES      14  PROGRAM     142

 END OF COMPILATION

// XEQ

 NO   SQUARE   SQ.RT.    CUBE    CU.RT.
   1      1.     1.00      1.     1.00
   2      4.     1.41      8.     1.25
   3      9.     1.73     27.     1.44
   4     16.     1.99*    64.     1.58
   5     25.     2.23    125.     1.70
   6     36.     2.44    216.     1.81
   7     49.     2.64    343.     1.91
   8     64.     2.82    512.     1.99
   9     81.     2.99    729.     2.08
  10    100.     3.16   1000.     2.15
  11    121.     3.31   1331.     2.22
  12    144.     3.46   1728.     2.28
  13    169.     3.60   2197.     2.35
  14    196.     3.74   2744.     2.41
  15    225.     3.87   3375.     2.46
  16    256.     3.99   4096.     2.51
  17    289.     4.12   4913.     2.57
  18    324.     4.24   5832.     2.62
  19    361.     4.35   6859.     2.66
  20    400.     4.47   8000.     2.71
  21    441.     4.58   9261.     2.75
  22    484.     4.69  10648.     2.80
  23    529.     4.79  12167.     2.84
  24    576.     4.89  13824.     2.88
  25    625.     4.99  15625.     2.92
  26    676.     5.09  17576.     2.96
  27    729.     5.19  19683.     2.99
  28    784.     5.29  21952.     3.03
  29    841.     5.38  24389.     3.07
  30    900.     5.47  27000.     3.10
```

*The square root of 4 is not 1.99. Errors of this type are due to use of real number arithmetic. In the print operation there is no rounding off. All extra decimals are simply dropped (truncated). Adding .005 to each quantity before printing would perform the appropriate round-off.

G. SUBSCRIPTED VARIABLES

1. Subscripts are used to identify one value of a list of values, having the same name. In FORTRAN, a list is known as an array.

2. Subscripts are integral constants or variables greater than zero. They are written in parentheses immediately after the variable name.
   AGE(4), DATE(N)

3. Restrictions — when expressions are used only those listed are permissable.
   a. Variable plus or minus a constant.
      AGE(N+4)
   b. Constant multiplied by a variable.
      DATE(2*N)
   c. Constant multiplied by a variable plus or minus a constant.
      GRADE(2*N+1)

H. DIMENSION STATEMENT

1. DIMENSION statements are used to reserve storage space for an array of values. Any program which contains subscripts must also have a DIMENSION statement.

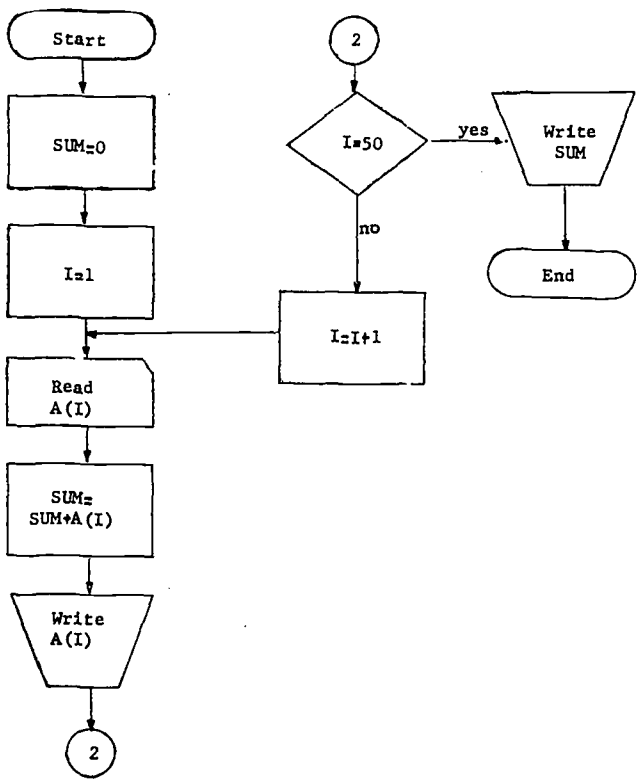
2. The word DIMENSION is followed by a list of variables of any mode separated by commas. Each variable is followed by a pair of parentheses which contains a number. The number designates the size of the array.
   DIMENSION AGE(10),DATE(10),NCODE(10)

3. The statement must be before any executable statements of the program.

I. PROBLEM USING SUBSCRIPTS

1. Problem — Read a list of 50 numbers, punched one to a card, and print the list and the sum of these numbers.

2. Flowchart



3. Coding Form

IBM FORTRAN Coding Form

```
C THIS PROGRAM WILL READ A LIST OF 50 NUMBERS, PUNCHED ONE TO A CARD.
C NUMBERS MUST BE LESS THAN THE ABSOLUTE VALUE OF 1000 TO THE NEAREST HUNDREDTH.
C THE NUMBERS ARE PRINTED AND THEIR SUM IS COMPUTED.
      DIMENSION A(50)
      SUM=0.
      DO 5 I=1,50
      READ(2,1) A(I)
    1 FORMAT(F7.2)
      SUM=SUM+A(I)
      WRITE(3,2)A(I)
    2 FORMAT(F9.2)
    5 CONTINUE
      WRITE(3,3)SUM
    3 FORMAT(2X,'SUM=',F9.2)
      CALL EXIT
      END
```

4. Printout

```
// FOR
*IOCS(CARD,DISK,1132PRINTER)
* LIST SOURCE PROGRAM
C THIS PROGRAM WILL READ A LIST OF 50 NUMBERS, PUNCHED ONE TO A CARD.
C NUMBERS MUST BE LESS THAN THE ABSOLUTE VALUE OF 1000 TO THE NEAREST HUNDREDTH.
C THE NUMBERS ARE PRINTED AND THEIR SUM IS COMPUTED.
      DIMENSION A(50)
      SUM=0.
      DO 5 I=1,50
      READ(2,1) A(I)
    1 FORMAT(F7.2)
      SUM=SUM+A(I)
      WRITE(3,2)A(I)
    2 FORMAT(F9.2)
    5 CONTINUE
      WRITE(3,3)SUM
    3 FORMAT(2X,'SUM=',F9.2)
      CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     106  PROGRAM      96

 END OF COMPILATION

// XEQ
```

38.00
39.00
48.00
20.00
7.00
-8.00
17.00
18.00
24.00
25.00
26.00
27.00
21.00
-22.00
23.00
30.00
31.00
32.00
-33.00
34.00
35.00
15.00
16.00
19.00
44.00
45.00
43.00
13.00
40.00
3.00
4.00
1.00
5.00
6.00
9.00
14.00
28.00
10.00
41.00
42.00
11.00
12.00
46.00
47.00
36.00
37.00
29.00
2.00
49.00
50.00
SUM= 1149.00

FORTRAN LANGUAGE

Unit IV

A. *LIBRARY FUNCTIONS*

1. The manufacturer provides a package of programs to solve some of the most often used mathematical functions, such as roots, logarithms, and trigonometric functions. These are called "library functions."

2. A name is assigned by the manufacturer to each library function. SQRT is the name for the function which computes the square-root.

3. The quantity which the computer needs to compute the desired result is called the argument.
   a. All library functions have one or more arguments.
   b. Arguments may be:
      (1) Variables
      (2) Constants
      (3) Legal arithmetic expressions, even those containing parentheses and other functions

4. The function statement consists of the name of the function followed by a pair of parentheses containing the argument or arguments.

5. The computer stops executing the main program while it computes the function. When the result is found the computer returns to the main program with this result, and continues the sequential execution of program steps.

6. Standard library for the IBM 1130. (see chart)

LIBRARY FUNCTIONS - IBM 1130

Function Name	Comments	Mode of Function	Mode of Argument	Example
SIN(A)	argument in radians	real	real	Y = SIN(A)
COS(A)	argument in radians	real	real	Y = A + COS(Z)
TANH(A) hyperbolic tangent	argument in radians	real	real	C = TANH(A*Z)
ATAN(A) angle whose tangent is A; $\tan^{-1}(A)$		real	real	ANGA = ATAN(TANA)
SQRT(A)	argument≥Ø	real	real	HYPOT = SQRT(A**2 + B**2)
EXP(A) exponential-e^A		real	real	POWER = EXP(Y)
ALOG(A) natural logarithm $\log_e A$	argument>Ø	real	real	Q = SIN(F) - ALOG(5.Ø)
ABS(A) absolute value		real	real	AROOT = SQRT(ABS(DISCR))
IABS(I)		integer	integer	IPOS = IABS(I)
FLOAT(I)		real	integer	REAL = FLOAT(INUM)
IFIX(A)		integer	real	INT = IFIX(REAL)

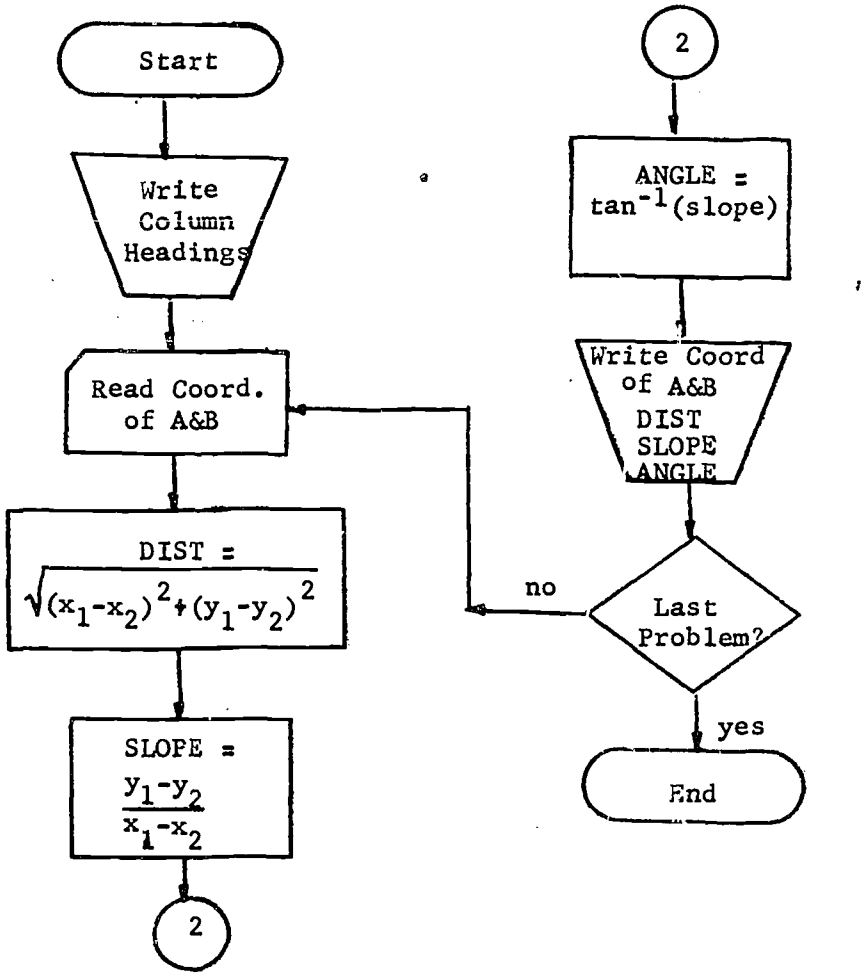
B. A PROBLEM USING LIBRARY FUNCTIONS

1. Problem

   Read the 4 coordinates of points A and B from a card. Find the distance between A & B, the slope of line AB, the angle that line AB makes with the X-axis, expressed in degrees.

   Print in column form, properly labeled, the given information, and the required solutions.

2. Flowchart



3. Coding Form

IBM FORTRAN Coding Form

FORTRAN STATEMENT

```
C THIS PROBLEM COMPUTES THE DISTANCE BETWEEN 2 POINTS WHOSE COORDINATES ARE
C KNOWN, THE SLOPE OF THE LINE THROUGH THESE POINTS, AND THE ANGLE THAT THE
C LINE MAKES WITH THE X-AXIS, EXPRESSED IN DEGREES. THE ABSOLUTE VALUE OF THE
C COORDINATES IS LESS THAN 100.
      WRITE (3, 1)
    1 FORMAT (8X,'A',15X,'B')
      WRITE (3, 2)
    2 FORMAT (4X,'X',6X,'Y',8X,'X',6X,'Y',6X,'LENGTH',3X,'SLOPE',4X,'ANG
     1LE')
   10 READ (2, 3) AX, AY, BX, BY
    3 FORMAT (4F6.2)
      DIST=SQRT((AX-BX)**2+(AY-BY)**2)
      SLOPE=(BY-AY)/(BX-AX)
      ANRAD=ATAN(SLOPE)
      ANGLE=ANRAD*180./3.14
      WRITE(3, 4)AX, AY, BX, BY, DIST, SLOPE, ANGLE
    4 FORMAT(2X,F6.2,1X,F6.2,3X,F6.2,1X,F6.2,3X,F6.2,3X,F6.2,3X,F6.2)
      GO TO 10
  999 CALL EXIT
      END
```

4. Printout

```
PAGE    1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0002        0002        0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
C THIS PRORLEM COMPUTES THE DISTANCE BETWEEN 2 POINTS WHOSE COORDINATES ARE
C KNOWN, THE SLOPE OF THE LINE THROUGH THESE POINTS, AND THE ANGLE THAT THE
C LINE MAKES WITH THE X-AXIS, EXPRESSED IN DEGREES. THE ABSOLUTE VALUE OF THE
C COORDINATES IS LESS THAN 100.
      WRITE (3,1)
    1 FORMAT (8X,'A',15X,'B')
      WRITE (3,2)
    2 FORMAT (4X,'X',6X,'Y',8X,'X',6X,'Y',6X,'LENGTH',3X,'SLOPE',4X,'ANG
     1LE')
   10 READ (2,3)AX,AY,BX,BY
    3 FORMAT (4F6.2)
      DIST=SQRT((AX-BX)**2+(AY-BY)**2)
      SLOPE=(BY-AY)/(BX-AX)
      ANRAD=ATAN(SLOPE)
      ANGLE=ANRAD*180./3.14
      WRITE(3,4)AX,AY,BX,BY,DIST,SLOPE,ANGLE
    4 FORMAT(2X,F6.2,1X,F6.2,3X,F6.2,1X,F6.2,3X,F6.2,3X,F6.2,3X,F6.2)
      GO TO 10
  999 CALL EXIT
      END

UNREFERENCED STATEMENTS
  999

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES      20  PROGRAM     176

 END OF COMPILATION

// XEQ

        A                B
    X       Y        X       Y      LENGTH   SLOPE    ANGLE
 -3.00   -4.00     3.00    4.00      10.00     1.33    53.15
  0.00    0.00    -5.00    7.00       8.60    -1.40   -54.48
```

C. ARITHMETIC STATEMENT FUNCTIONS

1. When no library function is available and the same process will be used several times within a program, an arithmetic statement function may be used.

2. The statement consists of: the function name, selected by the programmer, with its arguments, followed by an equal sign, and followed by the FORTRAN arithmetic expression, which defines the desired functional relationship.

   ANDEG(RAD) = RAD*180./3.1416

3. This type of statement must be placed before all executable steps of the program.

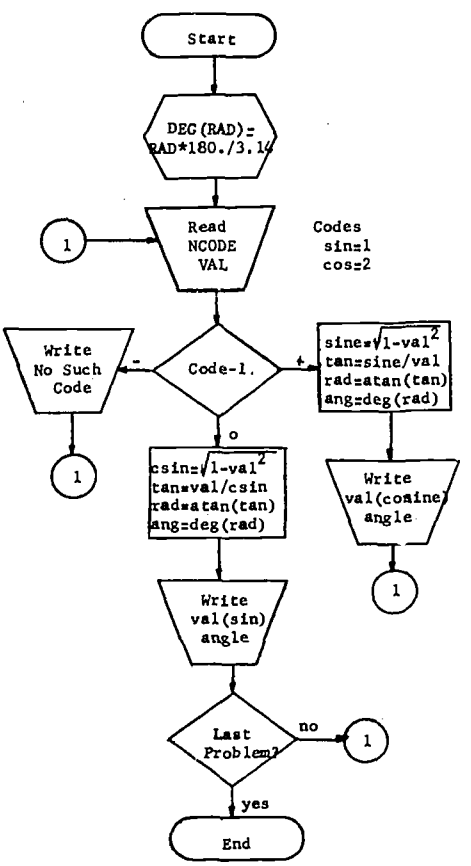
4. In the main program write the name of the function and its arguments in any valid arithmetic expression. The quantities used as arguments in the program will be used to compute the value of the function as prescribed by the function defining statement.

5. Restrictions
   a. Only functional relationships that can be computed in a single expression can be used.
   b. The expression on the right may contain unsubscripted variables, constants, or other functions.
   c. The function name contains 1 to 5 alphameric characters, the first a letter significant to the mode of the expression.

D. A PROBLEM USING ARITHMETIC STATEMENT FUNCTIONS

1. Problem: Write a program to find the number of degrees in an angle if either the sine or the cosine of the angle is known. Print out the given information and the number of degrees in the angle.

2. Flowchart



IBM FORTRAN Coding Form

```
C THIS PROGRAM WILL FIND THE NUMBER OF DEGREES IN AN ANGLE WHOSE SINE OR COSINE
C IS KNOWN. PUNCH A 1 IN COLUMN 2 IF THE SINE IS KNOWN, AND A 2 IF YOU KNOW THE
C COSINE.
      DEG(RAD)=RAD*180./3.14159
  100 READ(2,2)NCODE,VAL
    2 FORMAT(I2,F8.5)
      IF(NCODE-1)5,6,7
    5 WRITE(3,8)
    8 FORMAT(2X,'NO SUCH CODE')
      GO TO 100
    6 CSIN=SQRT(1.-VAL**2)
      TAN=VAL/CSIN
      RAD=ATAN(TAN)
      ANG=DEG(RAD)
      WRITE(3,9)VAL,ANG
    9 FORMAT(2X,'IF SIN A=',F8.5,1X,', THEN ANGLE A=',F8.2,1X,'DEGREES')
      GO TO 100
    7 IF(NCODE-2)6,10,5
   10 SINE=SQRT(1-VAL**2)
      TAN=SINE/VAL
      RAD=ATAN(TAN)
      ANG=DEG(RAD)
      WRITE(3,11)VAL,ANG
   11 FORMAT(2X,'IF COS A=',F8.5,1X,', THEN ANGLE A=',F8.2,1X,'DEGREES')
```

IBM FORTRAN Coding Form

```
      GO TO 100
  999 CALL EXIT
      END
```

4. Printout

```
PAGE    1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
   0000        0002        0002         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
C  THIS PROGRAM WILL FIND THE NUMBER OF DEGREES IN AN ANGLE WHOSE SINE OR COSINE
C  IS KNOWN. PUNCH A 1 IN COLUMN 2 IF THE SINE IS KNOWN, AND A 2 IF YOU KNOW THE
C  COSINE.
      DEG(RAD)=RAD*180./3.14159
  100 READ(2,2)NCODE,VAL
    2 FORMAT(I2,F8.5)
      IF(NCODE -1)5,6,7
    5 WRITE(3,8)
    8 FORMAT(2X,'NO SUCH CODE')
      GO TO 100
    6 CSIN =SQRT(1.-VAL**2)
      TAN=VAL/CSIN
      RAD=ATAN(TAN)
      ANG=DEG(RAD)
      WRITE(3,9)VAL,ANG
    9 FORMAT(2X,'IF SIN A=',F8.5,1X,', THEN ANGLE A=',F8.2,1X,'DEGREES')
      GO TO 100
    7 IF(NCODE-2)6,10,5
   10 SINE=SQRT(1-VAL**2)
      TAN=SINE/VAL
      RAD=ATAN(TAN)
      ANG=DEG(RAD)
      WRITE(3,11)VAL,ANG
   11 FORMAT(2X,'IF COS A=',F8.5,1X,', THEN ANGLE A=',F8.2,1X,'DEGREES')
      GO TO 100
  999 CALL EXIT
      END

UNREFERENCED STATEMENTS
 999

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     18  PROGRAM     228

 END OF COMPILATION

// XEQ

 IF SIN A= 0.50000 , THEN ANGLE A=    30.00 DEGREES
 IF COS A= 0.86602 , THEN ANGLE A=    30.00 DEGREES
 NO SUCH CODE
 IF COS A= 0.86162 , THEN ANGLE A=    30.50 DEGREES
 NO SUCH CODE
 IF SIN A= 0.93358 , THEN ANGLE A=    68.99 DEGREES
```

E. *COMPUTED GO TO*

1. The statement consists of the words GO TO followed by an open parenthesis containing any required number of statement numbers separated by commas, a closed parenthesis, a comma, and a non-subscripted integer variable.

   GO TO (10,8,7),K

2. If the integer variable has a value of 1, the next statement to be executed is the 1st statement in the parenthesized list, if the variable is 2, the 2nd statement is executed, and so on. The value of the integer is defined or computed in another part of the program and acts as a subscript to the list. The length of the list

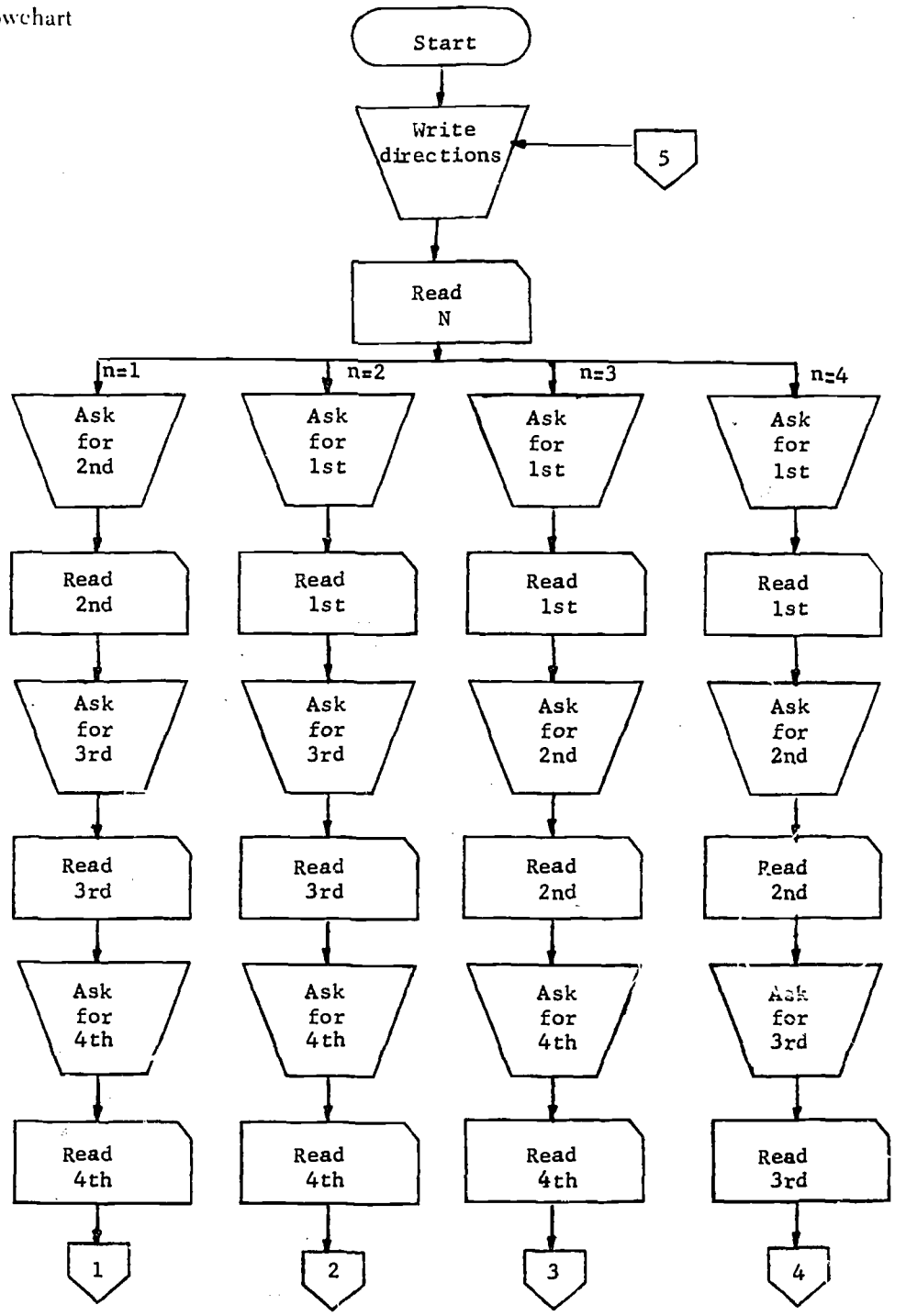
and the repeated appearance of the numbers of the list are not restricted, but the control variable may not be larger than the number of statements in the list.

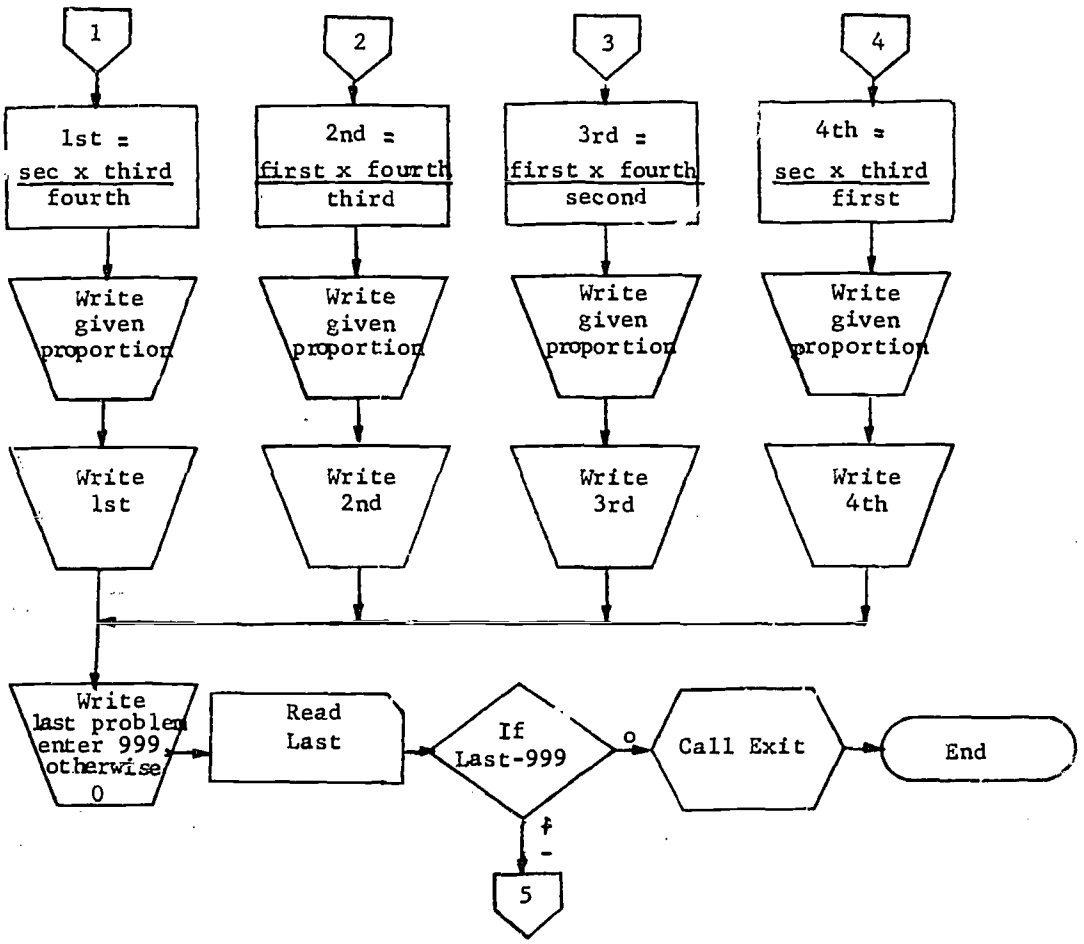
3. Whenever it is posible to select one of several parts of a program dependent on the value of a control variable, a computed GO TO may be used.

F. *PROBLEM USING A COMPUTED GO TO*

1. Problem — Write a program to compute any missing term of a proportion if three others are known. The known terms are to be entered on the typewriter after the operator types in the number of the proportional that is unknown.

2. Flowchart





3. Printout

```
PAGE    1
// JOB
LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0002         0002         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
  100 WRITE(1,1)
    1 FORMAT(2X,'THIS PROGRAM IS DESIGNED TO FIND THE UNKNOWN TERM OF A
     1PROPORTION WHEN THREE TERMS ARE KNOWN')
      WRITE(1,2)
    2 FORMAT(2X,'IF YOUR PROPORTION WERE WRITTEN IN THE FORM',//,2X,'FIR
     1ST IS TO SECOND = THIRD IS TO FOURTH',//,2X,'ENTER THE NUMBER OF T
     2HE MISSING TERM',//)
      READ(6,3)N
    3 FORMAT(I1)
      GO TO (10,20,30,40),N
   10 WRITE(1,4)
    4 FORMAT(2X,'ENTER THE 2ND TERM IN THE FORM XXXX.XX')
      READ(6,5)SEC
    5 FORMAT(F8.2)
      WRITE(1,6)
    6 FORMAT(2X,'ENTER THE 3RD TERM IN THE FORM XXXX.XX')
      READ(6,5)THIRD
      WRITE(1,7)
    7 FORMAT(2X,'ENTER THE 4TH TERM IN THE FORM XXXX.XX')
      READ(6,5)FORTH
      FIRST=SEC*THIRD/FORTH
      WRITE(1,25)SEC,THIRD,FORTH
   25 FORMAT(2X,'X  IS TO',F8.2,'=',F8.2,'IS TO'F8.2)
      WRITE(1,8)N,FIRST
    8 FORMAT(2X,'THE',I3,1X,'PROPORTIONAL IS',F9.2)
      GO TO 200
   20 WRITE(1,9)
    9 FORMAT(2X,'ENTER THE 1ST TERM IN THE FORM XXXX.XX')
      READ(6,5)FIRST
      WRITE(1,6)
      READ(6,5)THIRD
      WRITE(1,7)
```

```
      READ(6,5)FORTH
      SEC=FIRST*FORTH/THIRD
      WRITE(1,26)FIRST,THIRD,FORTH
   26 FORMAT(2X,F8.2,'IS TO  X  =',F8.2,'IS TO',F8.2)
      WRITE(1,8)N,SEC
      GO TO 200
   30 WRITE(1,9)
      READ(6,5)FIRST
      WRITE(1,4)
      READ(6,5)SEC
      WRITE(1,7)
      READ(6,5)FORTH
      THIRD=FIRST*FORTH/SEC
      WRITE(1,27)FIRST,SEC,FORTH
   27 FORMAT(2X,F8.2,'  IS TO ',F8.2,'  = X  IS TO',F8.2)
      WRITE(1,8)N,THIRD
      GO TO 200
   40 WRITE(1,9)
      READ(6,5)FIRST
      WRITE(1,4)
      READ(6,5)SEC
      WRITE(1,6)
      READ(6,5)THIRD
      FORTH=SEC*THIRD/FIRST
      WRITE(1,28)FIRST,SEC,THIRD
   28 FORMAT(2X,F8.2,'  IS TO ',F8.2,'  = ',F8.2,'IS TO  X')
      WRITE(1,8)N,FORTH
  200 WRITE(1,11)
   11 FORMAT(2X,'IF THIS IS YOUR LAST PROBLEM ENTER 999, IF NOT ENTER 0'
     1)
      READ(6,12)LAST
   12 FORMAT(I3)
      IF (LAST-999)100,300,100
  300 CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     12  PROGRAM    616

 END OF COMPILATION

// XEQ


 THIS PROGRAM IS DESIGNED TO FIND THE UNKNOWN TERM OF A PROPORTION WHEN THREE TERMS ARE KNOWN
 IF YOUR PROPORTION WERE WRITTEN IN THE FORM

 FIRST IS TO SECOND = THIRD IS TO FOURTH

 ENTER THE NUMBER OF THE MISSING TERM

1
 ENTER THE 2ND TERM IN THE FORM XXXX.XX
 25.
 ENTER THE 3RD TERM IN THE FORM XXXX.XX
100.
 ENTER THE 4TH TERM IN THE FORM XXXX.XX
125.
 X  IS TO   25.00=  100.00IS TO  125.00
 THE  1 PROPORTIONAL IS    20.00
 IF THIS IS YOUR LAST PROBLEM ENTER 999, IF NOT ENTER 0
4
 THIS PROGRAM IS DESIGNED TO FIND THE UNKNOWN TERM OF A PROPORTION WHEN THREE TERMS ARE KNOWN
 IF YOUR PROPORTION WERE WRITTEN IN THE FORM

 FIRST IS TO SECOND = THIRD IS TO FOURTH

 ENTER THE NUMBER OF THE MISSING TERM

4
 ENTER THE 1ST TERM IN THE FORM XXXX.XX
 25.
 ENTER THE 2ND TERM IN THE FORM XXXX.XX
100.
 ENTER THE 3RD TERM IN THE FORM XXXX.XX
125.
    25.00  IS TO   100.00  =   125.00IS TO  X
 THE  4 PROPORTIONAL IS   500.00
 IF THIS IS YOUR LAST PROBLEM ENTER 999, IF NOT ENTER 0
999
```

FORTRAN LANGUAGE

UNIT V

A. *READ STATEMENT*

1. In order to have the computer receive information from an input device a READ statement must be used. It tells the computer what to read.

2. The statement consists of the word READ, followed by a pair of parentheses containing 2 numerical codes separated by a comma, and a list of the variables separated by commas.

   READ (n_1,n_2) A,B,I,C(J)

   a. Input Codes

Logical Unit Code (n_1)	*Device*
2	Card Reader
4	Paper Tape Reader
6	Keyboard

   An integer variable, defined previously in the program may be used instead of the numerical code.

   b. Variables in Input List
      (1) The length of the list is not restricted.
      (2) The variables may be of any mode, and modes may be mixed in the same list.
      (3) Variables may or may not be subscripted as required by the specific program.
      (4) An integer read early in the list may be used as a subscript later in the list.

```
READ (2,1) I,A(I)
```

      However, subscripts do not necessarily have to be defined in this manner. Alternate methods are:
      (a) The subscript may be defined by index of a loop.

```
      DO 10 I=1,10
      READ (2,20)A(I)
10    CONTINUE
```

      (b) The subscript may be read in by a separate READ statement.

```
      READ (4,1)N
      DO 25 I=1,N
      READ (2,20)A(I)
25    CONTINUE
```

3. Self-indexed READ
   a. This type of statement is used to read arrays.
   b. The variables and their variable subscripts must be followed by a comma and an index definition like that of a DO loop.

```
READ (6,1) (A(I),I=1,25)
```

   c. The variable and index definition must be enclosed in parentheses.
   d. As is the case of the DO loop, the limits of the index may be variables or constants.
   e. The list of elements may contain other elements besides those that are self-indexed, either before or after the self-indexed element.

```
READ (2,1)I,J,K,(A(I),I=J,K),SUM,BIGA
```

   f. The input list may contain more than one self-indexed element.

```
READ (4,1) (A(I),I=1,10),(B(J),J=1,20)
```

   g. More than one subscripted variable may use the same index definition.

```
READ (6,1)(A(I),B(I),I=1,20)
```

   h. The disadvantages of the DO loop control of reading data as compared to the self-indexing method are:
      (1) The DO loop must execute the READ as many times as the upper limit of the loop.

(2) Under control of a DO loop, each time the READ is executed a *new card* must be read by the computer, thereby wasting cards.

°°°°°°°°°°

EXERCISES

1. Write a READ statement to read 2 variables named COST, SELPR from cards.
   ANSWER: READ (2,1)COST,SELPR

2. Write a self-indexed READ statement to read 20 values of A,B,C from 20 cards, each punched with A,B,C.
   ANSWER: READ(2,1) (A(I),B(I),C(I),I=1,20)

3. Write a READ statement to read in an integer variable called N and N variables called COST(I) from cards.
   ANSWER: READ(2,1)N,(COST(I),I=1,N)

4. Write a READ statement that will use the keyboard or card reader to define 3 variables called X,Y and Z, depending on the value of a variable called KRC.
   ANSWER: READ(KRC,1)X,Y,Z

5. Write a READ statement to read 100 pairs of X and Y coordinates of points from cards.
   ANSWER: READ(2,1) (X(I),Y(I),I=1,100)

6. Write a READ statement to read 100 values of A, then 100 values of B from cards.
   ANSWER: READ (2,1) (A(I),I=1,100), (B(I),I=1,100)

7. Indicate whether or not the following READ statements are valid, and if they are invalid write them correctly.
   a. READ (2,1) COST
   b. READ (-2,1) SELPR
   c. N=6
      READ (N,1) COST
   d. READ (2,1) NUMBR,COST,MRKUP,SELPR,TAX
   e. READ (2,1) (COST(I), I=1,20)
   f. READ (2,1) J,COST(I),I=1,20
   g. READ (2,1)COST(2*I+1),SELPR(J)
   h. READ (2,1) N,A(N)
   i. READ (2,1) N,(A(I),I=1,N)
   j. READ (2,1) L,M,(A(I)I=L,M)
   k. READ (2,1) A,B,(C(I),I=1,B),K2R,STP
   l. READ (A(I), B(I),C(I),I=1,10)

   ANSWERS
   a. Valid
   b. READ (2,1) SELPR
   c. Valid
   d. Valid
   e. Valid
   f. READ (2,1) J, (COST(I),I=1,20)
   g. Valid
   h. Valid
   i. Valid
   j. READ (2,1) L,M,(A(I),I=L,M)
   k. READ (2,1) A,IB,(C(I), I=1,IB),K2R,STP
   l. READ (2,1) (A(I),B(I),C(I), I=1,10)

°°°°°°°°°°

B. FORMAT STATEMENTS

1. A statement that provides information to the input or output statements about where numbers are to be found is a FORMAT statement. It is not executed and because of this fact does not necessarily have to be placed in sequential order. Ordinarily they are placed adjacent to the READ or WRITE statements or all together at the beginning of the program.

2. Every FORMAT statement must have a statement number.

3. The statement consists of the statement number, the word FORMAT, followed by a pair of parentheses containing FORMAT codes.
   a. FORMAT Conversion Codes
      (1) The FORMAT codes tell the field width of the number, the number of decimal places, if any, and its mode.
      (2) FORMAT codes combine both alphabetic and numeric characters.
          (F8.2)
          (a) Alphabetic
              F - Real numbers, decimal notation
              I - Integers
              E - Real numbers, exponential notation
              X - Blanks
              H - Hollerith characters
              A - Alphameric characters
          (b) Numeric
              1st integer - the number of spaces (card columns, print spaces) used to contain the number (field-width). When punched or typed the number must be right adjusted in the field. (See Data Card examples on pages 8 and 9.)
              2nd integer - the number of decimal places. (The computer will ignore this code if a decimal is punched in any position in the input.)
   b. READ Codes
      (1) REAL numbers with ordinary decimal notation require F conversion code, field width specification, and decimal specification.
          (F8.2)
      (2) Real numbers may be coded in E notation to reduce the amount of keypunching when very large or very small numbers are needed in a program.
          (a) A great deal of variation is allowed in reading E notation. Ordinarily the field width required must include a place for the E, one for the sign of the exponent, one or two for the exponent, one for the decimal point if any, or a total of four or five spaces more than will be occupied by the actual number of significant digits in the number. For example: .000072 could be written as 7.2E-5 or 7.2E-05 requiring a field-width of 6 or 7.
          (b) The construction of the E FORMAT conversion code corresponds closely with that of the F code. It consists of the alphabetic code, field width specification, period, decimal-specification. The code for reading the numbers above would be E6.1 or E7.1.
   c. I Code
      (1) The I code is used for converting integers.
      (2) The letter I is followed by an integer which specifies field width.
      (3) No decimal code is needed because integers never contain a decimal fraction.
   d. X Code
      (1) The X code may be used to indicate the number of spaces to be skipped.
      (2) The X is preceded by a positive integer indicating the number of spaces to be skipped.
          5X - do not read these 5 spaces
   e. H Code
      (1) H code is not used as much by the IBM 1130 as it is by some othe machines because of its special feature of copying material between apostrophes. However, it will be accepted. The use of H notation indicates that a number of spaces following the letter H will be occupied by Hollerith

symbols (combination of alphabetic and numeric information) to be copied literally.

(2) A positive integer preceding the H indicates the number of Hollerith characters that follow.

(14H1234 SOUTH ST.)

f. A Code

(1) A code is used to read alphanumeric information.

(2) No decimal specification is required.

(3) The field width of an A specification code must not exceed 4 spaces because alphanumeric information requires more storage space than ordinary numeric information. Extended precision must be used to increase the number of alphanumeric characters being read as one variable.

g. The parentheses of the FORMAT statement contain the complete specification for one card or one line. One number-conversion code indicates one number punched per card, or typed per line.

h. More than one code may be used in one statement. No limit is set on the number and variety of such codes as long as the limit of the card or print line is not exceeded.

12 FORMAT(F8.2,I3,E10.3,F2.0)

i. An integer may be placed before the conversion code to indicate how many times a code will be repeated in the layout.

11 FORMAT(3F8.2,2A3,4I2)

°°°°°°°°°°°

EXERCISES

1. Write a FORMAT statement to convert a real number having 4 decimal places from the first 10 columns of a card.

ANSWER: 1 FORMAT(F10.4)

2. In the FORMAT statement 2 FORMAT(F8.2), a) what does the F specify, b) the 8, c) the 2?

ANSWER: a) a real number, b) the field width, c) the decimal point position.

3. Write a FORMAT statement to convert 3 real numbers from one card, all having the same field width and decimal position.

ANSWER: 3 FORMAT(3F10.4)

4. Write a FORMAT statement to convert 3 real numbers from one card, all having different field widths and decimal positions.

ANSWER: 3 FORMAT(F10.4,F8.2,F7.1)

5. Write a FORMAT statement to convert 2 integers, 3 real numbers punched in regular decimal notation, and 2 in exponential notation from a card.

ANSWER: 4 FORMAT(2I5,3F10.2,2E20.6)

°°°°°°°°°°°

C. *COMPLETE INPUT OPERATION*

1. The complete input operation is defined by the READ statement working with a FORMAT.

2. Spaces that are not specified in the FORMAT codes are not read.

```
READ(2,1)A,B,C
1 FORMAT(3F10.2)
```

The sequence above only reads the 1st 30 card columns. Anything punched in columns 31 through 80 will not be read.

3. If the input list contains fewer variables than the FORMAT conversion code, only as many as are contained in the list will be read.

```
      READ(2,1)A,B,C
    1 FORMAT(7F8.2)
```
The sequence above will read only 3 variables.

4. If the number of variables is more than can be read from one card the computer will read as many values as are specified in the list.
```
      READ(2,2)(AGE(J),J=1,100)
    2 FORMAT(26F3.0)
```
The preceding sequence will read 100 values of AGE from 4 cards, three cards contain 26 values, the fourth contains 22 values.

5. The actual mode of conversion of the number from the form in which it is found on the card to the form which it will take inside the computer is determined by the FORMAT, regardless of whether or not the programmer has written them to agree. Lack of care in this instance could cause an error to exist. For example the sequence
```
      READ(2,5)I,J
    5 FORMAT(2F7.2)
```
would cause real numbers to be assigned to the integer variables I and J.

6. Separate Card Layouts
   a. Separate card layouts are indicated by the presence of a slash (/) in the FORMAT statement. For example the statement 6 FORMAT(5I3/6F8.2) could be used to read 5 integers from one card and 6 real numbers from another card. If the associated READ called for more than 11 variables to be read into storage the third card would contain 5 integers, the fourth would contain 6 real numbers, and so on until the input list is satisfied.
   b. Use of Parentheses
      (1) Parentheses can be used in conjunction with the slash to cause a single card to be read with one layout and many cards read from a second layout. For example the sequence
```
      READ(2,5)N,(A(I),I=1,100)
    5 FORMAT(I3/(8F10.2) )
```
      would cause the computer to read one value from the first card, and 8 values from each of the next 12 cards, and 4 from the fourteenth card.
      (2) Parentheses within parentheses after the slash are not permitted in this type of FORMAT statement.
      (3) Specifications for particular card columns may be repeated by the use of parentheses. For example the sequence
```
      READ(2,6)N,J,COST,I,L,SELLP
    6 FORMAT(2(2I3,F10.2) )
```
      would read 6 values from a card, reading the sequence (2I3,F10.2) twice as indicated by the integer preceding the parentheses.

°°°°°°°°°°°

EXERCISES

Explain what the following pairs of input statements will cause the computer to do.

1.
```
      READ(2,10)A,B,C
   10 FORMAT(3F7.2)
```

2.
```
      READ(2,10)(PRICE(N),N=1,20)
   10 FORMAT(8F10.2)
```

3.
```
      READ(2,10)STP
   10 FORMAT(E10.3)
```

4.
```
      READ(2,10)(ALPHA(I),I=1,4)
   10 FORMAT(4A1)
```

5.
```
      READ(6,10)K,(A(I),I=1,K)
   10 FORMAT(I4/(8F10.4) )
```

ANSWERS

1. 3 real numbers which define A,B, and C will be read from 21 card columns. Each number has 2 decimal places.

2. Real numbers which define a list of 20 variables called PRICE are read from cards. Each card contains 8 numbers and each number is allotted 10 card columns, and has 2 decimal places.

3. One real number in exponential notation is read from a 10 card column. If no decimal is punched on the card the FORMAT will provide 3 decimal places.

4. Reads 4 alphabetic characters from a card.

5. Reads a variable number of cards. The 1st card contains the integer variable which defines the limit of the loop. Each card following is punched with 8 real numbers. The statement will cause K variables to be read from as many cards as are needed to contain these variables if 8 are punched to 1 card.

°°°°°°°°°°°

D. *WRITE STATEMENTS*

1. In order to have the computer to output its solutions a WRITE statement must be used. The output statements are similar to the input or READ statement.

2. The statement consists of the word WRITE, followed by a pair of parentheses containing 2 numerical codes separated by commas, and a list of variables separated by commas.

   WRITE(n_1,n_2)A,B,C,J

   a. Output Codes

Logical Unit Code (n_1)	*Device*
1	Typewriter
2	Card Punch
3	Printer
4	Paper Tape Punch

   Notice that 2 and 4 are also used as input codes. However, used in conjunction with a READ they are considered input, and with a WRITE they are considered output. As is the case in READ statements, an integer variable, pre-defined in the program can be used for a logical unit number.

   b. Variables in Output List

      (1) All of the variations allowed for input lists are also valid for output lists.
         (a) Variables may be of either mode in the same list.
         (b) The self-indexed form may be used.
         (c) Subscripted variables may be used, unless the subscripts are mathematical expressions.

E. *FORMAT STATEMENTS*

1. Output FORMAT statements are similar to those used for input. They specify the layout of the solutions on the output device, and conversion codes are identical to those used with the READ.

2. The FORMAT statement used with a WRITE statement defines the layout of a single line on the printer or typewriter, or a single card to be punched. This line will be repeated until the variable list is satisfied.

a. Printer (or typewriter) lines consist of 120 print spaces; therefore the FORMAT specifications for the single unit cannot exceed 120 spaces.
b. The card-punch specifications cannot exceed 80 spaces per card.

3. Real Numbers
   a. The decimal point is printed.
   b. The programmer must determine the number of significant digits to be printed and construct the FORMAT accordingly. Do not call for more digits than are significant.
   c. Numbers are printed so that they are adjusted to the right of the alloted field and blank spaces occupy unused positions on the left. Spacing may therefore be provided by indicating a field width wider than is actually needed.
   d. Unused decimal places will be filled with zeros. 75.2 with F5.2 specification will print as 75.20.
   e. A place must be reserved for printing the sign of negative numbers. Positive numbers are printed without the sign. For example: -52.74 would require an F6.2 specification.
   f. If the number of decimal places in storage exceeds the FORMAT specification, the decimal will be truncated as prescribed by the FORMAT; for example: 25.7259 will be printed as b25.72 if an F 6.2 specification is used, b signifying a blank space.
   g. If the number of digits before the decimal exceeds the alloted space, the entire field will be filled with asterisks.
   h. If there are no digits preceding the decimal a zero will be printed, using one space. For example .75 will be printed as 0.75 requiring an F4.2 specification.
   i. Real numbers may be printed using the E conversion code.
      (1) For E output, numbers are normalized, that is, the decimal always appears to the left of the 1st significant digit and a zero precedes the decimal point.
      (2) The exponent correctly places the decimal point, the computer making the necessary adjustments to achieve this.
      (3) The number of decimals to be printed depends upon the decimal specification in the E code which is selected by the programmer. 79.234 with conversion code E12.4 would be printed as bb0.7923Eb02.
      (4) E conversion codes require at least seven spaces in addition to the number of significant digits which will be printed, four for the exponent, one for the decimal, one for the leading zero and one for the sign.
      (5) Extra spaces can be added to the field width to improve readability and appearance of output.
      (6) A FORMAT specification of E20.7 will provide sufficient space for any internal number with 6 leading spaces, and is therefore frequently used.

4. Integers
   a. I conversion code specifications are like those for input statements.
   b. No decimal point is printed.
   c. Numbers are right-adjusted in the allotted fields.

5. Slashes
   a. Slashes are used to signify the end of the specification for one line.
   b. More than one / in the output FORMAT causes multiple spacing, that is // would cause a line to be skipped between 2 lines of printing, and /// would cause 2 lines to be skipped.
   c. Slashes appearing before or after a FORMAT conversion code would cause a number of blank lines equal to the number of slashes before or after the printed line.

6. Literal Data
   a. Remarks, descriptions, labels or any type of alphabetic information may be printed by enclosing the information in apostrophes within the FORMAT statement.
   b. The computer will copy whatever is enclosed in single quotes as it is written by the programmer. The enclosed characters may be alphabetic, numeric, or special characters.
   c. No variable is listed in the output statement for literal data.
   d. Literal data may be printed before or after E, F, or I conversion codes.
   e. The number of spaces used in the specification for a line must not exceed 120.

f. To double space after a printed line slash marks (//) should follow the literal data.
g. Literal data can be read into storage with a similar FORMAT and printed out in the same form as it is read in, but it may not be altered in any way if the 'quote' method is used.

7. A Conversion
   a. To alter alphanumeric data A conversion code must be used.
   b. The code consists of the letter A followed by an integer which indicates the number of alphanumeric characters to be written under A conversion.
   c. A corresponding variable name must appear in the variable list for each alphanumeric group.
   d. The alphanumeric groups cannot contain more than 4 A characters if its name is a real variable or 2 characters if it is an integer variable. To read the 2 initials and an 8 character last name the following sequence could be used.

```
READ(2,1)INIT1,INIT2,SNAM1,SNAM2
1 FORMAT(2A2,2A4)
```

   The card would be punched with A.J.WILLIAMS in columns one through twelve.

8. X Conversion
   a. To provide spacing of output on a given line or card X conversion can be used.
   b. It consists of the letter X, preceded by an integer indicating the number of blanks to be defined.

9. Carriage Control
   a. When output is to be produced on the printer, special specifications must be included in the FORMAT statements to provide for control of the carriage of the printer.
   b. The 1st character of the output image is not printed but serves as a message to the computer to cause the printer to space down the page in various ways.
   c. Codes
      (1) If the following codes appear in the first space of the line to be printed, the corresponding carriage control operation will take place.

Code	*Control Operation*
b (blank)	normal spacing
0 (zero)	double space
1 (one)	begin on a new page
+ (plus)	write on same line

      (2) The carriage control codes apply only to printed output, not output on cards, tape or disk.

. . . . . .

EXERCISES

1. Revise the program in Unit III which finds the sum of 50 numbers read from cards. Assume that the numbers are punched 8 to a card with 10 field width and 2 decimal places. Use self-indexed READ and WRITE statements.

SOLUTION

```
PAGE    1
// JOB
LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0002         0002         0000
// FOR
*IOCS(CARD,DISK,1132PRINTER)
* LIST SOURCE PROGRAM
C THIS PROGRAM WILL READ A LIST OF 50 NUMBERS, PUNCHED ONE TO A CARD.
C NUMBERS MUST BE LESS THAN THE ABSOLUTE VALUE OF 1000 TO THE NEAREST HUNDREDTH.
C THE NUMBERS ARE PRINTED AND THEIR SUM IS COMPUTED.
```

```
      DIMENSION A(50)
      SUM=0.
      READ(2,1)(A(I),I=1,50)
    1 FORMAT(8F10.2)
      DO 5 I=1,50
      SUM=SUM+A(I)
    5 CONTINUE
      WRITE(3,1)(A(I),I=1,50)
      WRITE(3,3)SUM
    3 FORMAT(2X,'SUM=',F9.2)
      CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     106  PROGRAM     124

 END OF COMPILATION

// XEQ

   -33.00      7.50     36.00     17.00     38.00     49.52    -19.00     20.00
    50.00     35.00     10.00     47.00      2.00     18.00      5.00     29.00
    43.00     45.00    -44.00      0.01     12.00     13.00     26.00     40.00
     7.00      9.00     46.00     24.00     48.00     39.00     30.00     15.00
    44.00     42.00     23.00     -1.25     25.33     31.00     14.00     27.00
    34.00     22.00     41.00     37.00     31.00      4.99     28.00    -16.00
    -8.01      6.00
 SUM=  1050.09
```

2. Write a FORMAT statement to print CAPACITY = 15,791.54 GALLONS.
   ANSWER: 20 FORMAT(2X,'CAPACITY = 15,791.54 GALLONS')

3. Write a set of statements to print the following:
   SOLUTIONS TO QUADRATIC EQUATIONS
   Triple Space
   Print Column headings
   A B C X1 X2
   Double Space
   Print coefficients and answers of 10 problems under headings, each having 2 decimal places.
   ANSWER:

```
      WRITE (3,10)
   10 FORMAT (43X,'SOLUTIONS OF QUADRATIC EQUATIONS',///,41X,'A',6X,'B'
     1,6X,'C',15X,'X1',6X,'X2',//)
      WRITE (3,11)A,B,C,X1,X2
   11 FORMAT (36X,3F7.2,10X,2F7.2)
```

FORTRAN LANGUAGE

Unit VI

A. *MORE ABOUT DO-LOOPS*

1. Range — All of the statements from the DO statement to the statement whose statement number appears after the word DO are in the 'range' of the DO-loop.

2. A control statement (IF or GO TO) may never be used to end a loop. If the branch is necessary, it must transfer to the dummy statement, CONTINUE, which must end the loop that would ordinarily end with a control statement.

3. A 'special exit' may cause the computer to leave the loop before the upper index limit has been reached. In this case the index value at the time of the exit may be retained for future use in the program. Transfers from within the loop may be made to statements outside its range.

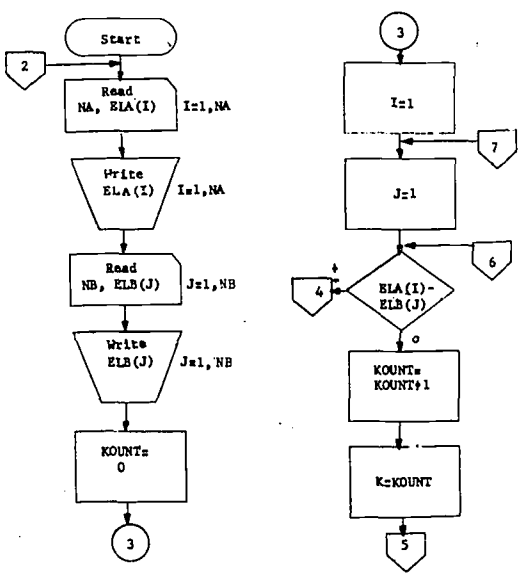
4. Transfers from statements outside the range of the loop cannot be made to statements within the range.

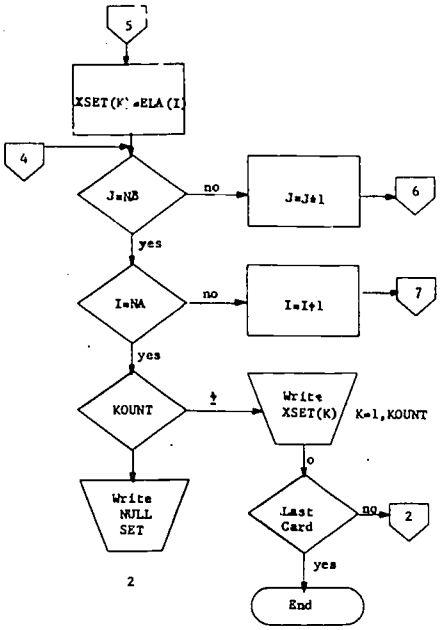
5. Nested DO-Loops
   a. Definition. DO-loops are nested if one is completely within the range of another loop, but the loops must not overlap in any way.
   b. Execution. The outer loop begins first and executes until the beginning of the inner loop. The inner loop completes its cycle, then the outer continues to its end, and begins again. The process continues until the upper limit of the outer loop is reached.
   c. Two or three loops within loops are often used.
   d. Loops may be used end to end within other loops.
   e. Two or more loops may end at the same statement.

B. *A PROBLEM USING NESTED DO-LOOPS*

1. Problem — Read alphabetic elements of sets A and B from cards. An integer representing the number of elements in the set occupies the first 2 card columns. Write the given sets and the intersection set, providing for a message to indicate the null set.

2. Flowchart





3. Coding Forms

IBM FORTRAN Coding Form

INTERSECTION OF SETS (XSET) 1 2

FORTRAN STATEMENT

```
      DIMENSION ELA(26),ELB(26),XSET(26)
300   READ(2,1)NA,(ELA(I),I=1,NA)
  1   FORMAT(I2,26A2)
      WRITE(3,5)(ELA(I),I=1,NA)
  5   FORMAT(2X,26A2)
      READ(2,2)NB,(ELB(J),J=1,NB)
  2   FORMAT(I2,26A2)
      WRITE(3,5)(ELB(J),J=1,NB)
      KOUNT=0
      DO 10 I=1,NA
      DO 10 J=1,NB
      IF(ELA(I)-ELB(J))10,20,10
 20   KOUNT=KOUNT+1
      K=KOUNT
      XSET(K)=ELA(I)
 10   CONTINUE
      IF(KOUNT)40,35,40
 35   WRITE(3,3)
  3   FORMAT(2X,'THE INTERSECTION SET IS THE NULL SET.')
      GO TO 300
 40   WRITE(3,4)(XSET(K),K=1,KOUNT)
  4   FORMAT(2X,'INTERSECTION SET =',26A3)
      GO TO 300
999   CALL EXIT
```

IBM FORTRAN Coding Form

2 2

FORTRAN STATEMENT

```
      END
```

4. Printout

```
PAGE    1

// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0002         0002         0000

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
      DIMENSION ELA(26),ELB(26),XSET(26)
  300 READ(2,1)NA,(ELA(I),I=1,NA)
    1 FORMAT(I2,26A2)
      WRITE(3,5)(ELA(I),I=1,NA)
    5 FORMAT(2X,26A2)
      READ(2,2)NB,(ELB(J),J=1,NB)
    2 FORMAT(I2,26A2)
      WRITE(3,5)(ELB(J),J=1,NB)
      KOUNT=0
      DO 10 I=1,NA,
      DO 10 J=1,NB
      IF(ELA(I)-ELB(J))10,20,10
   20 KOUNT=KOUNT+1
      K=KOUNT
      XSET(K)=ELA(I)
   10 CONTINUE
      IF(KOUNT)40,35,40
   35 WRITE(3,3)
    3 FORMAT(2X,'THE INTERSECTION SET IS THE NULL SET')
      GO TO 300
   40 WRITE(3,4)(XSET(K),K=1,KOUNT)
    4 FORMAT(2X,'INTERSECTION SET =',26A3)
      GO TO 300
  999 CALL EXIT
      END

UNREFERENCED STATEMENTS
 999

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     170  PROGRAM     288

 END OF COMPILATION

// XEQ

  A B C D E F G H I J
  B D E G I
 INTERSECTION SET = B  D  E  G  I
  A E G H D
  F I T
 THE INTERSECTION SET IS THE NULL SET
  A E T N R
  A T O P S
 INTERSECTION SET = A  T
```

C. *TYPE STATEMENTS*

1. The REAL statement consists of the word REAL followed by a list of variable names all of which will specify real numbers regardless of the ordinarily significant name of the variable.
   For example the statement:
   REAL IDENT, NUMBR
   would specify that IDENT and NUMBR are to be handled as real numbers.

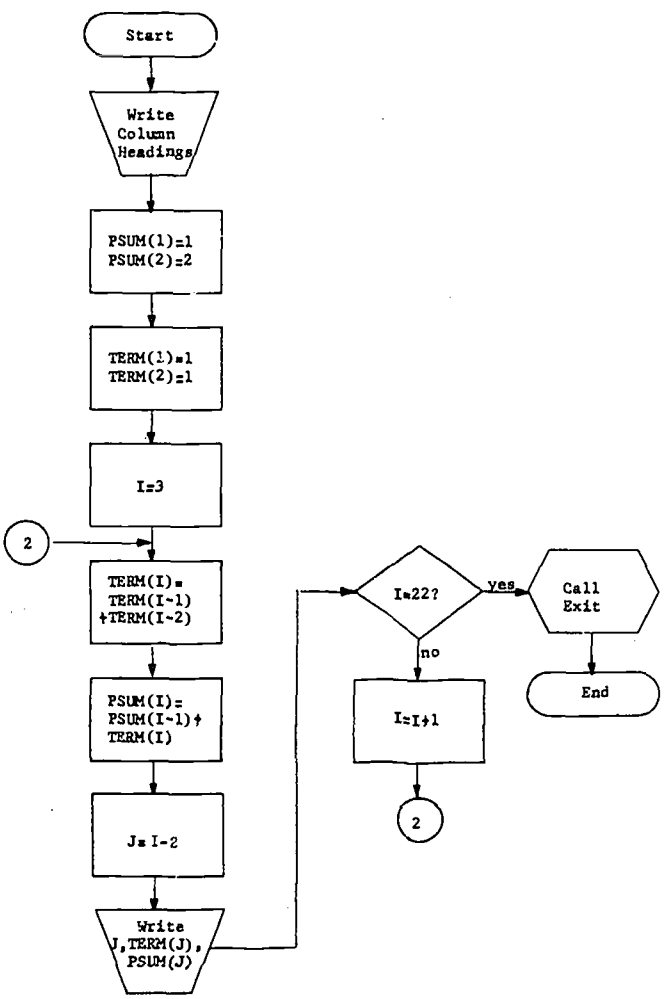
2. The INTEGER statement consists of the word INTEGER, followed by a list of variable names which will be considered as integers by the program.
   INTEGER BLANK, DOT, POLY, SUM
   All the listed variables will be integral.

3. The REAL or INTEGER statement must be the first statement of the program so that the computer will store and use the variables when their names are first encountered in the program.

D. *PROBLEM USING A TYPE STATEMENT*

1. Problem -- The Fibonacci Sequence, 1,1,2,3,5,8,13 . . ., starts with two ones and new terms are formed according to the rule that each term is equal to the sum of the 2 preceding terms. Find the first 30 terms of the sequence and all the partial sums, FSUM (I) of the sequence up to and including the Ith term.

2. Flowchart



3. Solution

IBM FORTRAN Coding Form

FIBONACCI SEQUENCE 1 1

```
C     THIS PROGRAM DEALS WITH THE FIBONACCI SEQUENCE 1,1,2,3,5,8,13,.....
C     START WITH TWO ONES AND FORM NEW TERMS ACCORDING TO THE RULE THAT
C     EACH TERM IS EQUAL TO THE SUM OF THE TWO PRECEDING TERMS.
C     THE SERIES IS GENERATED FOR TWENTY TERMS, AND PARTIAL SUMS ARE
C     GIVEN UP TO AND INCLUDING EACH TERM.
      INTEGER PSUM(30),TERM(30)
      WRITE(3,3)
    3 FORMAT(16X,'FIBONACCI SEQUENCE')
      WRITE(3,2)
    2 FORMAT(7X,'TERM NO.',11X,'TERM',9X,'PARTIAL SUM')
      PSUM(1)=1
      PSUM(2)=2
      TERM(1)=1
      TERM(2)=1
      DO 10 I=3,22
      TERM(I)=TERM(I-1)+TERM(I-2)
      PSUM(I)=PSUM(I-1)+TERM(I)
      J=I-2
      WRITE(3,1)J,TERM(J),PSUM(J)
    1 FORMAT(10X,I5,10X,I5,15X,I5)
   10 CONTINUE
      CALL EXIT
      END
```

4. Printout

```
PAGE    1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0001        0001         0000

V2 M04   ACTUAL  8K  CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
C     THIS PROGRAM DEALS WITH THE FIBONACCI SEQUENCE 1,1,2,3,5,8,13,....
C     START WITH TWO ONES AND FORM NEW TERMS ACCORDING TO THE RULE THAT
C     EACH TERM IS EQUAL TO THE SUM OF THE TWO PRECEDING TERMS.
C     THE SERIES IS GENERATED FOR TWENTY TERMS, AND PARTIAL SUMS ARE
C     GIVEN UP TO AND INCLUDING EACH TERM.
      INTEGER PSUM(30),TERM(30)
      WRITE(3,3)
    3 FORMAT(16X,'FIBONACCI SEQUENCE')
      WRITE(3,2)
    2 FORMAT(7X,'TERM NO.',11X,'TERM',9X,'PARTIAL SUM')
      PSUM(1)=1
      PSUM(2)=2
      TERM(1)=1
      TERM(2)=1
      DO 10 I=3,22
      TERM(I)=TERM(I-1)+TERM(I-2)
      PSUM(I)=PSUM(I-1)+TERM(I)
      J=I-2
      WRITE(3,1)J,TERM(J),PSUM(J)
    1 FORMAT(10X,I5,10X,I5,15X,I5)
   10 CONTINUE
      CALL EXIT
      END
```

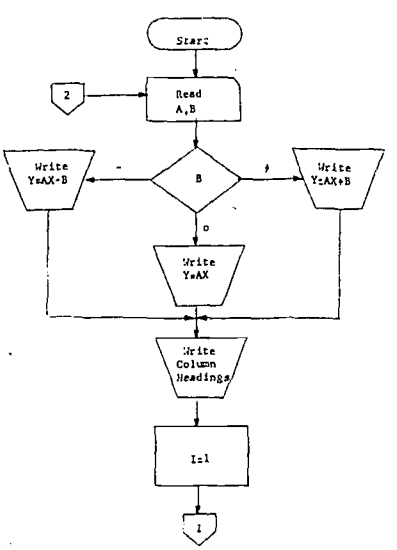
```
FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     128  PROGRAM     176

 END OF COMPILATION

// XEQ
```

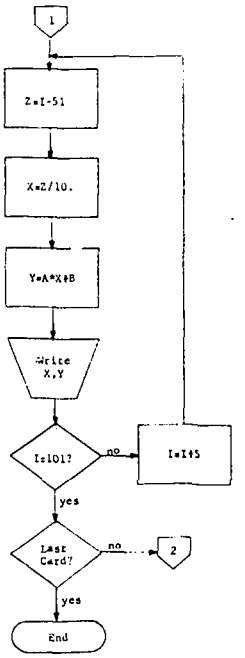
FIBONACCI SEQUENCE

TERM NO.	TERM	PARTIAL SUM
1	1	1
2	1	2
3	2	4
4	3	7
5	5	12
6	8	20
7	13	33
8	21	54
9	34	88
10	55	143
11	89	232
12	144	376
13	233	609
14	377	986
15	610	1596
16	987	2583
17	1597	4180
18	2584	6764
19	4181	10945
20	6765	17710

E. PROBLEMS WITH DO-LOOPS

1. Problem. Produce a table of values of x and y for a linear equation of the form y = ax + b. Read A and B from cards and compute values of y for each x from -5 to +5 at intervals of .5.
   a. Flowchart





b. Coding Forms

IBM FORTRAN Coding Form

{(X,Y)} for Y = AX+B

1 2

FORTRAN STATEMENT

```
C     THIS PROGRAM WILL PRODUCE A LIST OF VALUES OF X AND Y FOR A LINEAR
C     EQUATION,  Y=AX+B.
C     A AND B ARE READ FROM CARDS AND X IS BETWEEN -5 AND +5 AT INTERVALS
C     OF .5.
   50 READ(2,3)A,B
    3 FORMAT(2F7.2)
      IF(B)5,6,7
    5 WRITE(3,4)A,B
    4 FORMAT(2X,'EQUATION  Y=',F7.2,'X',F7.2)
      GO TO 10
    6 WRITE(3,8)A
    8 FORMAT(2X,'EQUATION  Y=',F7.2,'X')
      GO TO 10
    7 WRITE(3,9)A,B
    9 FORMAT(2X,'EQUATION  Y=',F7.2,'X +',F7.2)
   10 WRITE(3,11)
   11 FORMAT(7X,'X',6X,'Y')
      DO 20 I=1,101,5
      Z=I-51
      X=Z/10.
      Y=A*X+B
      WRITE(3,25)X,Y
   25 FORMAT(3X,F5.1,2X,F7.2)
   20 CONTINUE
```

IBM FORTRAN Coding Form

2 2

FORTRAN STATEMENT

```
      GO TO 50
   99 CALL EXIT
      END
```

c. Printout

```
PAGE    1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0001        0001        0000

V2 M04   ACTUAL  8K  CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
C       THIS PROGRAM WILL PRODUCE A LIST OF VALUES OF X AND Y FOR A LINEAR-
C       EQUATION,  Y=AX+B.
C       A AND B ARE READ FROM CARDS AND X IS BETWEEN -5 AND +5 AT INTERVALS
C       OF .5.
     50 READ(2,3)A,B
      3 FORMAT(2F7.2)
        IF(B)5,6,7
      5 WRITE(3,4)A,B
      4 FORMAT(2X,'EQUATION   Y=',F7.2,'X',F7.2)
        GO TO 10
      6 WRITE(3,8)A
      8 FORMAT(2X,'EQUATION   Y=',F7.2,'X')
        GO TO 10
      7 WRITE(3,9)A,B
      9 FORMAT(2X,'EQUATION   Y=',F7.2,'X +',F7.2)
     10 WRITE(3,11)
     11 FORMAT(7X,'X',6X,'Y')
        DO 20 I=1,101,5
        Z=I-51
        X=Z/10.
        Y=A*X+B
        WRITE(3,25)X,Y
     25 FORMAT(3X,F5.1,2X,F7.2)
     20 CONTINUE
        GO TO 50
     99 CALL EXIT
        END

UNREFERENCED STATEMENTS
  99

FEATURES SUPPORTED
  IOCS

CORE REQUIREMENTS FOR
  COMMON       0  VARIABLES      12  PROGRAM     186

  END OF COMPILATION

// XEQ

 EQUATION   Y=    1.00X +   2.00
        X          Y
     -5.0      -3.00
     -4.5      -2.50
     -4.0      -2.00
     -3.5      -1.50
     -3.0      -1.00
     -2.5      -0.50
     -2.0       0.00
     -1.5       0.50
     -1.0       1.00
     -0.5       1.50
      0.0       2.00
      0.5       2.50
      1.0       3.00
      1.5       3.50
      2.0       4.00
      2.5       4.50
      3.0       5.00
      3.5       5.50
      4.0       6.00
      4.5       6.50
      5.0       7.00
```

```
EQUATION    Y=   12.50X   -3.25
     X          Y
  -5.0     -65.75
  -4.5     -59.50
  -4.0     -53.25
  -3.5     -47.00
  -3.0     -40.75
  -2.5     -34.50
  -2.0     -28.25
  -1.5     -22.00
  -1.0     -15.75
  -0.5      -9.50
   0.0      -3.25
   0.5       3.00
   1.0       9.25
   1.5      15.50
   2.0      21.75
   2.5      28.00
   3.0      34.25
   3.5      40.50
   4.0      46.75
   4.5      53.00
   5.0      59.25
EQUATION    Y=    0.00X +    5.00
     X          Y
  -5.0       5.00
  -4.5       5.00
  -4.0       5.00
  -3.5       5.00
  -3.0       5.00
  -2.5       5.00
  -2.0       5.00
  -1.5       5.00
  -1.0       5.00
  -0.5       5.00
   0.0       5.00
   0.5       5.00
   1.0       5.00
   1.5       5.00
   2.0       5.00
   2.5       5.00
   3.0       5.00
   3.5       5.00
   4.0       5.00
   4.5       5.00
   5.0       5.00
```

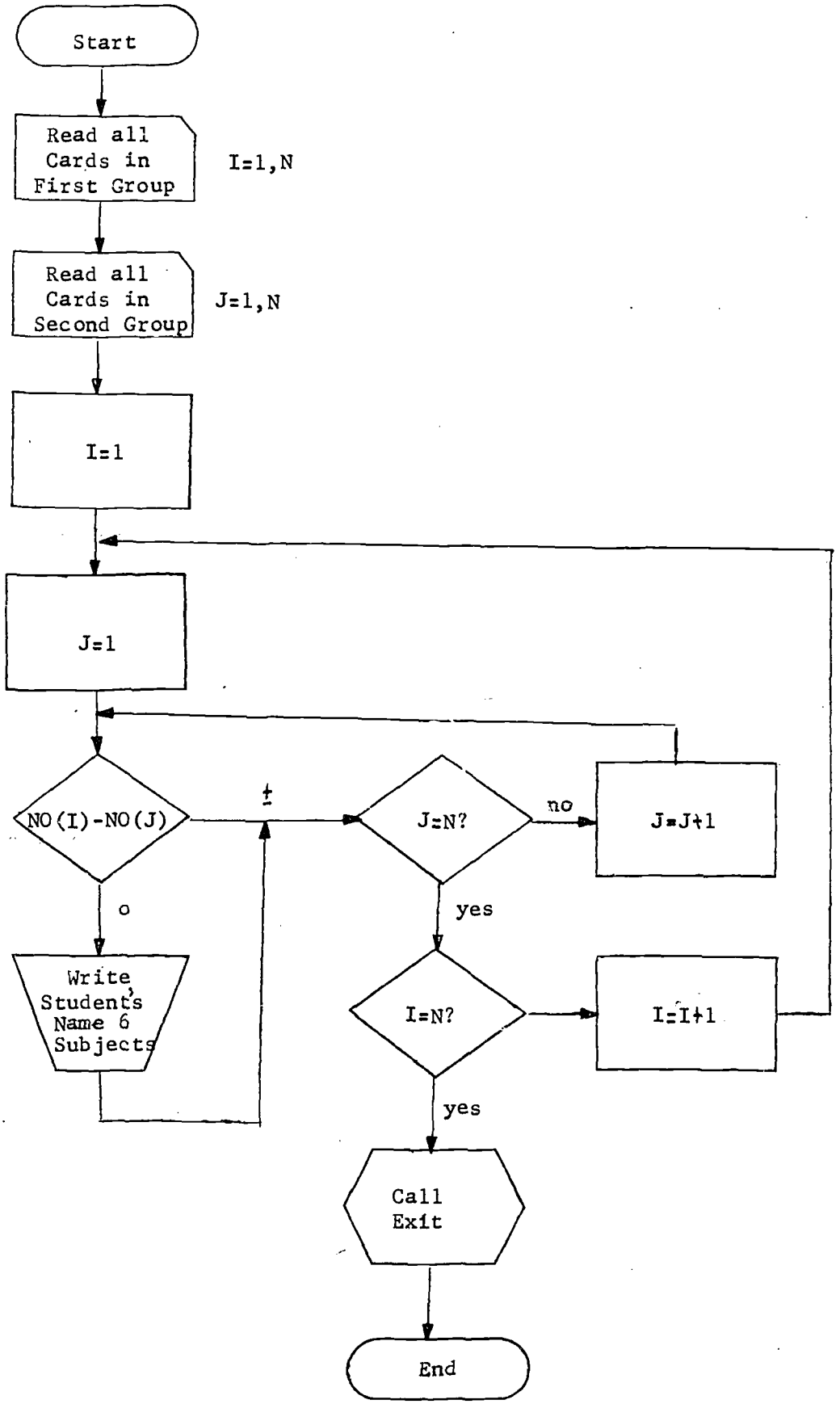
2. Problem. A school uses one data card to hold general student information as follows:

	Columns
Name	1-20
Address	21-40
Telephone	41-50
Zip	51-55
Student number	76-80

A second card contains the student's schedule, using room numbers and subject codes with the student number punched in columns 1-5, subject codes each require 3 digits and room numbers require 3 digits. Each student follows a schedule of 4 to 6 classes a day.

Write a program to print the student's name and his schedule.

a. Flowchart


Start
Read all Cards in First Group
I=1,N
Read all Cards in Second Group
J=1,N
I=1
J=1
NO(I)-NO(J)
±
J=N?
no
J=J+1
o
yes
Write Student's Name & Subjects
I=N?
I=I+1
yes
Call Exit
End

b. Coding Forms

IBM FORTRAN Coding Form

STUDENT'S SCHEDULE

FORTRAN STATEMENT

```
XEXTENDED PRECISION
X ONE WORD INTEGERS
      INTEGER STUNO(10), STNO2(10), RN1(10), RN2(10), RN3(10), RN4(10), RN5(10
     1), RN6(10)
      REAL NAM1(10), NAM2(10), NAM3(10), NAM4(10)
      DIMENSION SUB1(10), SUB2(10), SUB3(10), SUB4(10), SUB5(10), SUB6(10)
      READ(2,4)N
    4 FORMAT(I2)
      READ(2,1)(NAM1(I), NAM2(I), NAM3(I), NAM4(I), STUNO(I), I=1,N)
    1 FORMAT(4A5,55X,I5)
      READ(2,2)(STNO2(J), SUB1(J), RN1(J), SUB2(J), RN2(J), SUB3(J), RN3(J), SU
     1B4(I), RN4(J), SUB5(J), RN5(J), SUB6(J), RN6(J), J=1,N)
    2 FORMAT(I5,A5,I3,A5,I3,A5,I3,A5,I3,A5,I3,A5,I3)
      DO 10 I=1,N
      DO 10 J=1,N
      IF(STNO2(J)-STUNO(I))10,9,10
    9 WRITE(3,3)NAM1(I), NAM2(I), NAM3(I), NAM4(I), SUB1(I), RN1(J), SUB2(J),
     1RN2(J), SUB3(J), RN3(J), SUB4(I), RN4(J), SUB5(J), RN5(J), SUB6(J), RN6(J)
    3 FORMAT(2X,4A5,///,2X,'PERIOD 1',5X,A5,5X,I3,//,9X,'2',5X,A5,5X,I3,
     2//,9X,'3',5X,A5,5X,I3,//,9X,'4',5X,A5,5X,I3,//,9X,'5',5X,A5,5X,I3,
     2//,9X,'6',5X,A5,5X,I3,////)
   10 CONTINUE
      CALL EXIT
      END
```

IBM FORTRAN Coding Form

DATA (STUDENT'S SCHEDULE)

FORTRAN STATEMENT

```
0.3                                                                    00109
BLOW JOE 0                                                             00209
NEX MEX                                                                00309
CARL SANDY
CC109EN  I2C1LG I2C3GN HS309PH EDL11STUDY2C6NM 4K301
CC209LG  I2C3EN  I2C1GN HS309PH EDL11NM  4K301STUDSLC4
CC309GN  HS309LG I2C3EN  I2C1GN SE315PH EDL12STUDYLC4
```

c. Printout

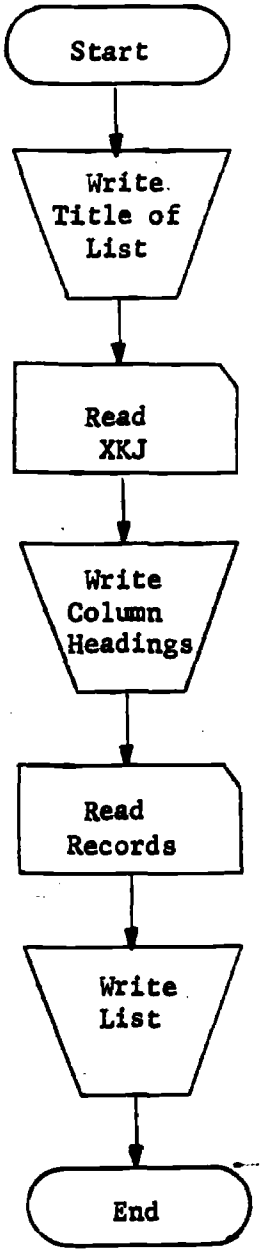
```
PAGE    1
// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000          0001         0001         0000

V2 M04    ACTUAL   8K   CONFIG   8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
*EXTENDED PRECISION
* ONE WORD INTEGERS
      INTEGER STUNO(10),STNO2(10),RN1(10),RN2(10),RN3(10),RN4(10),RN5(10
     1),RN6(10)
      REAL NAM1(10),NAM2(10),NAM3(10),NAM4(10)
      DIMENSION SUB1(10),SUB2(10),SUB3(10),SUB4(10),SUB5(10),SUB6(10)
      READ(2,4)N
    4 FORMAT(I2)
      READ(2,1)(NAM1(I),NAM2(I),NAM3(I),NAM4(I),STUNO(I),I=1,N)
    1 FORMAT(4A5,55X,I5)
      READ(2,2)(STNO2(J),SUB1(J),RN1(J),SUB2(J),RN2(J),SUB3(J),RN3(J),SU
     1B4(J),RN4(J),SUB5(J),RN5(J),SUB6(J),RN6(J),J=1,N)
    2 FORMAT(I5,A5,I3,A5,I3,A5,I3,A5,I3,A5,I3,A5,I3)
      DO 10 I=1,N
      DO 10 J=1,N
      IF(STNO2(J)-STUNO(I))10,9,10
    9 WRITE(3,3)NAM1(I),NAM2(I),NAM3(I),NAM4(I),SUB1(J),RN1(J),SUB2(J),
     1RN2(J),SUB3(J),RN3(J),SUB4(J),RN4(J),SUB5(J),RN5(J),SUB6(J),RN6(J)
    3 FORMAT(2X,4A5,///,2X,'PERIOD 1',5X,A5,5X,I3,//,9X,'2',5X,A5,5X,I3,
     1//,9X,'3',5X,A5,5X,I3,//,9X,'4',5X,A5,5X,I3,//,9X,'5',5X,A5,5X,I3,
     2//,9X,'6',5X,A5,5X,I3,////)
   10 CONTINUE
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     388  PROGRAM     346

 END OF COMPILATION

// XEQ

 BLOW JOE 0

 PERIOD 1      EN  I      201
        2      ALG I      203
        3      GN HS      309
        4      PH ED      111
        5      STUDY      206
        6      HM MK      301

 NIX MIX

 PERIOD 1      ALG I      203
        2      EN  I      201
        3      GN HS      309
        4      PH ED      111
        5      HM MK      301
        6      STUDY      104

 CLAUS SANTA

 PERIOD 1      GN HS      309
        2      ALG I      203
        3      EN  I      201
        4      GN SI      315
        5      PH ED      112
        6      STUDY      104
```

F. A PROBLEM USING ALPHABETIC INFORMATION

Problem. Write a program to produce a list of names in the order, first name, last name, address, telephone number. Data cards will be made up with the first name in columns 1-10, last name in columns 11-20, address in columns 21-40, and telephone number in columns 41-50, each variable being left adjusted in its field. Write the program to provide for any number of cards less than 81. Properly identify the list and provide appropriate column headings and spacing for neatness.

2. Solution:
   a. Because the size of the allowable field of alphabetic information is limited, EXTENDED PRECISION may be used to allow grouping variables in 5 letter groups. However, since each field is longer than 5 letters, two or more variables must be used to represent one record e.g.
   FNAM1, FNAM2 will be the two parts of the first name.
   b. A variable representing the number of cards to be read is first defined by reading from a card.

3. Flowchart (see following page)



4. Coding Form

IBM FORTRAN Coding Form

LIST - NAME, ADDRESS, TELEPHONE

FORTRAN STATEMENT

```
* EXTENDED PRECISION
C THIS IS A PROGRAM TO PRODUCE LISTS OF STUDENTS ENROLLED IN THE SUMMER PROGRAM
      DIMENSION FNAM1(80),FNAM2(80),SNAM1(80),SNAM2(80),ADD1(80)
      DIMENSION ADD2(80),ADD3(80),ADD4(80),TEL1(80),TEL2(80)
      WRITE(3,1000)
 1000 FORMAT(2X,'STUDENTS ENROLLED',/,2X,'JOHN F. KENNEDY SENIOR HIGH
     1SCHOOL',/,2X,'PROBLEM SOLVING-COMPUTER STYLE',/,2X,'SUMMER 1968',/
     2///)
      READ (2,5) XKJ
    5 FORMAT (F10.2)
      KJ=XKJ
      WRITE(3,1)
    1 FORMAT(1H1)
      WRITE(3,4)
    4 FORMAT (2X,'FIRST NAME',3X,'LAST NAME',4X,'ADDRESS',16X,'TELEPHONE
     1',////)
      READ (2,2) (FNAM1(I),FNAM2(I),SNAM1(I),SNAM2(I),ADD1(I),ADD2(I),AD
     1D3(I),ADD4(I),TEL1(I),TEL2(I),I=1,KJ)
    3 FORMAT (2X,2A5,3X,2A5,3X,4A5,3X,2A5,//)
    2 FORMAT (10A5)
      WRITE(3,3) (FNAM1(I),FNAM2(I),SNAM1(I),SNAM2(I),ADD1(I),ADD2(I),AD
     1D3(I),ADD4(I),TEL1(I),TEL2(I),I=1,KJ)
      CALL EXIT
      END
```

5. Printout

```
PAGE    1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000         0001         0001         0000

V2 M04   ACTUAL  8K  CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
* EXTENDED PRECISION
C THIS IS A PROGRAM TO PRODUCE LISTS OF STUDENTS ENROLLED IN THE SUMMER PROGRAM
      DIMENSION FNAM1(80),FNAM2(80),SNAM1(80),SNAM2(80),ADD1(80)
      DIMENSION ADD2(80),ADD3(80),ADD4(80),TEL1(80),TEL2(80)
      WRITE(3,1000)
 1000 FORMAT(2X,'STUDENTS ENROLLED',/,2X,'JOHN F. KENNEDY SENIOR HIGH
     1SCHOOL',/,2X,'PROBLEM SOLVING-COMPUTER STYLE',/,2X,'SUMMER 1968',/
     2///)
      READ (2,5) XKJ
    5 FORMAT (F10.2)
      KJ=XKJ
      WRITE(3,1)
    1 FORMAT(1H1)
      WRITE(3,4)
    4 FORMAT (2X,'FIRST NAME',3X,'LAST NAME',4X,'ADDRESS',16X,'TELEPHONE
     1',////)
      READ (2,2) (FNAM1(I),FNAM2(I),SNAM1(I),SNAM2(I),ADD1(I),ADD2(I),AD
     1D3(I),ADD4(I),TEL1(I),TEL2(I),I=1,KJ)
    3 FORMAT (2X,2A5,3X,2A5,3X,4A5,3X,2A5,//)
    2 FORMAT (10A5)
      WRITE(3,3) (FNAM1(I),FNAM2(I),SNAM1(I),SNAM2(I),ADD1(I),ADD2(I),AD
     1D3(I),ADD4(I),TEL1(I),TEL2(I),I=1,KJ)
      CALL EXIT
      END
```

```
FEATURES SUPPORTED
 EXTENDED PRECISION
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    2410  PROGRAM     258

 END OF COMPILATION

// XEQ

 STUDENTS ENROLLED
 JOHN F. KENNEDY SENIOR HIGH    SCHOOL
 PROBLEM SOLVING-COMPUTER STYLE
 SUMMER 1968

 FIRST NAME     LAST NAME      ADDRESS                    TELEPHONE


 AYLIE          ARTUS          1713 TENNESSEE ST.         943-7298

 WALTER         BARNES         14 THRUSH ST.              282-9557

 SALVATORE      BECNEL         11200 CHEF MENTEUR         242-1150

 CHARLES        BIBBINS        1220 S. MIRO               529-2342
```

G. *ANOTHER VERSION OF THE SAME PROGRAM*

1. Problem. Write a program similar to the one just illustrated, but arrange the list in block style, as would be used to address envelopes. Zip codes will be found in columns 51-55.

2. Solution:
   a. Since the alphabetic information has been given variable names, it can be moved by adjusting the FORMAT statements. The same type of output could be produced for extremely long lists without storing all the variables using the continuous loop technique.
   b. Coding form.

IBM FORTRAN Coding Form

MAILING LIST 1 1

FORTRAN STATEMENT

```
*EXTENDED PRECISION
C  THIS IS A PROGRAM TO PRODUCE A MAILING LIST OF NOT MORE THAN 80 NAMES
      DIMENSION FNAM1(80),FNAM2(80),SNAM1(80),SNAM2(80),ADD1(80),ADD2(80
     1),ADD3(80),ADD4(80),ZIP(80)
      READ(2,5)KJ
    5 FORMAT(I2)
      WRITE(3,1)
    1 FORMAT(1H1)
      WRITE(3,1000)
 1000 FORMAT(2X,'STUDENTS ENROLLED',/,2X,'JOHN F. KENNEDY SENIOR HIGH SC
     1HOOL',/,2X,'PROBLEM SOLVING-COMPUTER STYLE',/,2X,'SUMMER 1969',///
     2/)
      READ(2,2)(FNAM1(I),FNAM2(I),SNAM1(I),SNAM2(I),ADD1(I),ADD2(I),ADD
     13(I),ADD4(I),ZIP(I),I=1,KJ)
    2 FORMAT(2X,4A5,/,2X,4A5,/,2X,'NEW ORLEANS, LOUISIANA',2X,A5,////)
      CALL EXIT
      END
```

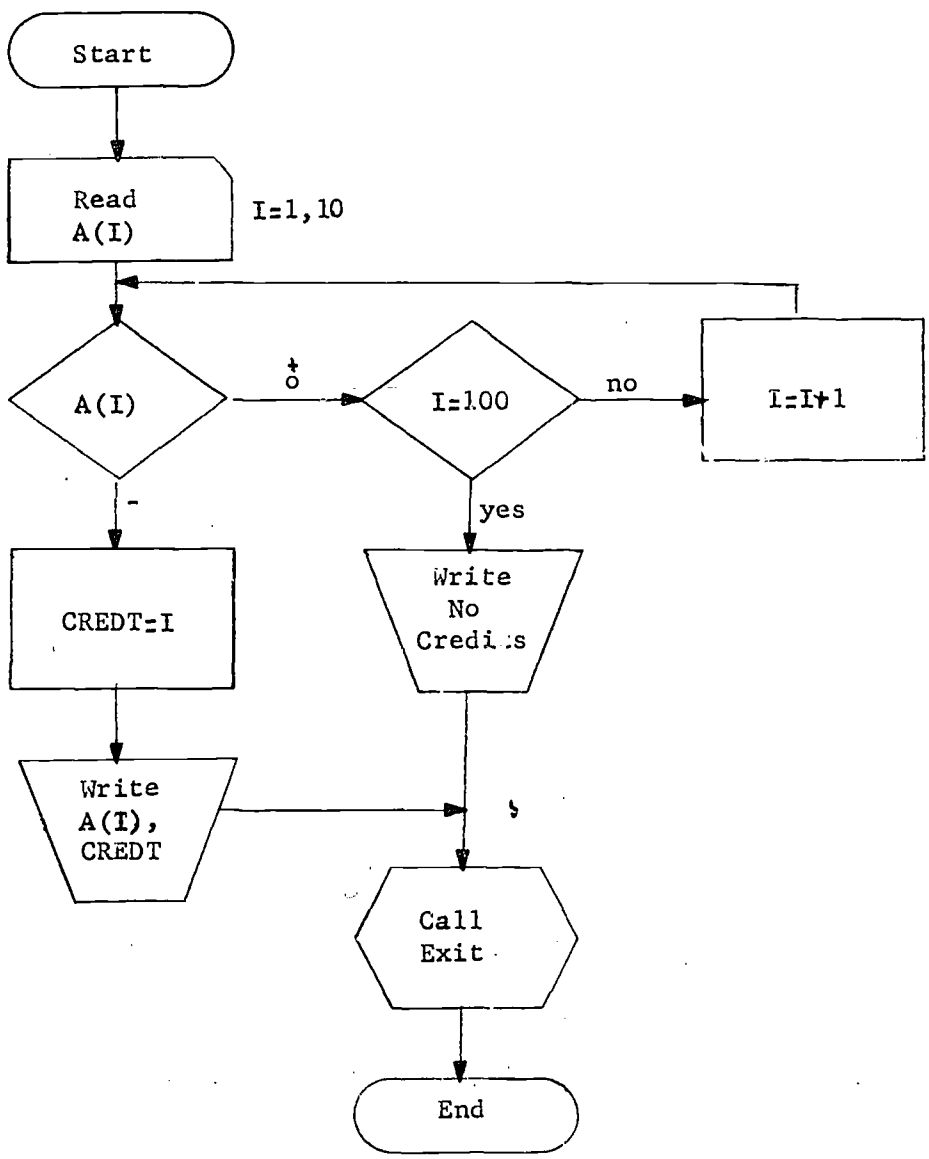
c. Printout.

```
PAGE    1
// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0001         0001         0000

V2 M04    ACTUAL  8K  CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
*EXTENDED PRECISION
C       THIS IS A PROGRAM TO PRODUCE A MAILING LIST OF NOT MORE THAN 80 NAME
        DIMENSION FNAM1(80),FNAM2(80),SNAM1(80),SNAM2(80),ADD1(80),ADD2(80
      1),ADD3(80),ADD4(80),ZIP(80)
        READ(2,5)KJ
    5   FORMAT(I2)
        WRITE(3,1)
    1   FORMAT(1H1)
        WRITE(3,1000)
 1000   FORMAT(2X,'STUDENTS ENROLLED',/,2X,'JOHN F. KENNEDY SENIOR HIGH SC
      1HOOL',/,2X,'PROBLEM SOLVING-COMPUTER STYLE',/,2X,'SUMMER 1969',///
      2/)
        READ(2,2) (FNAM1(I),FNAM2(I),SNAM1(I),SNAM2(I),ADD1(I),ADD2(I),ADD
      13(I),ADD4(I),ZIP(I),I=1,KJ)
    2   FORMAT(8A5,10X,A5)
        WRITE(3,3)(FNAM1(I),FNAM2(I),SNAM1(I),SNAM2(I),ADD1(I),ADD2(I),ADD
      13(I),ADD4(I),ZIP(I),I=1,KJ)
    3   FORMAT(2X,4A5,/,2X,4A5,/,2X,'NEW ORLEANS, LOUISIANA',2X,A5,////)
        CALL EXIT
        END

FEATURES SUPPORTED
 EXTENDED PRECISION
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    2168  PROGRAM     226

 END OF COMPILATION

// XEQ


 STUDENTS ENROLLED
 JOHN F. KENNEDY SENIOR HIGH SCHOOL
 PROBLEM SOLVING-COMPUTER STYLE
 SUMMER 1969



 AYLIE      ARTUS
 1713 TENNESSEE ST.
 NEW ORLEANS, LOUISIANA  70117



 WALTER     BARNES
 14 THRUSH ST.
 NEW ORLEANS, LOUISIANA  70124



 SALVATORE BECNEL
 11200 CHEF MENTEUR
 NEW ORLEANS, LOUISIANA  70128
```

H. A PROBLEM HAVING A SPECIAL EXIT FROM A LOOP

1. Problem. Write a program which will examine a list of 10 numbers representing debits and credits to find the first location in which a credit (negative number) is found. The location of this number on the list and the number itself are printed.

2. Solution.
   a. Flowchart



b. Coding form

IBM FORTRAN Coding Form

FIRST CREDIT

FORTRAN STATEMENT

```
C THIS PROGRAM FINDS THE LOCATION OF THE FIRST CREDIT FROM A LIST OF 10
C DEBITS AND/OR CREDITS. IT ILLUSTRATES A SPECIAL EXIT FROM A LOOP
      DIMENSION A(10)
    8 READ(2,1)(A(I),I=1,10
    1 FORMAT(8F10.2)
      DO 2 I=1,10
      IF(A(I))3,2,2
    2 CONTINUE
      GO TO 5
    3 CREDIT=I
      WRITE(3,4)CREDIT,A(I)
    4 FORMAT(2X,'CREDIT ACCOUNT NO.',I3,'$'F6.2)
      GO TO 8
    5 WRITE(3,6)
    6 FORMAT(2X,'NO ACCOUNTS ARE CREDITED')
      GO TO 8
    7 CALL EXIT
      END
```

c. Printout

```
PAGE    1
// JOB

LOG DRIVE    CART SPEC    CART AVAIL    PHY DRIVE
  0000         0001         0001          0000

V2 M04    ACTUAL  8K  CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
C THIS PROGRAM FINDS THE LOCATION OF THE FIRST CREDIT FROM A LIST OF 10
C DEBITS AND/OR CREDITS. IT ILLUSTRATES A SPECIAL EXIT FROM A LOOP
      DIMENSION A(10)
    8 READ(2,1)(A(I),I=1,10)
    1 FORMAT(8F10.2)
      DO 2 I=1,10
      IF(A(I))3,2,2
    2 CONTINUE
      GO TO 5
    3 CREDT=I
      WRITE(3,4)CREDT,A(I)
    4 FORMAT(2X,'CREDIT ACCOUNT NO.',F3.0,1X,'$'F6.2)
      GO TO 8
    5 WRITE(3,6)
    6 FORMAT(2X,'NO ACCOUNTS ARE CREDITED')
      GO TO 8
    7 CALL EXIT
      END

UNREFERENCED STATEMENTS
  7

FEATURES SUPPORTED
  IOCS

CORE REQUIREMENTS FOR
  COMMON       0  VARIABLES      26  PROGRAM     142

  END OF COMPILATION

// XEQ

  NO ACCOUNTS ARE CREDITED
  CREDIT ACCOUNT NO. 6. $-14.98
```

FORTRAN LANGUAGE

Unit VII

A. *FUNCTION SUBPROGRAM*

1. When a function is to be used repeatedly in a program, but cannot be expressed in a single FORTRAN arithmetic expression, a function subprogram can be written, stored on the disk, and used in the same manner as a library function, or an arithmetic statement function.

2. Form
   a. FUNCTION Statement — The word FUNCTION followed by a function name, an open parenthesis, a list of dummy arguments separated by commas, and a closed parenthesis.
      FUNCTION DIST (A,B)
      (1) The name has 1-5 alphameric characters, the first of which is alphabetic.
      (2) The mode of the FUNCTION is implied by the first letter of its name.
      (3) Arguments
         (a) The arguments need not agree in mode with the FUNCTION.
         (b) The arguments are *dummy* variables. Their value is not defined in the subprogram but will be supplied by the statement in the main program which mentions the function name.
   b. After the FUNCTION statement any combination of program steps may follow to determine the value of a single result.
   c. A statement must be included in the program steps that sets the NAME of the function equal to the desired results.
   d RETURN Statement
      (1) The word RETURN
      (2) The computer is told to start the main program where it left off.
   e. END Statement
      (1) The word END
      (2) Every subprogram must have its own END statement.
   f. If the FUNCTION deals with arrays, it must have a DIMENSION statement. This would be placed immediately following the FUNCTION statement.

3. Execution — The FUNCTION is called upon by using its name in an expression in the main program.

B. *WRITING A FUNCTION SUBPROGRAM*

1. Problem: Although the 1130 library does not contain the arcsin function, a programmer solving trigonometry problems would probably have use for this function. Write a FUNCTION subprogram called ASINE which will compute the value of THETA in radians for any known value of the sine of theta. Remember that the word SIN can not be used as a variable as it is a library function.

2 Solution:

IBM FORTRAN Coding Form

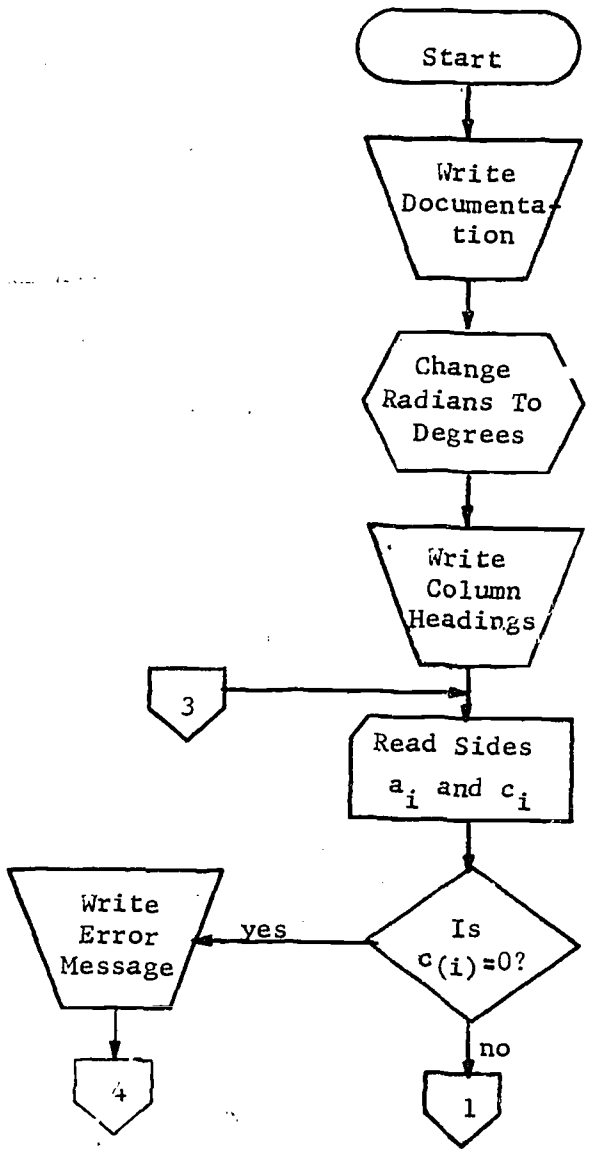
FORTRAN STATEMENT

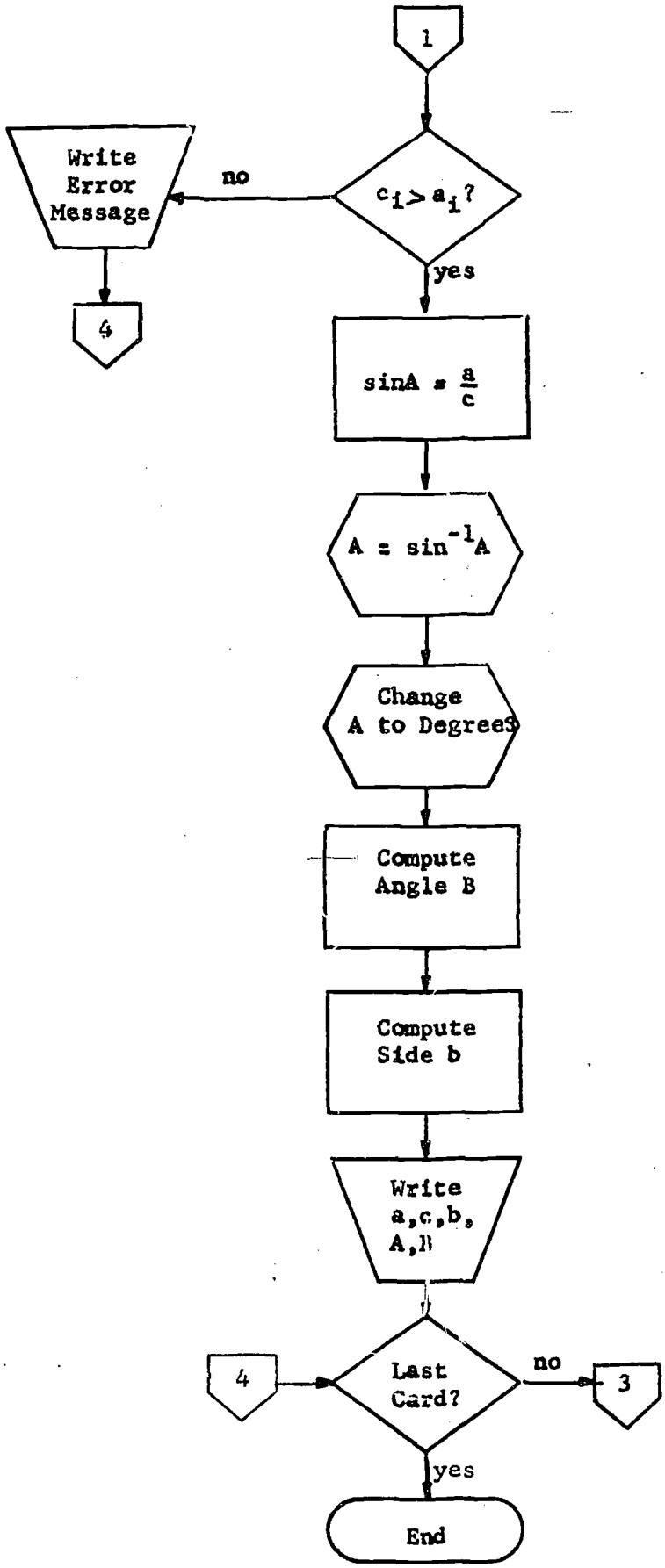
```
FUNCTION ASINE(SIANG)
TAN=SIANG/SQRT(1-SIANG**2)
THETA=ATAN(TAN)
ASINE=THETA
RETURN
END
```

C. *PROGRAM USING A FUNCTION SUBPROGRAM*

1. Problem — Sides A and C of 20 right triangles ABC are known. Solve the triangles. Allow space for values of A and C below 1000 to the nearest hundredth. Print out the given information and the values of the angles in degrees to the nearest tenth and the missing sides to the nearest hundredth. If A and C are not feasible sides of triangles print out a message to this effect. Flowchart the problem first.

2. Flowchart




1
Write Error Message
no
$c_i > a_i$?
yes
4
sinA = a/c
A = sin^{-1}A
Change A to Degrees
Compute Angle B
Compute Side b
Write a,c,b, A,B
4
Last Card?
no
3
yes
End

3. Solution

IBM

FORTRAN Coding Form

```
C NAME SIHYP
C RIGHT TRIANGLE PROBLEM. A IS A SIDE AND C IS THE HYPOTENUSE. A AND C ARE KNOWN
C TO BE LESS THAN 1000. ANGLES ARE COMPUTED TO HUNDREDTHS OF DEGREES. SIDES ARE
C COMPUTED TO THE NEAREST HUNDREDTH.
      DIMENSION A(20),B(20),C(20),DEGRA(20),DEGRB(20)
      DEG(RAD)=RAD*180./3.1416
      WRITE(3,2)
    2 FORMAT(14X,'GIVEN',28X,'SOLUTION')
      WRITE(3,3)
    3 FORMAT(9X,'A',12X,'C',14X,'B',9X,'ANGLE A',6X,'ANGLE B')
      DO 20 I=1,20
      READ(2,4) A(I),C(I)
    4 FORMAT(2X,F8.2)
      SINA=A(I)/C(I)
      IF(C(I))6,7,6
    7 WRITE(3,8)I
    8 FORMAT(2X,I2,2X,'C MUST BE GIVEN, NO SOLUTION')
      IF (C(I)-A(I))10,10,6
   10 WRITE(3,11)
   11 FORMAT(2X,'C MUST BE GREATER THAN A, NO SOLUTION')
      GO TO 20
    6 RADA=ARSINE(SINA)
      DEGRA(I)=DEG(RADA)
      DEGRB(I)=90.-DEGRA(I)
```

IBM

FORTRAN Coding Form

```
      B(I)=SQRT(C(I)**2-A(I)**2)
      WRITE(3,5)I,A(I),C(I),B(I),DEGRA(I),DEGRB(I)
    5 FORMAT(2X,I2,2X,F8.2,5X,F8.2,7X,F8.2,5X,F8.2,5X,F8.2)
   20 CONTINUE
      CALL EXIT
      END
```

4. Printout

```
PAGE    1
// JOB
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
   0000        0001        0001        0000
V2 M05   ACTUAL 8K CONFIG 8K
// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
* NAME SIHYP
C RIGHT TRIANGLE PROBLEM. A IS A SIDE AND C IS THE HYPOTENUSE. A AND C ARE KNOWN
C TO BE LESS THAN 1000. ANGLES ARE COMPUTED TO HUNDREDTHS OF DEGREES. SIDES ARE
C COMPUTED TO THE NEAREST HUNDREDTH.
      DIMENSION A(20),B(20),C(20),DEGRA(20),DEGRB(20)
      DEG(RAD)=RAD*180./3.1415
      WRITE(3,2)
    2 FORMAT(14X,'GIVEN',28X,'SOLUTION')
      WRITE(3,3)
    3 FORMAT(9X,'A',12X,'C',14X,'B',9X,'ANGLE A',6X,'ANGLE B')
      DO 20 I=1,20
      READ(2,4) A(I),C(I)
    4 FORMAT(2F8.2)
      SINA=A(I)/C(I)
      IF(C(I))9,7,9
    7 WRITE(3,8)
```

```
    8 FORMAT(2X,'C MUST BE GIVEN, NO SOLUTION')
      GO TO 20
    9 IF(C(I)-A(I))10,10,6
   10 WRITE(3,11)
   11 FORMAT(2X,'C MUST BE GREATER THAN A, NO SOLUTION')
      GO TO 20
    6 SINA=A(I)/C(I)
      RADA=ASINF(SINA)
      DEGRA(I)=DEG(RADA)
      DEGRB(I)=90.-DEGRA(I)
      B(I)=SQRT(C(I)**2-A(I)**2)
      WRITE(3,5)A(I),C(I),B(I),DEGRA(I),DEGRB(I)
    5 FORMAT(2X,F8.2,5X,F8.2,7X,F8.2,5X,F8.2,5X,F8.2)
   20 CONTINUE
      CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR SIHYP
 COMMON       0  VARIABLES    214  PROGRAM    290

 END OF COMPILATION

// DUP

*STORE      WS  UA  SIHYP
CART ID 0001   DB ADDR  1FFA   DB CNT   0015


PAGE  1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
   0000        0001        0001         0000

V2 M05   ACTUAL  8K  CONFIG  8K

// FOR
* LIST SOURCE PROGRAM
*NAME ASINE
      FUNCTION ASINE(SIANG)
      TAN=SIANG/SQRT(1.-SIANG**2)
      THETA=ATAN(TAN)
      ASINE=THETA
      RETURN
      END

CORE REQUIREMENTS FOR ASINE
 COMMON       0  VARIABLES      8  PROGRAM     38

 END OF COMPILATION

// DUP

*STORE      WS  UA  ASINE
CART ID 0001   DB ADDR  200E   DB CNT   0004

// XEQ SIHYP

              GIVEN                              SOLUTION
         A          C          B          ANGLE A    ANGLE B
      4.00       5.00       3.00       53.12      36.87
      7.00      12.00       9.74       35.68      54.31
      3.50       6.25       5.17       34.05      55.94
    154.27     200.01     127.29       50.47      39.52
     38.50      62.75      49.55       37.84      52.15
C MUST BE GIVEN, NO SOLUTION
C MUST BE GIVEN, NO SOLUTION
C MUST BE GREATER THAN A, NO SOLUTION
C MUST BE GREATER THAN A, NO SOLUTION
    184.00     276.00     205.71       41.81      48.18
     18.00      36.00      31.17       29.99      60.00
     14.00      18.00      11.31       51.05      38.94
   5321.00    7832.00    5746.92       42.79      47.20
     45.00    5621.00    5620.82        0.45      89.54
     20.00      30.00      22.36       41.81      48.18
     12.00      56.00      54.69       12.37      77.62
   4532.00    4561.00     513.51       83.53       6.46
     89.00     852.00     847.33        5.99      84.00
     40.00     200.00     195.95       11.53      78.46
     65.00      89.00      60.79       46.91      43.08

PAGE  1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
   0000        0001        0001         0000

V2 M05   ACTUAL  8K  CONFIG  8K

// DUP

*DELETE          ASINE
CART ID 0001   DB ADDR  200E   DB CNT   0004

*DELETE          SIHYP
CART ID 0001   DB ADDR  1FFA   DB CNT   0015
```

D. *HOW TO STORE A FUNCTION SUBPROGRAM*

1. The deck for the subprogram may be placed after the END card of the main program.

2. Regular control cards precede the FUNCTION subprogram deck.
   a. // JOB
   b. // FOR
   c. *LIST SOURCE PROGRAM (optional)

3. Special cards which are needed to store the function subprogram on the disk.
   a. // DUP
      This is a monitor control card which activates the disk utility programs which accomplish storage on the disk.
   b. *STORE WS UA NAME
      Spacing on the storage card must be exactly as described. * in column *1*, STORE in columns *2* thru *6*, *6* blank spaces, WS in columns *13* and *14*, *2* blank spaces, UA in columns *17* and *18*, *2* blank spaces, NAME (the name of the program, formulated by the programmer in accordance with naming rules) begins in column 21.

E. *DELETING A PROGRAM FROM THE DISK*

1. To keep the USER area of the disk from becoming filled, and thereby unavailable, it is necessary to remove subprograms when they are no longer needed.

2. Three cards are needed to remove a program from the disk. They are:
   // JOB
   // DUP
   *DELETE ASINE
   The FORMAT of the DELETE card must be precise, * in column *1*, DELETE in columns *2* thru *7* and the subprogram name begins in column 21.

F. *HOW TO STACK THE DECK OF A PROGRAM HAVING SUBPROGRAMS*

```
// JOB
// FOR
*IOCS
*LIST SOURCE PROGRAM (optional)
*NAME SIHYP
C TRIANGLE PROBLEM. A IS A SIDE AND C, etc.
.
.
.
.
      (card deck for main program)
      .
      .
      .
      .
      END
// DUP
*STORE      WS  UA  SIHYP
// JOB
// FOR
*LIST SOURCE PROGRAM (optional)
      FUNCTION ASINE(SIANG)
      .
      .
      .
```

(card deck for FUNCTION subprogram)

```
      RETURN
      END
// DUP
*STORE    WS  UA  ASINE
// XEQ SIHYP
      DATA CARDS
// JOB
// DUP
*DELETE           ASINE
// DUP
*DELETE           SIHYP
```

G. *SUBROUTINE SUBPROGRAMS*

1. This type of program is similar in many ways to the FUNCTION subprogram with one exception being that it may be used to compute as many results as a problem may require.

2. The SUBROUTINE is used when a set of operations which will be used many times is grouped together.

3. It is brought into use by a special statement consisting of the word CALL, the subprogram NAME, selected by the programmer, and as many dummy arguments and variables as are needed to perform the computations. The variables are separated by commas and enclosed in parentheses.

   ```
   CALL ROOTS (A,B,C)
   ```

4. The SUBROUTINE is constructed using as many FORTRAN arithmetic statements as are needed to perform the desired operations.
   a. A statement beginning with the word SUBROUTINE and followed by the name and its arguments must be the first statement.

      ```
      SUBROUTINE ROOTS (A,B,C)
      ```

   b. The execution statements must end with a RETURN statement.
   c. Every subprogram must have its own END statement.
   d. If the SUBROUTINE deals with arrays, place a DIMENSION statement after the SUBROUTINE statement.
   e. COMMON Statement
      (1) COMMON statement is used to allow the sharing of storage between programs and subprograms, so that a value computed by a subprogram can become a part of the main program or of another subprogram.
      (2) The statement consists of the word COMMON followed by a list of variables or array names, separated by commas.

      ```
      COMMON HI, LOC
      ```

      (3) Every COMMON statement usually has a counterpart in another program or subprogram.
      (4) The variables listed in the COMMON statement will share storage locations in the order in which they are listed. If the first variable on the COMMON list of the main program is PRICE and the first on the subroutine COMMON list is COST, then PRICE and COST will share the same storage locations.
      (5) Any number of programs may have COMMON statements.
      (6) If there is no variable in the main program to be shared according to the variable list in a subprogram, the place or order in the list can be retained by inclusion of dummy variables as placeholders.
      (7) The COMMON statement is placed after the DIMENSION statement and must precede any executable program steps.

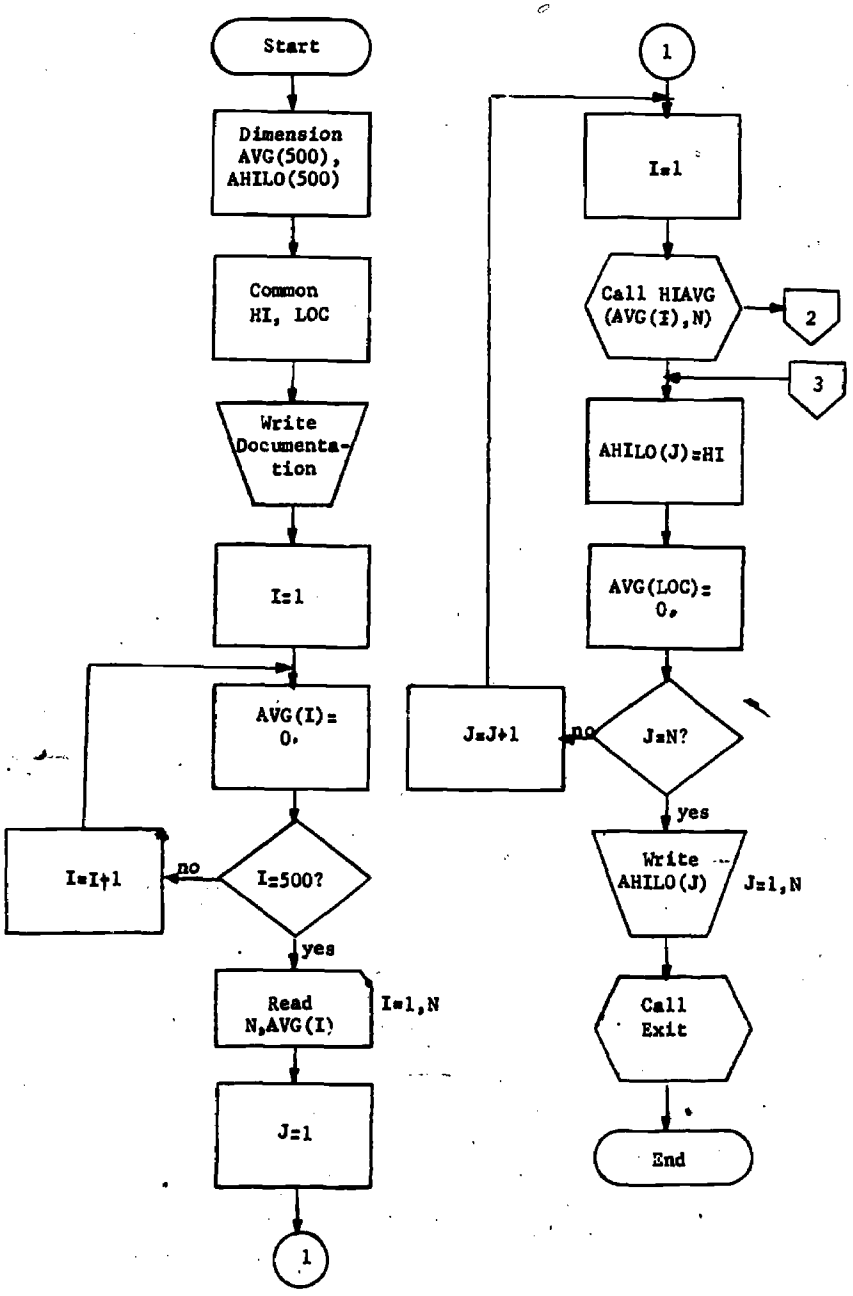
H. WRITING A SUBROUTINE SUBPROGRAM

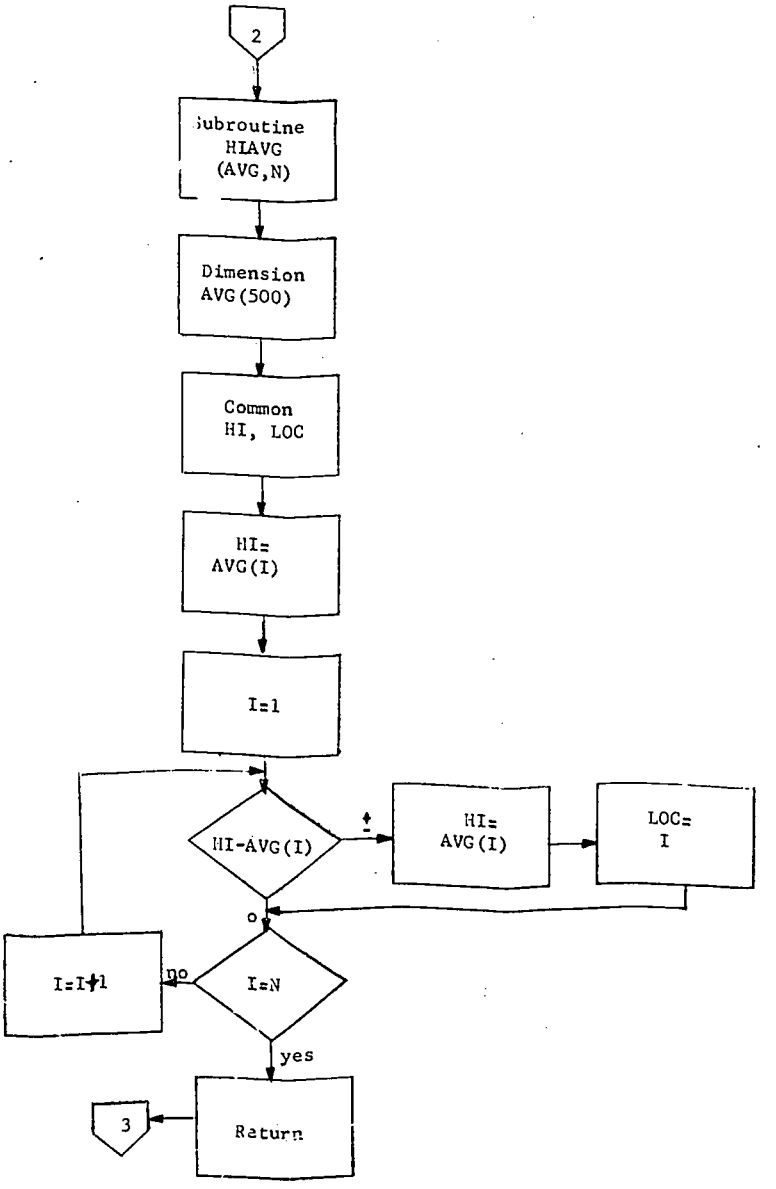
1. Problem: Write a subroutine that will find the highest number in a list of real numbers. The number of variables in the list must be flexible, but the program must accommodate at least 500 positive numbers, with a maximum value of 100.00.

2. Solution and flowcharts follow.

I. PROBLEM USING A SUBROUTINE

1. Problem: Sort a list of averages (real numbers) from the highest to the lowest using the subroutine HIAVG to find the largest in the array. Place HIAVG in a new array each time it is found, and remove it from the old array so that it is not used again.
2. Solution and flowcharts follow.




2
Subroutine HIAVG (AVG,N)
Dimension AVG(500)
Common HI, LOC
HI= AVG(I)
I=1
HI-AVG(I)
+
HI= AVG(I)
LOC= I
o
I=I+1
no
I=N
yes
Return
3

IBM

FORTRAN Coding Form

FORTRAN STATEMENT

```
#NAME  AHILO
       DIMENSION AVG(500), AHILO(500)
       COMMON HI, LOC
     1 WRITE(3,2)
     2 FORMAT (2X,'THIS PROGRAM WILL ARRANGE A LIST OF NUMBERS IN DESCEND
    1ING ORDER. UP TO 500 NUMBERS MAY BE ENTERED.',/,2X,'THE FIRST CARD
    2 ENTERED CONTAINS AN INTEGER SIGNIFYING THE SIZE OF THE LIST.'//)
       DO 40 I=1,500
    40 AVG(I)=0.
       READ(2,3)N,(AVG(I),I=1,N)
     3 FORMAT(I4/(8F10.2))
       DO 30 J=1,N
       I=1
       CALL HIAVG(AVG(I),N)
       AHILO(J)=HI
     7 AVG(LOC)=0.
    30 CONTINUE
       WRITE(3,4)(AHILO(J),J=1,N)
     4 FORMAT(2X,5F20.2)
       CALL EXIT
       END
```

IBM

FORTRAN Coding Form

SUBROUTINE HIAVG

FORTRAN STATEMENT

```
       SUBROUTINE HIAVG(AVG,N)
       DIMENSION AVG(500)
       COMMON HI,LOC
       HI=AVG(I)
       DO 20 I=1,N
       IF(HI-AVG(I))5,5,20
     5 HI=AVG(I)
       LOC=I
    20 CONTINUE
       RETURN
       END
```

```
PAGE   1
// JOB
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0001        0001         0000
V2 M04   ACTUAL  8K  CONFIG  8K
// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
*NAME AHILO
      DIMENSION AVG(500),AHILO(500)
       COMMON HI,LOC
    1 WRITE(3,2)
    2 FORMAT (2X,'THIS PROGRAM WILL ARRANGE A LIST OF NUMBERS IN DESCEND
     1ING ORDER. UP TO 500 NUMBERS MAY BE ENTERED.',/,2X,'THE FIRST CARD
     2 ENTERED CONTAINS AN INTEGER SIGNIFYING THE SIZE OF THE LIST.'//)
      DO 40 I=1,500
   40 AVG(I)=0.
      READ(2,3)N,(AVG(I),I=1,N)
    3 FORMAT(I4/(8F10.2))
      DO 30 J=1,N
      I=1
      CALL HIAVG(AVG( ),N)
      AHILO(J)=HI
    7 AVG(LOC)=0.
   30 CONTINUE
      WRITE(3,4)(AHILO(J),J=1,N)
    4 FORMAT(2X,5F20.2)
      CALL EXIT
      END
UNREFERENCED STATEMENTS-
 1
FEATURES SUPPORTED
 IOCS
CORE REQUIREMENTS FOR AHILO
 COMMON      4  VARIABLES   2008  PROGRAM    262
 END OF COMPILATION
// DUP
*STORE      WS  UA  AHILO
CART ID 0001   DB ADDR  200A   DB CNT   0012

PAGE   1
// JOB
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0001        0001         0000
V2 M04   ACTUAL  8K  CONFIG  8K
// FOR
*LIST SOURCE PROGRAM
      SUBROUTINE HIAVG(AVG,N)
      DIMENSION AVG(500)
      COMMON HI,LOC
      HI=AVG(1)
      DO 20 I=1,N
      IF(HI-AVG(I))5,5,20
    5 HI=AVG(I)
      LOC= I
   20 CONTINUE
      RETURN
      END
CORE REQUIREMENTS FOR HIAVG
 COMMON      4  VARIABLES      4  PROGRAM     60
 END OF COMPILATION
// DUP
*STORE      WS  UA  HIAVG
CART ID 0001   DB ADDR  201C   DB CNT   0005
PAGE   1
// JOB
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0001        0001         0000
V2 M04   ACTUAL  8K  CONFIG  8K
// XEQ AHILO
 THIS PROGRAM WILL ARRANGE A LIST OF NUMBERS IN DESCENDING ORDER. UP TO 500 NUMBERS MAY BE ENTERED.
 THE FIRST CARD ENTERED CONTAINS AN INTEGER SIGNIFYING THE SIZE OF THE LIST.

                100.00              98.50              97.00              96.33              95.00
                 93.00              92.00              90.00              87.50              83.33
                 82.05              82.00              81.50              81.50              74.00
                 74.00              70.00              69.00              63.75              63.50
                 62.00              61.00              59.75              52.00
// DUP
*DELETE             HIAVG
CART ID 0001   DB ADDR  201C   DB CNT   0005
*DELETE             AHILO
CART ID 0001   DB ADDR  200A   DB CNT   0012
```

FORTRAN LANGUAGE

Unit VIII

A. *DOUBLE SUBSCRIPTS*

1. To review, subscripts are used to identify a particular variable in an array or a list of variables.

2. When there is more than one list in the array double subscripts may be used.

3. The form of a double subscripted variable name is the name followed by a pair of parentheses enclosing the two subscripts separated by commas.
   LIST (I,J)

4. The rules for single subscripts apply also to double subscripted variables.

5. The first subscript signifies the position of the variable in the sub-array, and the second subscript signifies the number of the sub-array.
   LIST (5,2)
   The variable is the fifth member of the second list.

6. A double-subscripted DIMENSION statement must be placed in the program if a double subscripted array is used.
   DIMENSION LIST (10,4)
   The statement would reserve storage for 40 values of LIST, 4 arrays of 10 values each.

B. *MATRICES*

1. A group of numbers arranged in a rectangular array is known as a matrix.

2. Double subscripted notation is used to indicate the positions of variables by row number and column number. A 3 x 2 matrix may be indicated as:

   A (1,1) A (1,2)
   A (2,1) A (2,2)
   A (3,1) A (3,2)

   The 1st subscript refers to the *row* and the second to the *column* of the array.

3. Within the computer the two-dimensional array is stored in single array form as follows. A (1,1), A (2,1), A (3,1) A (1,2), A (2,2), A (3,2). Each column is laid out in order before a new column is begun. Notice that the 1st subscript is completely cycled before the second subscript moves up one.

4. 1130 FORTRAN also allows 3-dimensional array notation.
   a. Triple subscripts are used to identify positions.
   b. The computer treats the 3-dimensional array as an array divided into sub-arrays, each of which is also divided into a 2-dimensional sub-array.
   c. The first subscript is advanced most rapidly, each time it reaches its limit the second subscript is advanced and when the second reaches its limit the 3rd is advanced.

C. *DOUBLE SUBSCRIPTED INPUT/OUTPUT STATEMENTS*

1. The FORMAT of the self-indexed input/output is:
   READ (2,1)((AGE(I,J),I=1,5),J=1,40)

This FORMAT is precise, two open parentheses, the variable name, open parenthesis, two dummy subscripts separated by a comma, closed parenthesis, comma, one index definition, closed parenthesis, comma, second index definition, closed parenthesis.

2. The double-indexing works like nested DO-loops, the innermost index cycles completely, then the outermost is stepped up by 1, and the innermost cycles completely again before the outermost steps up again.

3. The READ illustrated above will read the quantities in the following order:

AGE (1,1), AGE (2,1), AGE (3,1), ... AGE (5,1),
AGE (1,2), AGE (2,2), ... AGE (5,2),
AGE (1,3), AGE (2,3), ... etc.

4. To reverse the order of reading an array the index definitions may be reversed.

READ (2,1)((AGE(I,J),J=1,4),I=1,5)

Will read values in the order:

AGE (1,1), AGE (1,2), AGE (1,3), AGE (1,4), AGE (1,5)
AGE (2,1), AGE (2,2), ... AGE (2,5)
AGE (3,1), AGE (3,2), ... AGE (3,5)

5. To write an array it is usually convenient to print a row at a time so the second subscript should be cycled most rapidly. The same sequence as listed on the preceding page in the READ would be used with the word WRITE.

WRITE (3,1)(AGE(I,J),J=1,4),I=1,5)

6. All variations of listing which apply to self-indexed input/output statements also apply if these variables are doubled subscripted.

D. *A PROBLEM USING DOUBLE SUBSCRIPTED VARIABLES*

1. Problem — The sum of two matrices may be illustrated as follows:

k(1,1)	k(1,2)	k(1,3)
k(2,1)	k(2,2)	k(2,3)
k(3,1)	k(3,2)	k(3,3)

+

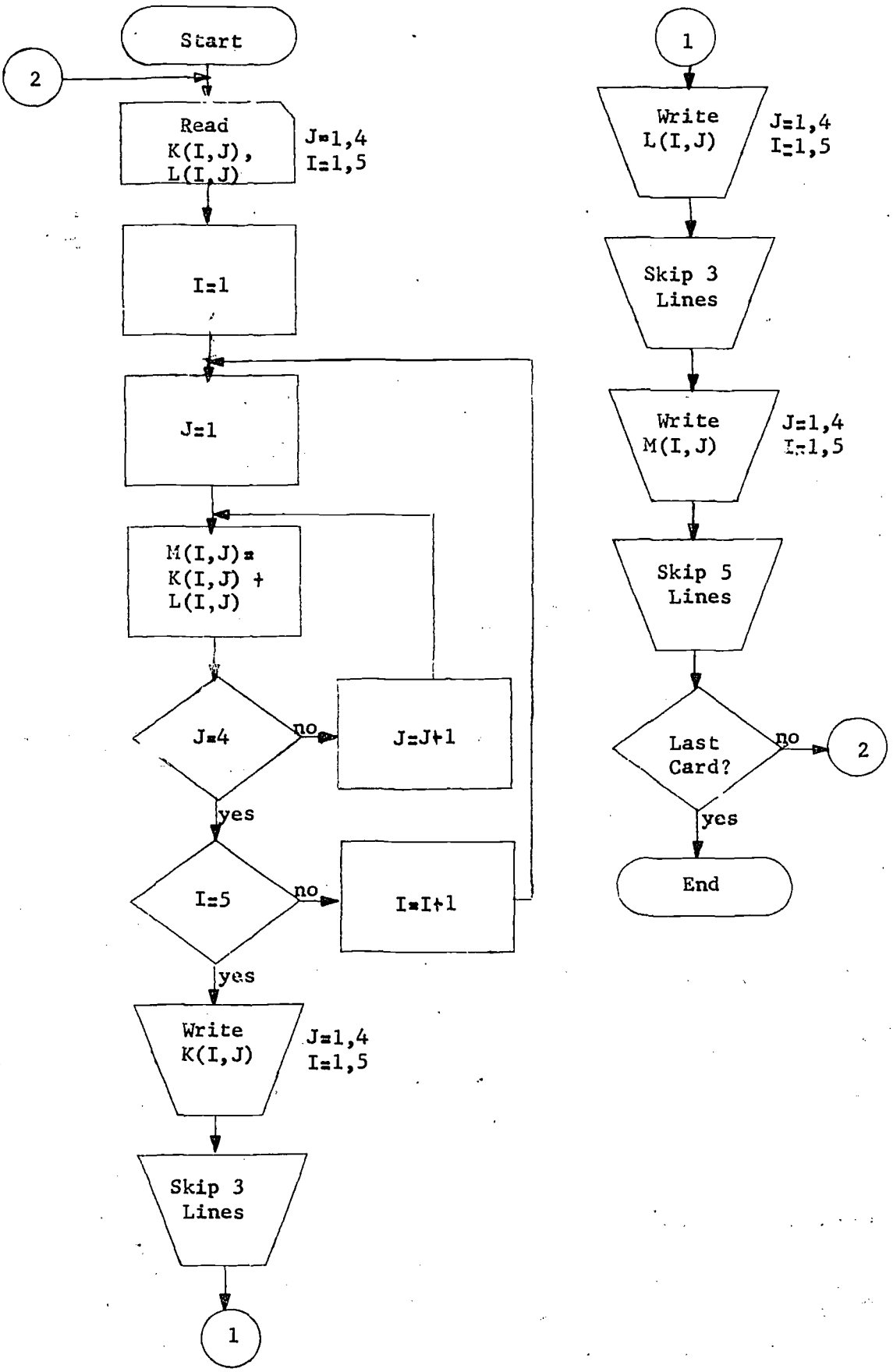
l(1,1)	l(1,2)	l(1,3)
l(2,1)	l(2,2)	l(2,3)
l(3,1)	l(3,2)	l(3,3)

=

k(1,1)+l(1,1)	k(1,2)+l(1,2)	k(1,3)+l(1,3)
k(2,1)+l(2,1)	k(2,2)+l(2,2)	k(2,3)+l(2,3)
k(3,1)+l(3,1)	k(3,2)+l(3,2)	k(3,3)+l(3,3)

Write a program that will add two 5 X 4 matrices.

2. Flowchart


Start
2
Read
K(I,J),
L(I,J)
J=1,4
I=1,5
I=1
J=1
M(I,J)=
K(I,J) +
L(I,J)
J=4
no
J=J+1
yes
I=5
no
I=I+1
yes
Write
K(I,J)
J=1,4
I=1,5
Skip 3
Lines
1
1
Write
L(I,J)
J=1,4
I=1,5
Skip 3
Lines
Write
M(I,J)
J=1,4
I=1,5
Skip 5
Lines
Last
Card?
no
2
yes
End

3. Solution

IBM FORTRAN Coding Form

ADD TWO MATRICES

FORTRAN STATEMENT

```
C       PROGRAM TO ADD TWO MATRICES AND PRINT OUT RESULTS.
        DIMENSION K(5,4), L(5,4), M(5,4)
  100   READ(2,1)((K(I,J),J=1,4),I=1,5),((L(I,J),J=1,4),I=1,5)
    1   FORMAT(4I10)
        DO 3 I=1,5
        DO 3 J=1,4
        M(I,J)=K(I,J)+L(I,J)
    3   CONTINUE
        WRITE(3,2)((K(I,J),J=1,4),I=1,5)
    2   FORMAT(2X,4I10)
        WRITE(3,5)
    5   FORMAT(///)
        WRITE(3,2)((L(I,J),J=1,4),I=1,5)
        WRITE(3,5)
        WRITE(3,2)((M(I,J),J=1,4),I=1,5)
        WRITE(3,4)
    4   FORMAT(///////)
        GO TO 100
  111   CALL EXIT
        END
```

4. Printout

```
PAGE    1

// JOB

LOG DRIVE    CART SPEC    CART AVAIL    PHY DRIVE
  0000         0001         0001          0000

V2 M04    ACTUAL  8K   CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
C       PROGRAM TO ADD TWO MATRICES AND PRINT OUT RESULTS.
        DIMENSION K(5,4), L(5,4), M(5,4)
  100   READ(2,1)((K(I,J),J=1,4),I=1,5),((L(I,J),J=1,4),I=1,5)
    1   FORMAT(4I10)
        DO 3 I=1,5
        DO 3 J=1,4
        M(I,J)=K(I,J)+L(I,J)
    3   CONTINUE
        WRITE(3,2)((K(I,J),J=1,4),I=1,5)
    2   FORMAT(2X,4I10)
        WRITE(3,5)
    5   FORMAT(///)
        WRITE(3,2)((L(I,J),J=1,4),I=1,5)
        WRITE(3,5)
        WRITE(3,2)((M(I,J),J=1,4),I=1,5)
        WRITE(3,4)
    4   FORMAT(///////)
        GO TO 100
  111   CALL EXIT
        END
```

```
UNREFERENCED STATEMENTS
  111

FEATURES SUPPORTED
  IOCS

CORE REQUIREMENTS FOR
  COMMON       0  VARIABLES     126  PROGRAM     288

  END OF COMPILATION

// XEQ
```

-4	8	3	9
6	-1	2	3
7	4	1	3
-2	8	4	9
6	5	4	3

-3	2	1	0
8	7	6	5
-5	-6	-7	-8
4	5	8	3
1	2	4	3

-7	10	4	9
14	6	8	8
2	-2	-6	-5
2	13	12	12
7	7	8	6

E. MORE DO-LOOP PROBLEMS

1. Although a plotter is available for the IBM 1130, satisfactory graphs can be produced by using the double subscripted variables and nested DO-loop facilities of the computer.

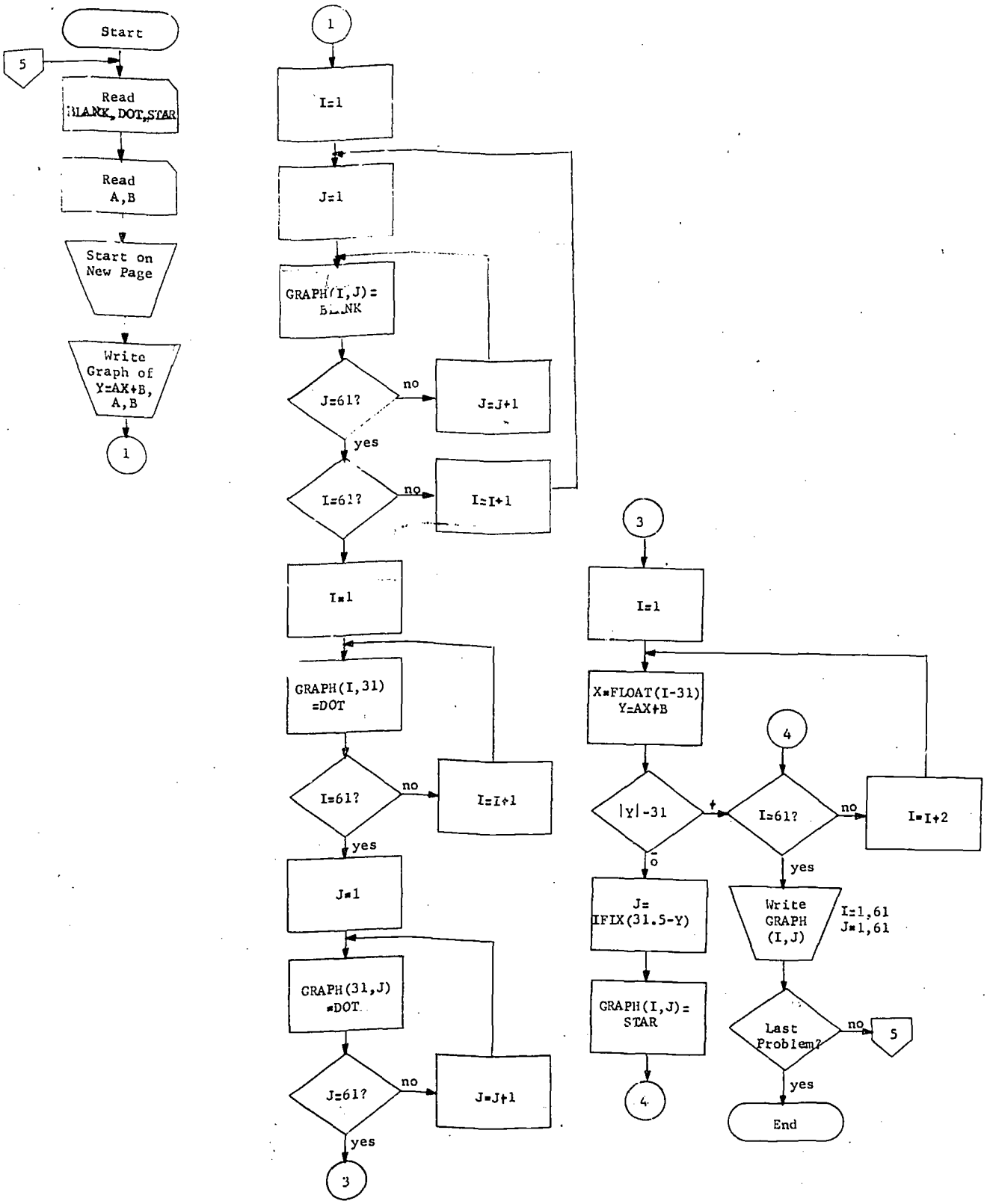
2. Problems
   a. Write a program to produce a graph of a linear function of the form $y = ax + b$, for values of $-30 \leq x \leq +30$ at intervals of 2.

      NOTE: BLANK, DOT, STAR are 3 alphameric variables which are entered on a card, the first column is blank, the second column contains a period, and the third contains an asterisk.

      (1) Plan for solution.
      - (a) Read alphameric characters and variables a and b.
      - (b) Fill a 61 x 61 array with blanks.
      - (c) Fill column 31 with dots.
      - (d) Fill row 31 with dots.
      - (e) Under control of the index of a loop (I) set up values of x from -30 to +30 at intervals of 2.
      - (f) Compute y from each value of x.
      - (g) If y is greater than |31| the point will not be on the graph, so try another value of x.
      - (h) If y is less than |31| determine its location on the graph, and find the subscript corresponding to this location.

        NOTE: Adding .5 to the value of y before changing it to an integer to represent the subscript has the effect of rounding off the number e.g., if y = 7.3 adding .5 before the IFIX yields 7.8 which is truncated to 7, if y = 7.6 adding .5 gives 8.1 which is truncated to 8 by the IFIX.
      - (i) Set the point determined by (x,y) to a star.
      - (j) When all possible values have been calculated write the GRAPH array row-wise.

(2) Flowchart.


Start
5
Read
BLANK, DOT, STAR
Read
A,B
Start on
New Page
Write
Graph of
Y=AX+B,
A,B
1
1
I=1
J=1
GRAPH(I,J)=
BLANK
J=61?
no
J=J+1
yes
I=61?
no
I=I+1
I=1
GRAPH(I,31)
=DOT
I=61?
no
I=I+1
yes
J=1
GRAPH(31,J)
=DOT
J=61?
no
J=J+1
yes
3
3
I=1
X=FLOAT(I-31)
Y=AX+B
|Y|-31
t
o
J=
IFIX(31.5-Y)
GRAPH(I,J)=
STAR
4
4
I=61?
no
I=I+2
yes
Write
GRAPH
(I,J)
I=1,61
J=1,61
Last
Problem?
no
5
yes
End

(3) Coding sheets

IBM FORTRAN Coding Form

GRAPH OF Y = AX + B

1 2

	FORTRAN STATEMENT
* ONE	WORD INTEGERS
	INTEGER BLANK, DOT, STAR, GRAPH(61, 61)
100	READ(2, 1) BLANK, DOT, STAR
1	FORMAT(3A1)
	READ(2, 2) A, B
2	FORMAT(2F5.2)
	WRITE(3, 3)
3	FORMAT(1H1)
	WRITE(3, 4) A, B
4	FORMAT(2X, 'GRAPH OF Y = AX + B ,A = ', F5.2, 5X, 'B = ', F5.2, //)
	DO 5 I = 1, 61
	DO 5 J = 1, 61
5	GRAPH(I, J) = BLANK
	DO 6 I = 1, 61
6	GRAPH(I, 31) = DOT
	DO 7 J = 1, 61
7	GRAPH(31, J) = DOT
	DO 9 I = 1, 61, 2
	X = FLOAT(I - 31)
	Y = A * X + B
	IF(ABS(Y) - 31.) 8, 8, 9
8	J = IFIX(31.5 - Y)
	GRAPH(I, J) = STAR
9	CONTINUE

IBM FORTRAN Coding Form

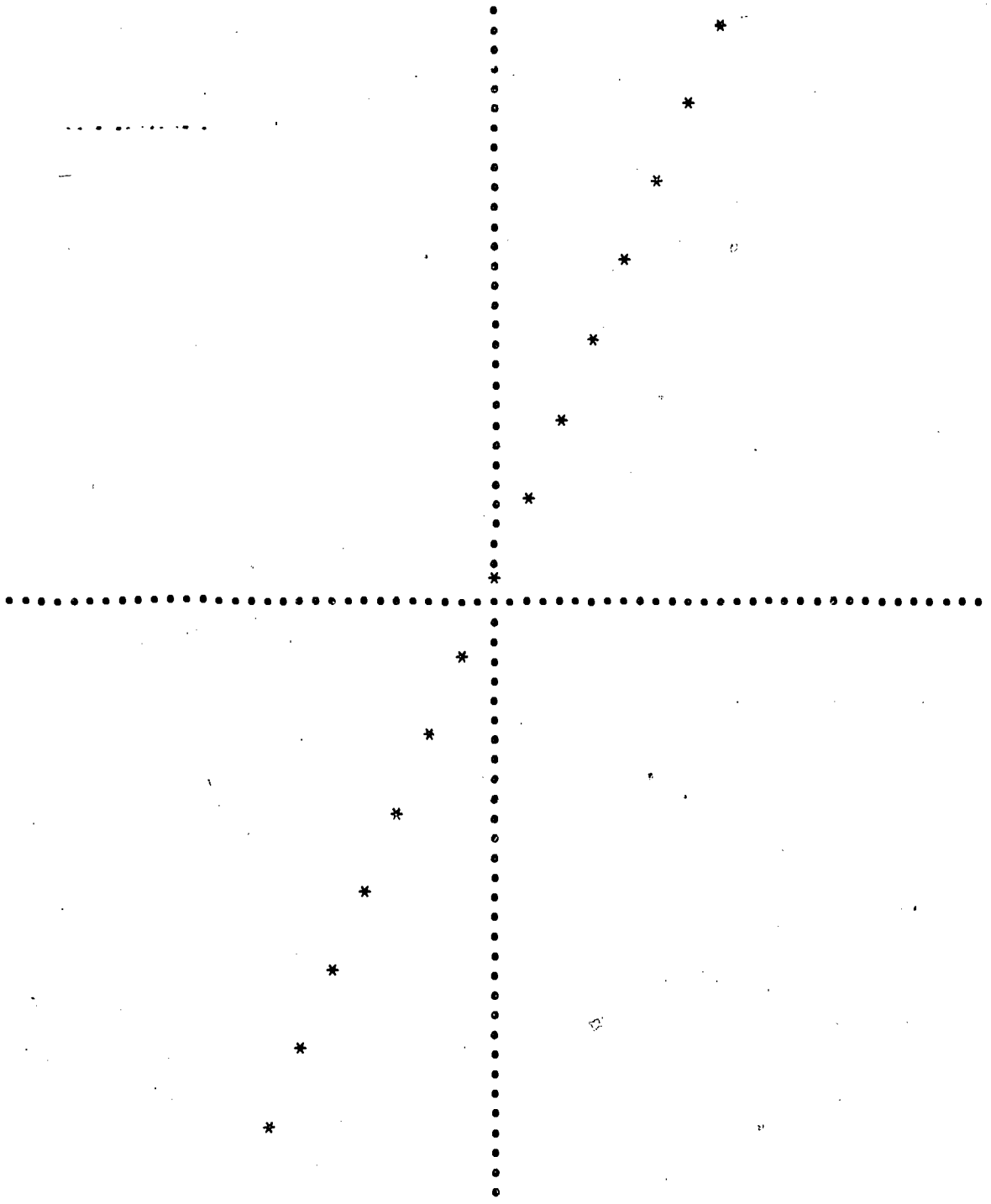
2 2

	FORTRAN STATEMENT
	WRITE(3, 10)((GRAPH(I, J), I = 1, 61), J = 1, 61)
10	FORMAT(2X, 61A1)
	GO TO 100
99	CALL EXIT
	END

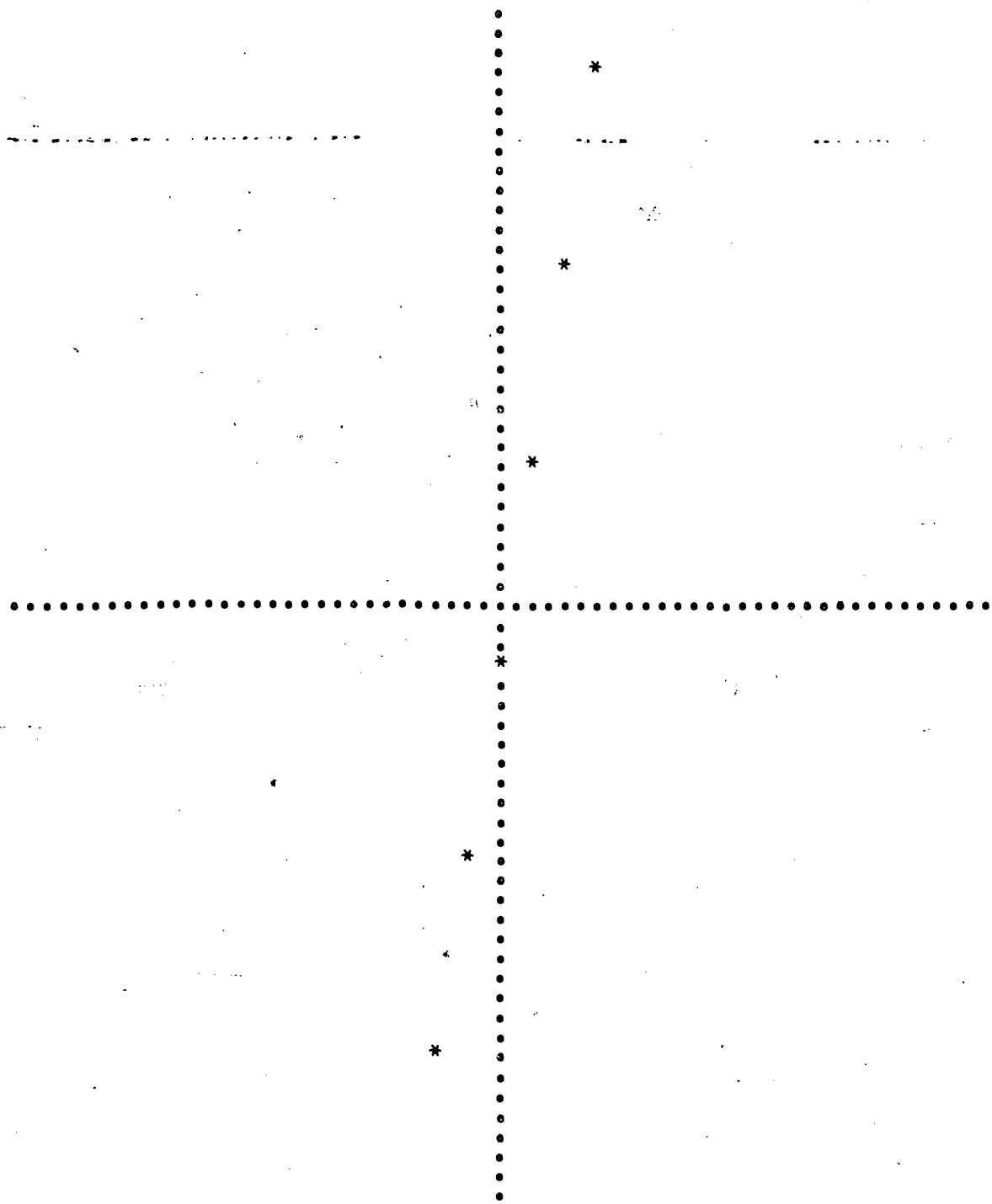
(4) Printout

```
PAGE    1
// JOB
LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0001        0001        0000
V2 M04   ACTUAL  8K  CONFIG  8K
// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
      INTEGER BLANK,DOT,STAR,GRAPH(61,61)
100 READ(2,1)BLANK,DOT,STAR
  1 FORMAT(3A1)
    READ(2,2)A,B
  2 FORMAT(2F5.2)
    WRITE(3,3)
  3 FORMAT(1H1)
    WRITE(3,4)A,B
  4 FORMAT(2X,'GRAPH OF Y = AX + B ,A = ',F5.2,5X,'B = ',F5.2,//)
    DO 5 I=1,61
    DO 5 J=1,61
  5 GRAPH(I,J)=BLANK
    DO 6 I=1,61
  6 GRAPH(I,31)=DOT
    DO 7 J=1,61
  7 GRAPH(31,J)=DOT
    DO 9 I=1,61,2
    X=FLOAT(I-31)
    Y=A*X+B
    IF(ABS(Y)-31.)8,8,9
  8 J=IFIX(31.5-Y)
    GRAPH(I,J)=STAR
  9 CONTINUE
    WRITE(3,10)((GRAPH(I,J),I=1,61),J=1,61)
 10 FORMAT(2X,61A1)
    GO TO 100
 99 CALL EXIT
    END
UNREFERENCED STATEMENTS
 99

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    3738  PROGRAM     284

 END OF COMPILATION
// XEQ
```

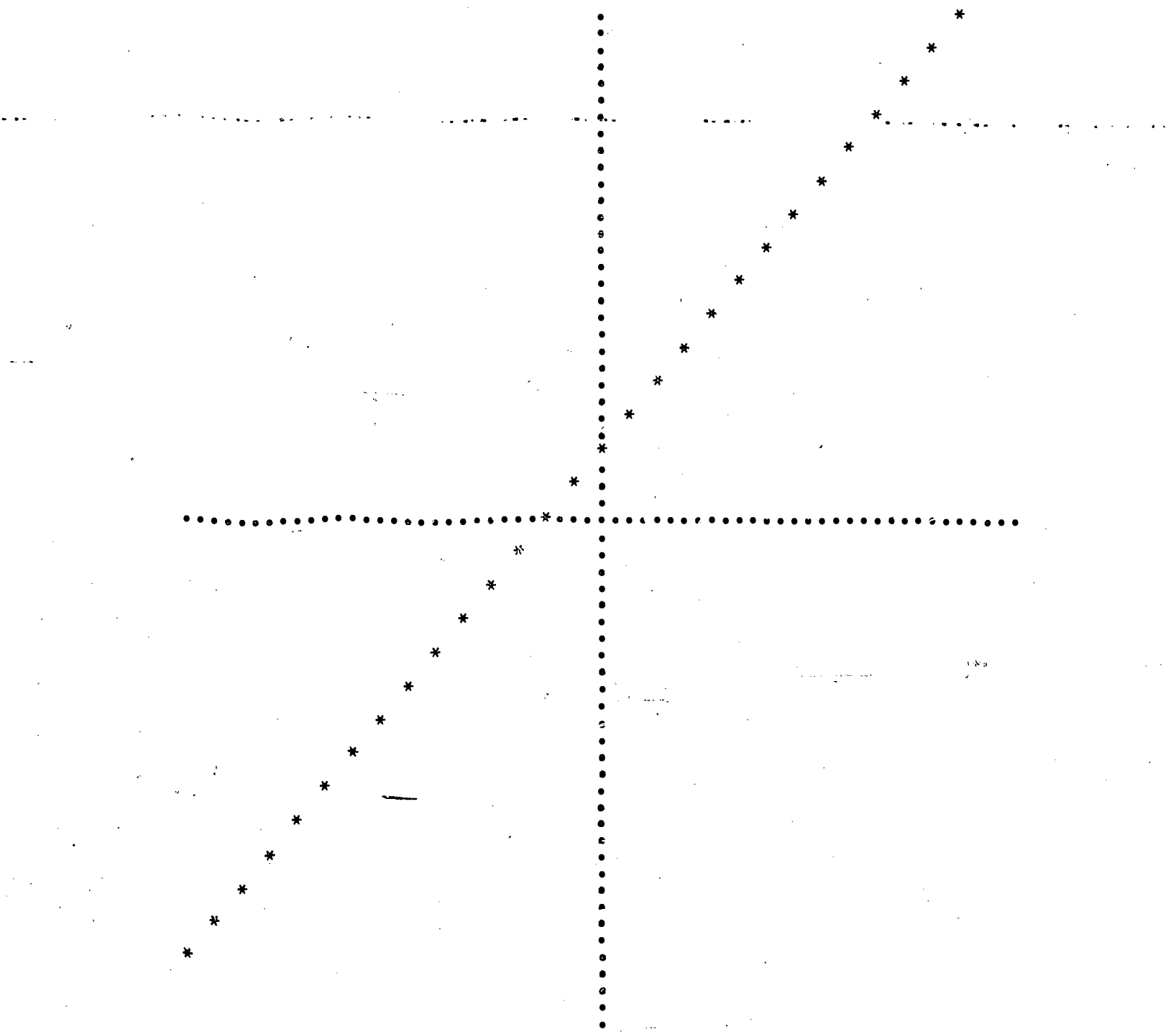
GRAPH OF Y = AX + B ,A = 2.00 B = 1.00



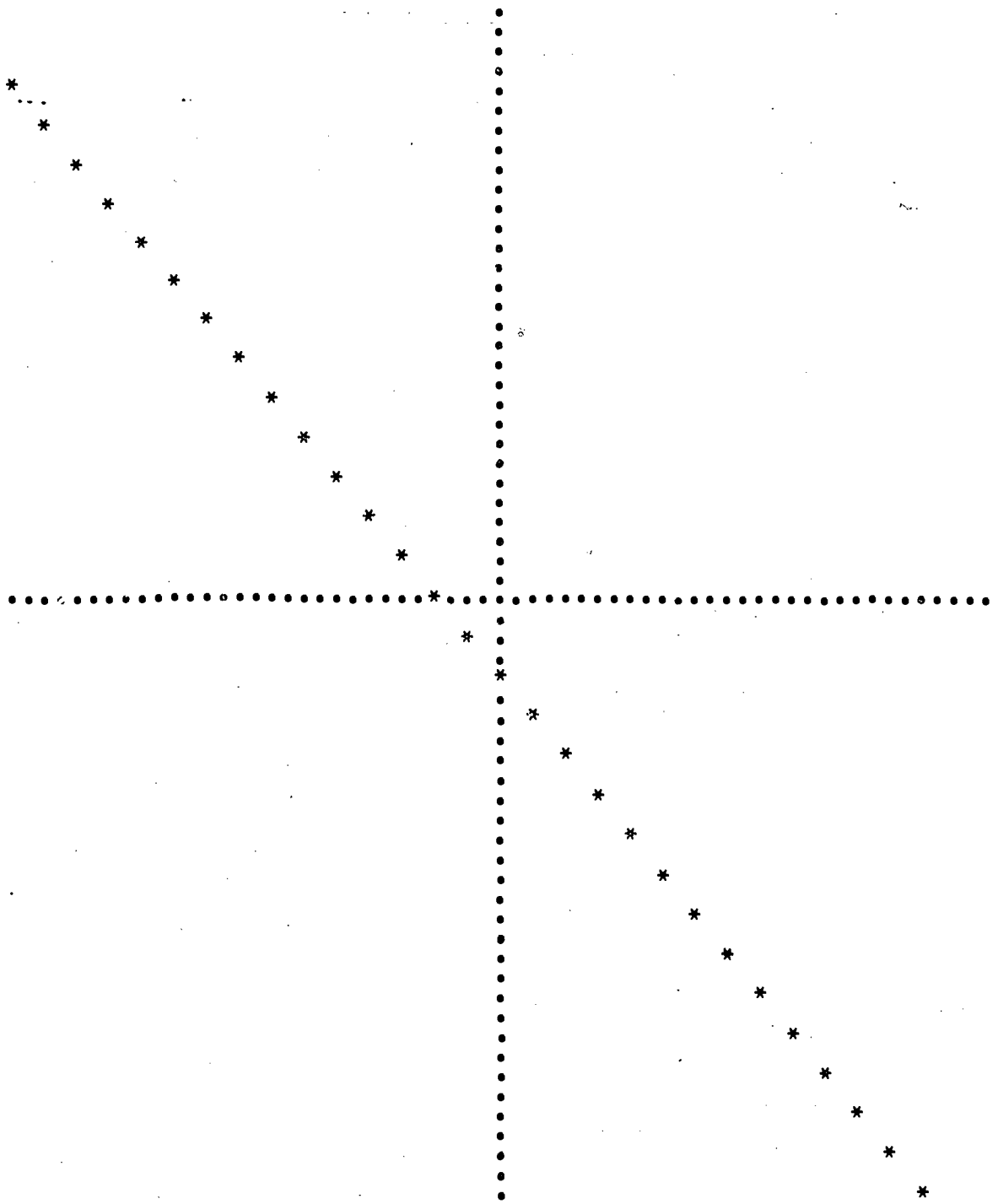
GRAPH OF Y = AX + B ,A = 5.00 B = -3.00



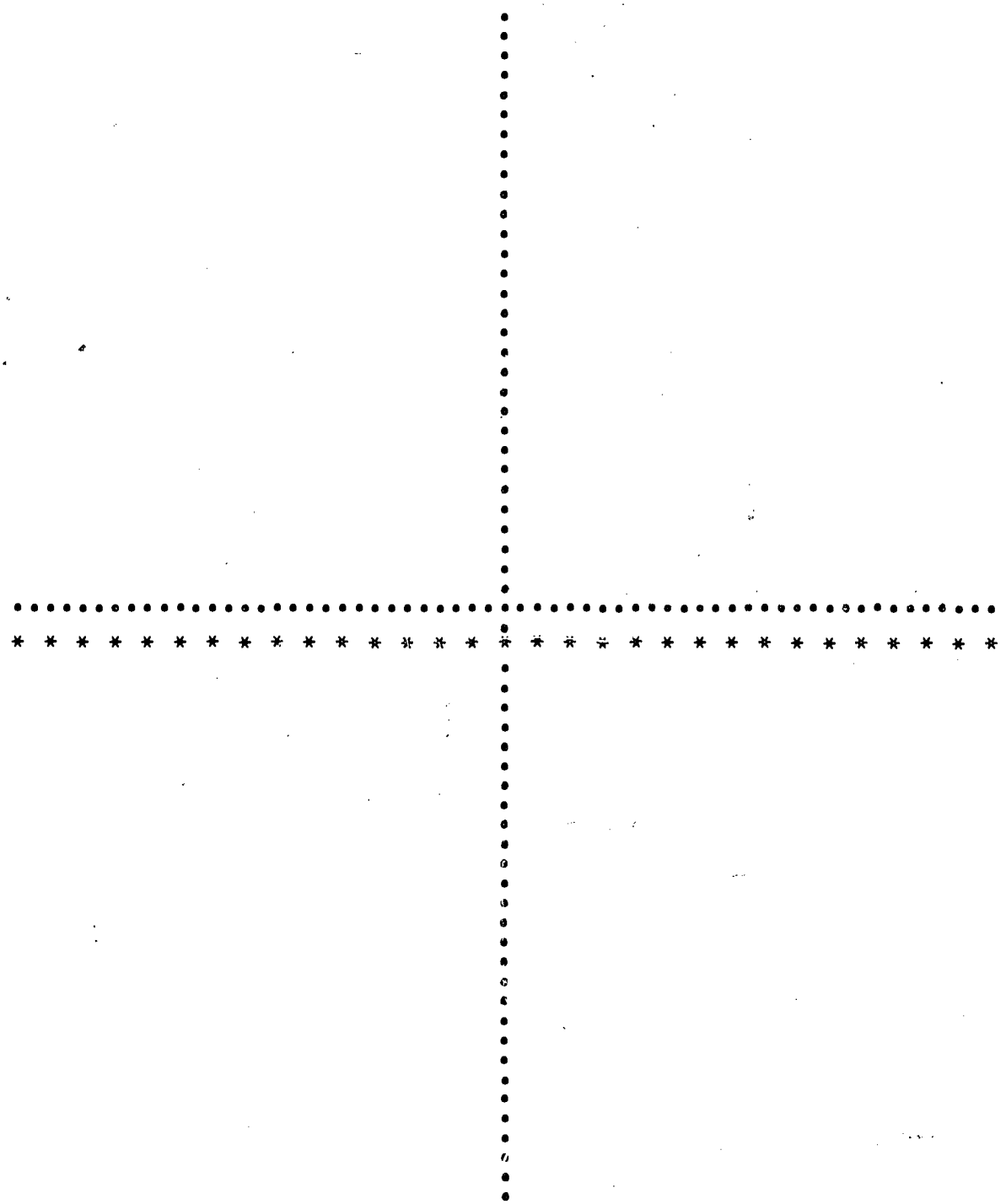
GRAPH OF Y = AX + B ,A = 1.00 B = 4.00



GRAPH OF Y = AX + B ,A = -1.00 B = -4.00



GRAPH OF Y = AX + B ,A = 0.00 B = -2.00



b. Using the same technique as in the preceding problem plot the graphy of y = sin (x) from 0 radians to 11.5 radians at intervals of .2 radian.

NOTE: Since the sine never exceeds|1|, do not use a square array, a rectangular array will allow 120 spaces along the x-axis. A 21 row array would allow for values of the sine at every tenth from -1 to -1.

(1) Solution:

Follow the same plan as in the preceding problem, first reserving space for the array, then filling it with blanks, and then filling the axes with dots. Compute values for the sine of x for every value of x from 0 radians to 11.5 radians. Round off the value of the sine to the nearest tenth and use this value to find its position in the array. When all values are computed print the array row-wise.

(2) Coding form.

IBM FORTRAN Coding Form

GRAPH OF Y=SIN(X) 1 2

FORTRAN STATEMENT

```
N ONE  WORD INTEGERS
       INTEGER BLANK, DOT, AST, FUNC(118,21), X, Y
       READ (2,12)BLANK,DOT,AST,X,Y
    12 FORMAT (5A1)
       DO 2 L=1,118
       DO 2 K=1,21
     2 FUNC(L,K)=BLANK
       DO 5 L=2,117
     5 FUNC(L,11)=AST
       DO 10 K=1,21
    10 FUNC(3,K)=AST
       DO 20 L=3,118,2
       Z=L-3
       A=Z/10.
       B=SIN(X)+.05
       K=11-IFIX(B*10.)
       FUNC(L,K)=DOT
    20 CONTINUE
       FUNC(1,11)=X
       FUNC(3,1)=Y
       FUNC(3,21)=Y
       FUNC(118,11)=X
       WRITE (3,13)
    13 FORMAT (2X,'THIS IS THE GRAPH OF Y=SIN(X)',///)
```

IBM FORTRAN Coding Form

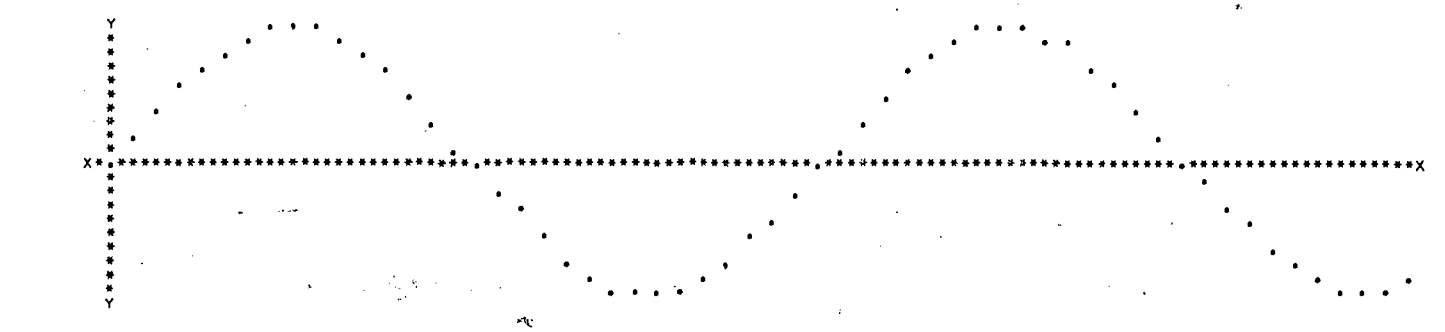
2 2

FORTRAN STATEMENT

```
       WRITE (3,14)((FUNC(L,K),L=1,118),K=1,21)
    14 FORMAT (2X,118A1)
       CALL EXIT
       END
```

(3) Printed output.

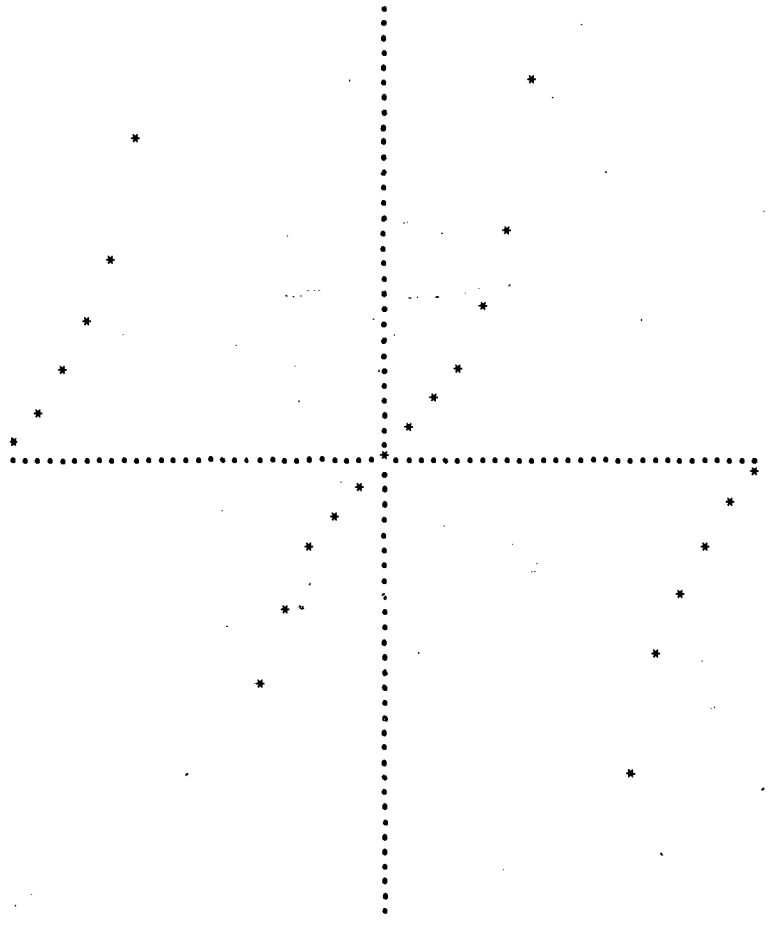
```
PAGE   1
// JOB
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0001        0001         0000

V2 M04   ACTUAL  8K  CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
      INTEGER BLANK,DOT,AST,FUNC(118,21),X,Y
      READ (2,12)BLANK,DOT,AST,X,Y
   12 FORMAT (5A1)
      DO 2 L=1,118
      DO 2 K=1,21
    2 FUNC(L,K)=BLANK
      DO 5 L=2,117
    5 FUNC(L,11)=AST
      DO 10 K=1,21
   10 FUNC(3,K)=AST
      DO 20 L=3,118,2
      Z=L-3
      A=Z/10.
      B=SIN(A)+.05
      K=11 -IFIX(B*10.)
      FUNC(L,K)=DOT
   20 CONTINUE
      FUNC(1,11)=X
      FUNC(3,1)=Y
      FUNC(3,21)=Y
      FUNC(118,11)=X
      WRITE (3,13)
   13 FORMAT (2X,'THIS IS THE GRAPH OF Y=SIN(X)',///)
      WRITE (3,14)((FUNC(L,K),L=1,118),K=1,21)
   14 FORMAT (2X,118A1)
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON      0  VARIABLES   2498  PROGRAM    286

 END OF COMPILATION

// XEQ
 THIS IS THE GRAPH OF Y=SIN(X)
```



c. Output - $y = \tan(x)$

```
PAGE   1
// JOB
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0001        0001         0000

V2 M04   ACTUAL  8K  CONFIG  8K

// FOR
*IOCS(CARD,DISK,1132PRINTER,TYPEWRITER,KEYBOARD)
*LIST SOURCE PROGRAM
* ONE WORD INTEGERS
      INTEGER BLANK,DOT,STAR,GRAPH(61,61)
  100 READ(2,1)BLANK,DOT,STAR
    1 FORMAT(3A1)
      WRITE(3,3)
    3 FORMAT(1H1)
      DO 5 I=1,61
      DO 5 J=1,61
    5 GRAPH(I,J)=BLANK
      DO 6 I=1,61
    6 GRAPH(I,31)=DOT
      DO 7 J=1,61
    7 GRAPH(31,J)=DOT
      DO 9 I=1,61,2
      X=FLOAT(I-31)/10.
      IF(COS(X))30,9,30
   30 Y=SIN(X)/COS(X)
      IF(ABS(Y)-3.1)8,8,9
    8 J=31-(IFIX(Y*10.))
      GRAPH(I,J)=STAR
    9 CONTINUE
      WRITE(3,10)((GRAPH(I,J),I=1,61),J=1,61)
   10 FORMAT(2X,61A1)
      GO TO 100
   99 CALL EXIT
      END
```

```
UNREFERENCED STATEMENTS
 99

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    3734  PROGRAM     258

 END OF COMPILATION

// XEQ
```



F. SPECIAL FORTRAN SUPPLIED SUBROUTINES

1. SLITE and SLITET — These subroutines simulate the testing of lights that can be turned on and off and tested by program statements. The lights do not actually exist on the 1130, but the program may use the subroutines to produce the same effects as if they were present.
   a. To make use of the SLITE subroutine the statement,
      CALL SLITE (I)
      is used. "I" is an integer variable which may only have the value of 0, 1, 2, 3 or 4. If I = 0, all 4 lights are off; if I = 1, light 1 will be turned on; and so on.
   b. To test the status of the test lights the statement,
      CALL SLITET (I,N)
      might be used "I" is an integer variable which may equal 1,2,3 or 4 and N is an integer variable which may equal 1 or 2. This second variable N will be set to 1 if the light (I) is on and 2 if it is off. If the sense light that was tested was on, it would then be turned off after the execution of the subroutine.

2. Overflow and Underflow
   a. When a number becomes too large to be handled by the computer, a condition called overflow exists. A number that is too small, a decimal fraction with too many zeros between the decimal and the 1st significant digit, causes underflow. These conditions may be tested by a subroutine called OVERFL.
   b. The form of the statement is —
      CALL OVERFL (I)
      where I is an integer variable name. When an overflow exists, I is set to 1, when an underflow exists, I is set to 3, when neither condition exists, I is set to 2.
   c. Execution of the subroutine includes resetting the computer to a no overflow condition.
   d. A computed GO TO is usually used after an overflow to direct the computer to various error messages.

3. Division by Zero
   a. An attempt to divide by zero will cause the computer to indicate a "divide check." However, the program is not interrupted, but continues as if the division had been completed.
   b. To check the status of this "divide check" indicator, a subroutine DVCHK (I) has been supplied by the manufacturer.
   c. The statement is
      CALL DVCHK (I)
      where I is an integer variable which will be set to 1 if the indicator is on and 2 if it is off.
   d. As is the case with OVERFL, the subroutine resets the indicator to an off condition.

BIBLIOGRAPHY

Textbooks

Allendoerfer, C.B. and Oakley, C.O. *Principles of Mathematics*. second edition. New York: McGraw-Hill Book Company, 1963.

CHEMS Chemistry, *An Experimental Science*. San Francisco: W.H. Freeman and Company, 1963.

Dolciani, M.P. et al. *Modern Algebra and Trigonometry*. Boston: Houghton Mifflin Company, 1963.

Modern Introductory Analysis. Boston: Houghton Mifflin Company, 1964.

PSSC Physics, second edition. Boston: D.C. Heath and Company, 1965.

Sienko, M.J. and Plane, R.A. *Chemistry*, second edition. New York: McGraw-Hill Book Company, 1961.

Smoot, R.C. et al. *Chemistry, A Modern Course*. Columbus, Ohio: C.E. Merrill Books, Inc., 1965.

Stollberg, and Hill, F.F. *Physics, Fundamentals and Frontiers*. Boston: Houghton Mifflin Company, 1965.

Welchons, A.M. et al. *Modern Trigonometry*. Boston: Ginn and Company, 1962.

Resource and Reference Books and Pamphlets

Adler, I. *Thinking Machines*. New York: John Day Company, 1961.

Andree, R.V. *Computer Programming and Related Mathematics*. New York: John Wiley and Sons, Inc., 1967

Association for Educational Data Systems; Bushnell, D.A. and Allen, D.W., editors. *The Computer in American Education*. New York: John Wiley and Sons, Inc., 1967.

Bork, A.M. *Using the IBM 1130*. Reading, Massachusetts: Addison and Wesley, 1968.

Crowder, N.A. *The Arithmetic of Computers*. New York: Doubleday and Company, Inc., 1960.

Dodes, I.A. and Greitzer, S.L. *Numerical Analysis*. New York: Hayden Book Company, 1964.

Dorn, W.S. and Greenberg, H.J. *Mathematics and Computing*. New York: John Wiley and Sons, Inc., 1967.

Fink, D.G. *Computers and the Human Mind*. New York: Doubleday and Company, Inc., 1966.

W.H. Freeman and Company. *Information* (a "Scientific American" book). San Franisco: The Company, 1966.

Gardner, M. *Logic Machines and Diagrams*. New York: McGraw-Hill Book Company, 1958.

Gruenberger, F.J. and McCracken, D.*Introduction to Electronic Computers: Problem Solving with the IBM 1620* New York: John Wiley and Sons, Inc., 1963.

Gruenberger, F. and Jaffray, G. *Problems for Computer Solution*. New York: John Wiley and Sons, Inc., 1963.

IBM DPD Education Development, Education Center. *FORTRAN for the IBM 1130*. Endicott, New York: 1965.

Jacobowitz, H. *Computer Arithmetic*. New York: Hayden Book Company, 1964.

Kovach, L.D. *Computer Oriented Mathematics*. San Francisco: Holden-Day Inc., 1964.

Larrson, D. *Equalities and Approximations with Fortran Programming*. New York: John Wiley and Sons, Inc., 1963.

Louden, R.K. *Programming the IBM 1130 and 1800*. New Jersey, Prentice Hall, Inc.

Lovis, F.B. *Computers 1*. Boston: Houghton Mifflin Company, 1964.

Computers 2. Boston: Houghton Mifflin Company, 1964.

McCalla, T.R. *Introduction to Numerical Methods and Fortran Programming*. New York: John Wiley and Sons, Inc., 1967.

McCracken, D.D. *A Guide to Fortran Programming*. New York: John Wiley and Sons, Inc., 1961.

A Guide to Fortran IV Programming. New York: John Wiley and Sons, Inc. 1965.

Digital Computer Programming. New York: John Wiley and Sons, Inc., 1957.

Mullish, H. *Modern Programming: FORTRAN IV*. Reading, Massachusetts: Blaisdel Publishing Company, Inc., 1968.

Murphy, J.S. *Basics of Digital Computers*. New York: Hayden Book Company, 1958.

National Council of Teachers of Mathematics. *Computer Facilities for Mathematics Instruction*. Washington D.C.: The Council, 1967.

Computer Oriented Mathematics: An Introduction for Teachers. Washington, D.C.: The Council, 1963.

Introduction to An Algorithmic Language (Basic). Washington, D.C.: The Council, 1968. On order

National Science Teachers Association. *Computers-Theory and Uses*. Washington, D.C., 1964.

Organick, E.I. *A Fortran IV Primer*. Reading, Massachusetts: Addison Wesley Publishing Company, Inc., 1966.

School Mathematics Study Group. *Algorithmic Computation and Mathematics*, Revised Edition. Pasadena, California: A.C. Vroman, Inc., 1966.

Algorithms, Computation and Mathematics (Fortran Supplement). Pasadena, California: A.C. Vroman, Inc., 1966.

Siegel P. *Understanding Digital Computers*. New York: John Wiley and Sons, Inc., 1961.

Stibitz, G.R. and Larrivee, J.A. *Mathematics and Computers*. New York: McGraw-Hill Book Company, 1957.

von Neumann, J. *The Computer and the Brain*. New Haven: Yale University Press, 1958.

Young, F. *Digital Computers and Related Mathematics*. Boston: Ginn and Company., 1961.

IBM Manuals

	Form Nos.
IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide.	C26-3717-3
IBM 1130 Functional Characteristics.	A26-5881-4
IBM 1130/1800 Basic FORTRAN IV Language.	C26-3715-3
IBM 1130 Subroutine Library.	C26-5929-4
Catalog of Programs for IBM 1130 Computing System and IBM 1800 Data Acquisition and Control System.	C20-1630-5
1130 Scientific Subroutine Package (1130-CM-02X) Programmer's Manual.	H20-0252-3

PROBLEM SUPPLEMENT

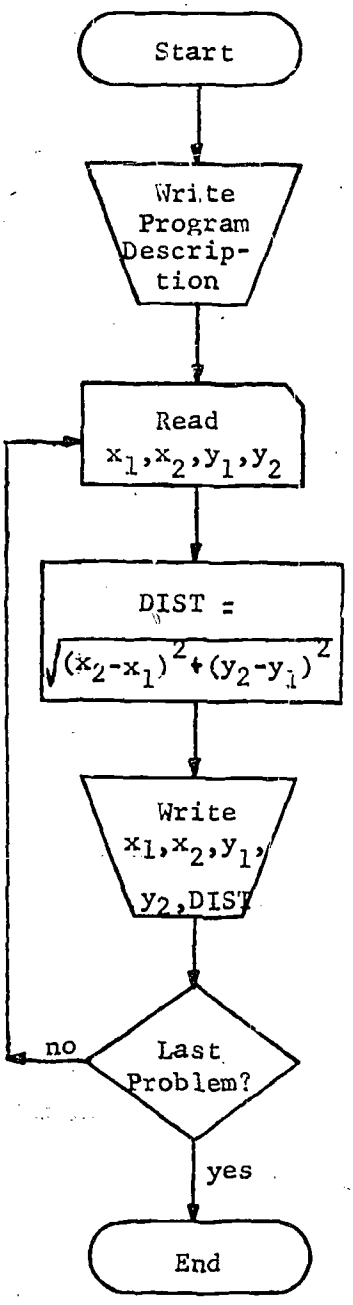
Algebra Section I

TWO-POINT

This program will determine the distance between any two points provided the rectangular coordinates are known.

VARIABLES

X1 - the X coordinate of the first point
Y1 - the Y coordinate of the first point
X2 - the X coordinate of the second point
Y2 - the coordinate of the second point



IBM FORTRAN Coding Form

TWO-POINT

FORTRAN STATEMENT

```
      WRITE(3,1)
    1 FORMAT(2X,'THIS PROGRAM WILL FIND THE DISTANCE BETWEEN TWO POINTS
     1IF THE RECTANGULAR COORDINATES ARE KNOWN',///)
   10 READ(2,2)X1,X2,Y1,Y2
    2 FORMAT(4F7.1)
      DIST=SQRT((X2-X1)**2+(Y2-Y1)**2)
      WRITE(3,4)X1,Y1,X2,Y2,DIST
    4 FORMAT(2X,'X1=',F6.1,5X,'Y1=',F6.1,10X,'X2=',F6.1,5X,'Y2=',F6.1,10
     1X,'DISTANCE=',F8.2)
      GO TO 10
   11 CALL EXIT
      END
```

THE SQUARE ROOT OF COMPLEX NUMBERS

This program determines the square root of complex numbers by the algebraic process of equating $\sqrt{X+Yi}$ to $A+Bi$ squaring both sides and equating the real parts and the imaginary parts. The resulting system is solved by the simultaneous method for A and B. A considerable amount of the solution must be obtained by algebraic methods before programming is begun. The algebraic solution follows:

$$\sqrt{X+Yi} = A+Bi$$

$$X+Yi = A^2+2ABi-B^2$$

$$(1)\quad X = A^2-B^2$$

$$Yi = 2ABi$$

$$(2)\quad Y = 2AB$$

$$\boxed{A = \frac{Y}{2B}}$$

$$(1)\quad X = \left(\frac{Y}{2B}\right)^2 - B^2$$

$$X = \frac{Y^2}{4B^2} - B^2$$

$$4B^2X = Y^2 - 4B^4$$

$$4B^2X + 4B^4 = +Y^2$$

$$4B^4 + 4B^2X = +Y^2$$

$$B^4 + B^2X = \frac{+Y^2}{4}$$

$$B^4 + B^2X + \frac{X^2}{4} = \frac{Y^2}{4} + \frac{X^2}{4}$$

$$\left(B^2 + \frac{X}{2}\right)^2 = \frac{Y^2+X^2}{4}$$

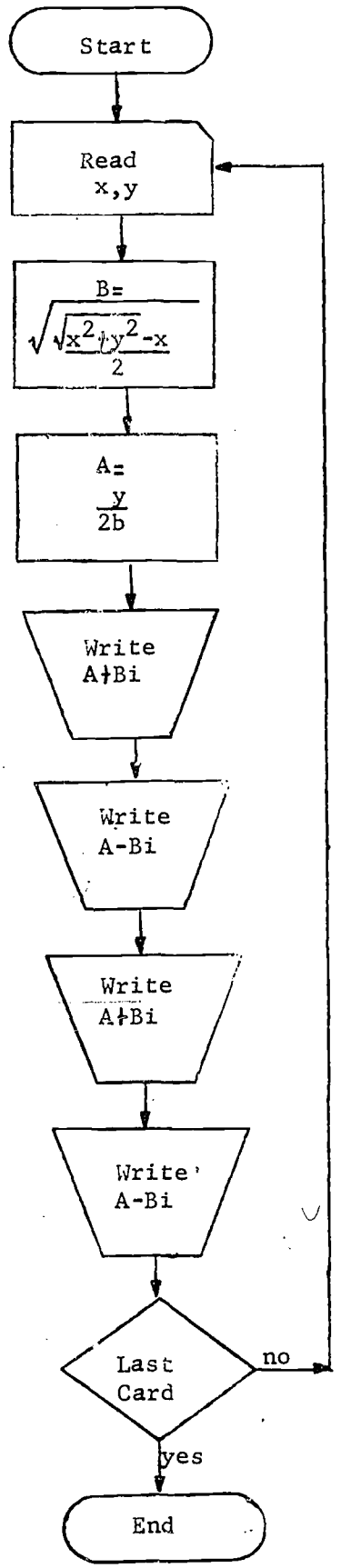
$$B^2 + \frac{X}{2} = \sqrt{\frac{Y^2+X^2}{2}}$$

$$B^2 = \sqrt{\frac{Y^2+X^2}{2}} - \frac{X}{2}$$

$$\boxed{B = \sqrt{\frac{\sqrt{Y^2+X^2} - X}{2}}}$$

VARIABLES

X - the real part of the given number
Y - the coefficient of the imaginary part of the given number
A - the real part of the solution
B - the coefficient of the imaginary part of the solution



IBM FORTRAN Coding Form

FORTRAN STATEMENT

```
C     THIS PROGRAM WILL FIND THE SQUARE ROOT OF COMPLEX NUMBERS
   50 READ(2,1)X,Y
    1 FORMAT(2F7.2)
      B=SQRT((-X+SQRT(X**2+Y**2))/2.)
      A=Y/(2.*B)
      WRITE(3,2)A,B
    2 FORMAT(10X,'+',F7.2,'+',F7.2,'I')
      WRITE(3,3)A,B
    3 FORMAT(10X,'+',F7.2,'-',F7.2,'I')
      WRITE(3,4)A,B
    4 FORMAT(10X,'-',F7.2,'-',F7.2,'I')
      WRITE(3,5)A,B
    5 FORMAT(10X,'-',F7.2,'+',F7.2,'I')
      GO TO 50
  234 CALL EXIT
      END
```

DIVISION OF COMPLEX NUMBERS

This program determines the quotient of two complex numbers.

Multiplying the numerator and denominator by the conjugate of the denominator must be accomplished prior to writing the program.

$$\frac{a+bi}{c+di} = r+ei$$

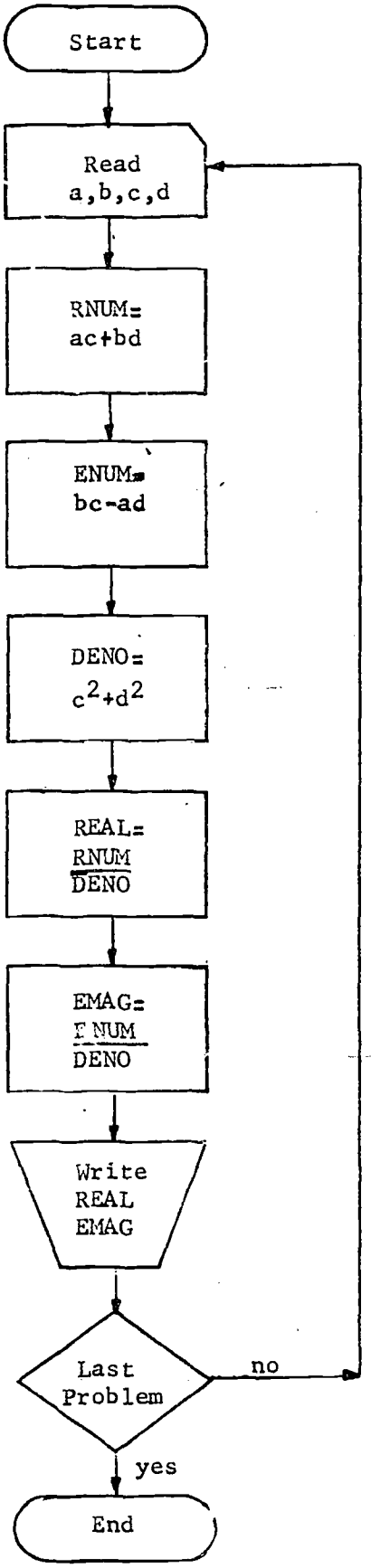
$$\frac{a+bi}{c+di} \times \frac{c-di}{c-di} = \frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$$

$$r = \frac{ac+bd}{c^2+d^2}$$

$$e = \frac{bc-ad}{c^2+d^2}\, i$$

VARIABLES

A - real part of numerator
C - real part of denominator
B - coefficient of imaginary part of numerator
D - coefficient of imaginary part of denominator
REAL - real part of quotient
EMAG - coefficient of imaginary part of quotient
DENO - denominator of quotient



IBM FORTRAN Coding Form

```
C PROGRAM TO DEMONSTRATE DIVISION OF COMPLEX NUMBERS
   20 READ(2,1)A,B,C,D
    1 FORMAT (4F7.2)
      RNUM= A*C+B*D
      ENUM= B*C-A*D
      DENO= C**2 + D**2
      REAL= RNUM/DENO
      EMAG= ENUM/DENO
      WRITE(3,2)REAL,EMAG
    2 FORMAT (2X,'THE COMPLEX QUOTIENT IS ',F9.2,'+ ',F9.2,'I')
      GO TO 20
   50 CALL EXIT
      END
```

ITERATIVE DEMONSTRATION

This program is a most powerful tool of the programmer and demonstrates clearly the "replacement" value concept of computer programming. The technique can be found in most manuals of programming.

Given an equation such as:

$$3x - 4 = \sqrt{2x}$$

Set each side equal to y:

$$y = 3x - 4$$

$$y = \sqrt{2x}$$

Sketch a graph with both equations on the same set of axes. The intersection of these graphs is then considered to be an approximation of the solution set of the original equation.

Partially solving the original equation by normal algebraic methods yields:

$$x \text{ (XNEW)} = \frac{\sqrt{2x} + 4}{3}$$

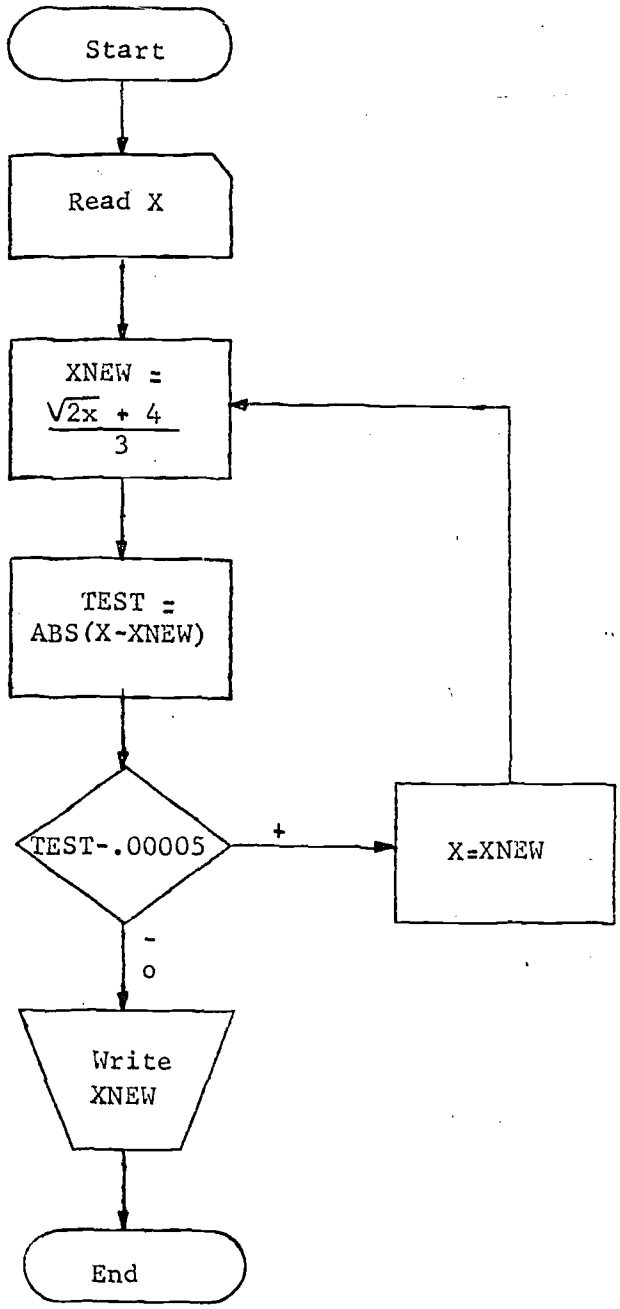
The approximation which was found on the graph is used to compute a new x. The two values of x are then compared. If the difference between the two is more than a predetermined TEST value, the new x is used in the expression to evaluate another XNEW and the process continues until a value equal to or smaller than the TEST constant is reached.

The value of this program can be demonstrated by changing the equation card only, and the input x value to solve each of the following:

$X = .2E^{.5x}$

$X = \cos(X)$

$\sin(x) = \log_{10}(x)$



IBM FORTRAN Coding Form

ITERATIVE DEMONSTRATION

FORTRAN STATEMENT

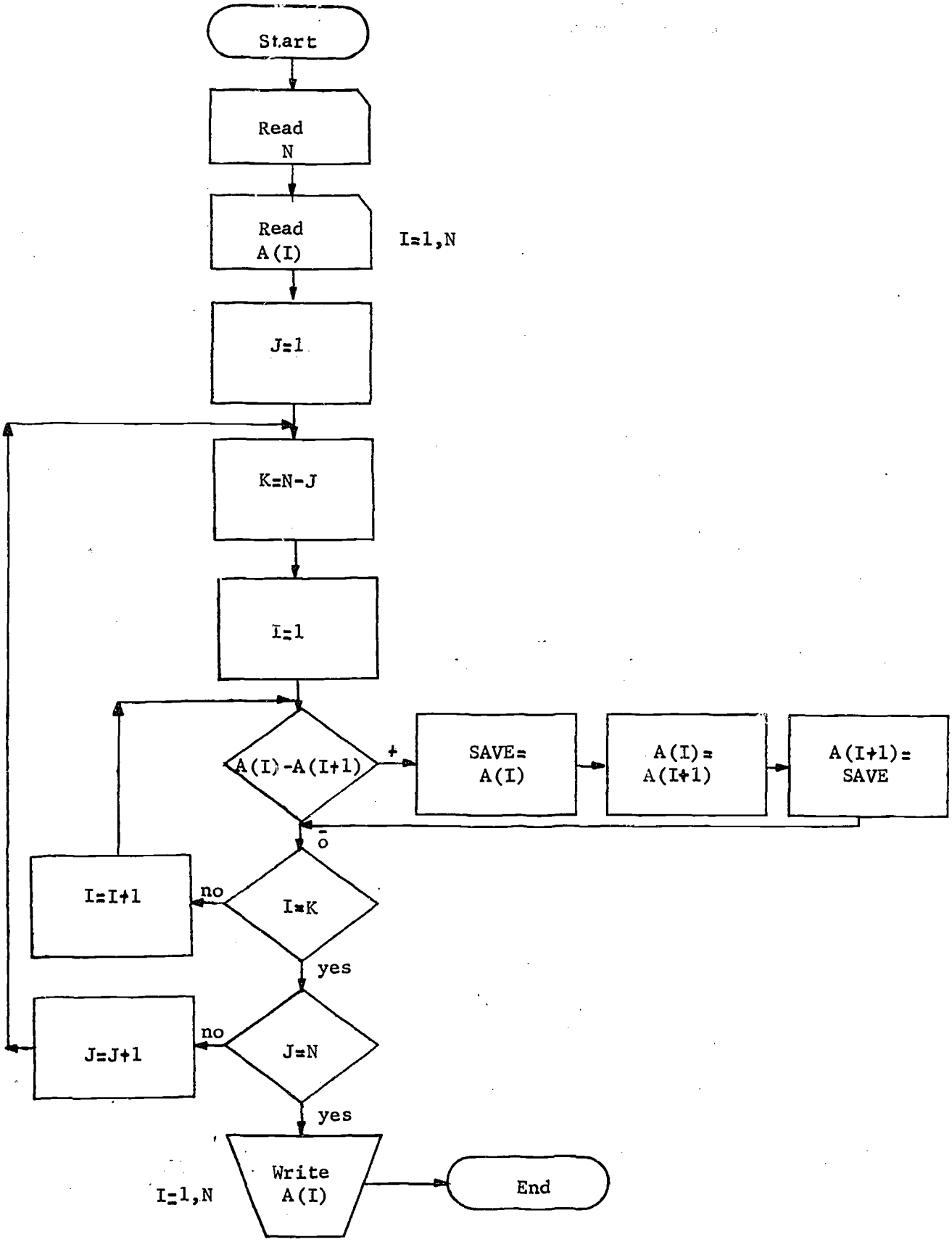
```
      READ (2,1) X
    1 FORMAT (F10.7)
   50 XNEW=(SQRT(2.*X)+4.)/3.
      TEST=ABS(X-XNEW)
      IF(TEST-.00005)25,25,35
   35 X=XNEW
      GO TO 50
   25 WRITE (3,2) XNEW
    2 FORMAT (1X,'A VALUE OF X WHICH SATISFIES THE EQUATION IS',F10.7)
      CALL EXIT
      END
```

SORTLIST

This program will sort a list of numbers into ascending order. It accomplishes this by comparing adjacent numbers and rearranging them if they are out of order. This is an important technique in programming.

VARIABLES

N - the number of numbers in the list
A - the actual numbers comprising the list
SAVE - a variable needed to keep a number when its order is being changed. It would otherwise be destroyed when another number is moved into its place.


Start
Read N
Read A(I)
I=1,N
J=1
K=N-J
I=1
A(I)-A(I+1)
+
SAVE= A(I)
A(I)= A(I+1)
A(I+1)= SAVE
0
I=I+1
no
I=K
yes
J=J+1
no
J=N
yes
I=1,N
Write A(I)
End

IBM FORTRAN Coding Form

SORTLIST

FORTRAN STATEMENT

```
C       THIS PROGRAM WILL SORT A LIST OF NUMBERS INTO ASCENDING ORDER
        DIMENSION A(1000)
        READ(2,1)N
      1 FORMAT(I4)
        READ(2,2)(A(I),I=1,N)
      2 FORMAT(11F6.3)
        DO 50 J=1,N
        K=N-J
        DO 50 I=1,K
        IF(A(I)-A(I+1))50,50,30
     30 SAVE=A(I)
        A(I)=A(I+1)
        A(I+1)=SAVE
     50 CONTINUE
     10 WRITE(3,3)(A(I),I=1,N)
      3 FORMAT(10F10.2)
        CALL EXIT
        END
```

DISTANCE FROM A POINT TO A LINE

Because this program gives the appearance of being relatively short and simple, it has the effect of impressing students who follow its development.

It demonstrates vividly the advantages of simplifying all algebraic equations and combining them where possible so that the complete program consists of a minimum number of statements.

The system —

$$A_1x+B_1y+C_1=0$$

$$A_2x+B_2y+C_2=0$$

was used and solved for x and y. The relationships used in the program were derived from these solutions.

The steps include:

1. Determine slope of given line. Slope $_1 = \frac{A_1}{B_1}$
2. Using this slope determine the equation of the line passing through the point and perpendicular to the given line. (The slope of this line is the negative multiplicative inverse of the slope of the given line.)

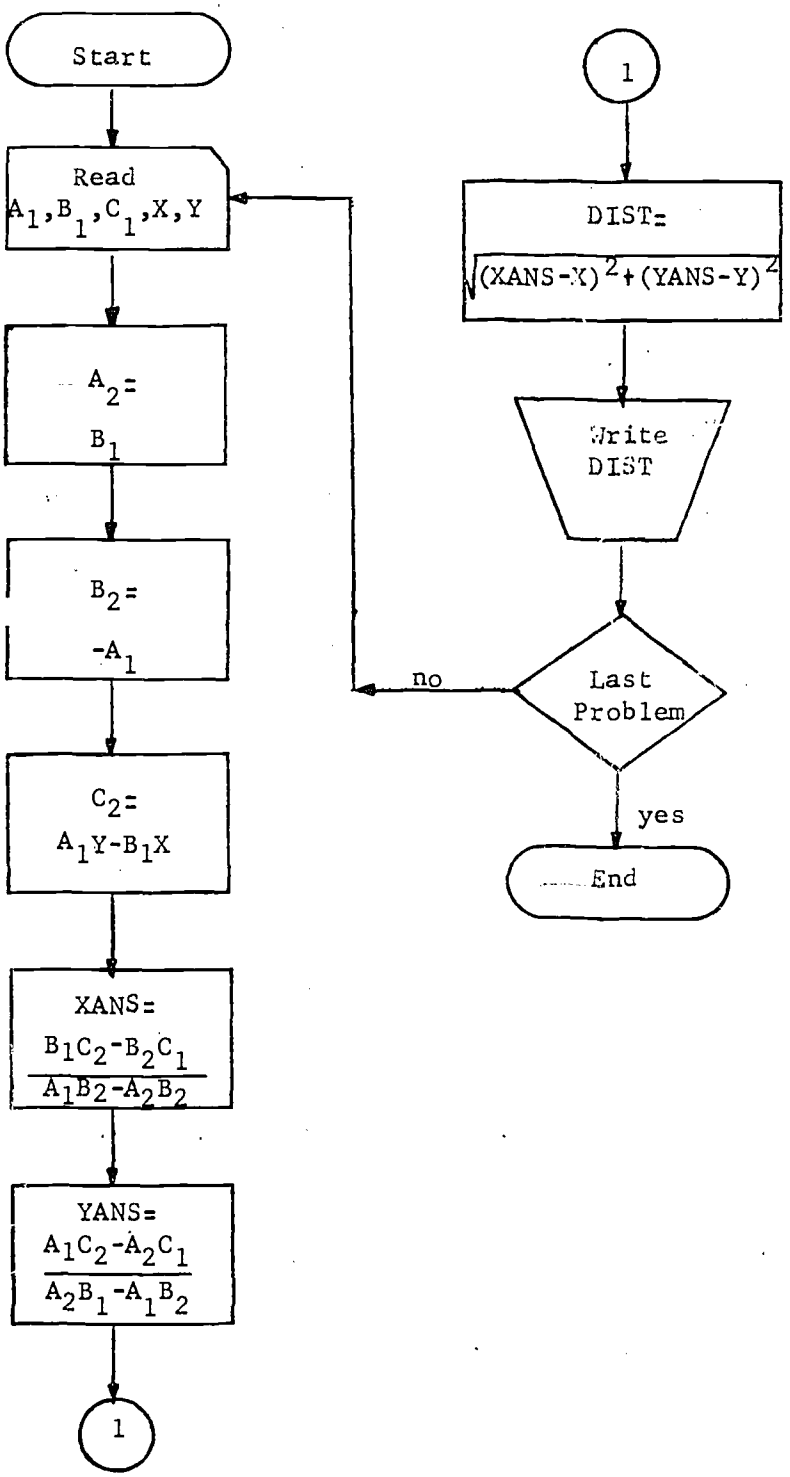
   Slope $_2 = \frac{B_1}{-A_1}$
3. Determine the point of intersection of the two lines. (Solve the system for x and y.)
4. Using this determined point of intersection and the given point, find the distance between them.

   $$d = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

VARIABLES

A1 - the coefficient of X in equation 1

A2 - the coefficient of X in equation 2
B1 - the coefficient of Y in equation 1
B2 - the coefficient of Y in equation 2
C1 - the constant in equation 1
C2 - the constant in equation 2
XANS - the x coordinate of the intersection of the two lines
YANS - the y coordinate of the intersection of the two lines
DIST - the distance from the point to the line.



IBM FORTRAN Coding Form

FORTRAN STATEMENT

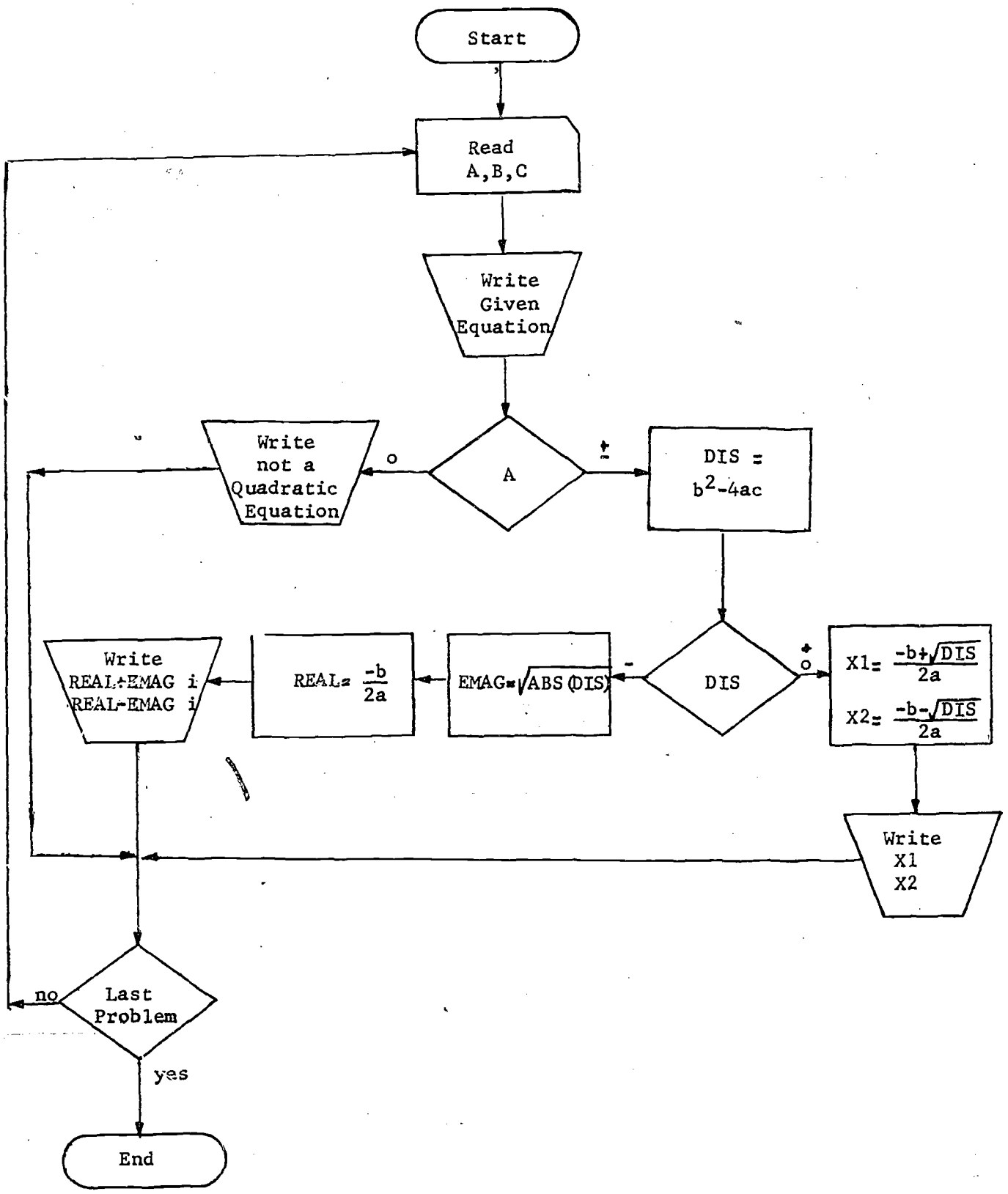
```
C   THIS PROGRAM DETERMINES THE DISTANCE FROM A POINT TO A LINE
6   READ(2,5)A1,B1,C1,X,Y
5   FORMAT(5F7.2)
    A2=B1
    B2=-A1
    C2=A1*Y-B1*X
    XANS=(B1*C2-B2*C1)/(A1*B2-A2*B1)
    YANS=(A1*C2-A2*C1)/(A2*B1-A1*B2)
    DIST=SQRT((XANS-X)**2+(YANS-Y)**2)
    WRITE(3,8)DIST
8   FORMAT(2X,'DISTANCE =',F7.2)
    GO TO 6
13  CALL EXIT
    END
```

SOLUTION OF QUADRATIC EQUATIONS

This program will solve any quadratic equation, including those with imaginary roots, provided the coefficients A, B, and C are properly entered.

VARIABLES

A - the coefficient of X squared
B - the coefficient of X
C - constant
DIS - the discriminant, b^2-4ac
REAL - the real part of the complex solution
EMAG - the coefficient of the imaginary part of the complex solution


Start
Read
A,B,C
Write
Given
Equation
A
0
Write
not a
Quadratic
Equation
±
DIS =
b2-4ac
DIS
-
EMAG=√ABS(DIS)
REAL= -b/2a
Write
REAL+EMAG i
REAL-EMAG i
+
0
X1= (-b+√DIS)/2a
X2= (-b-√DIS)/2a
Write
X1
X2
Last
Problem
no
yes
End

IBM FORTRAN Coding Form

SOLUTION OF A QUADRATIC EQUATION

FORTRAN STATEMENT

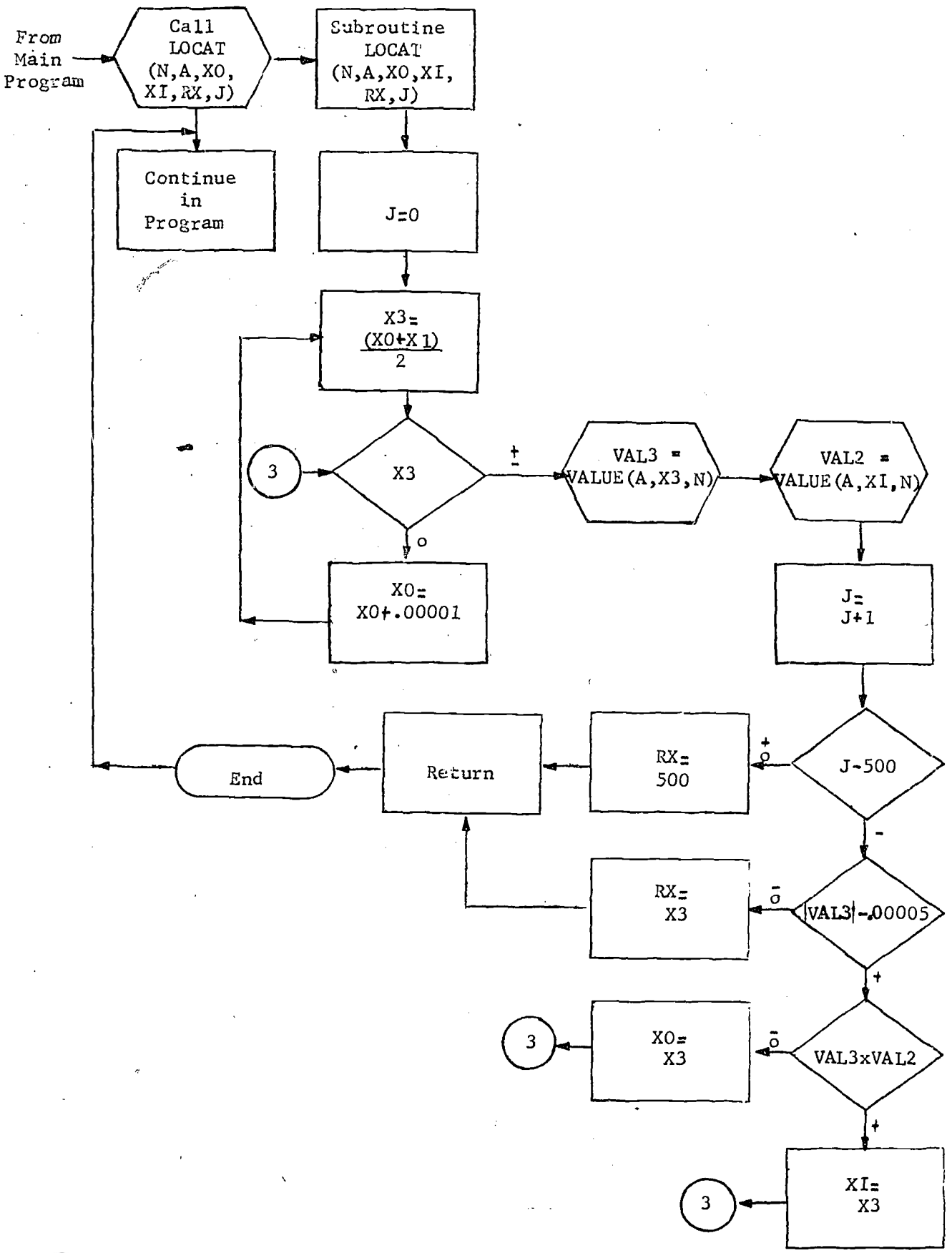
```
C THIS PROGRAM WILL SOLVE ALL QUADRATIC EQUATIONS
   10 READ (2,1)A,B,C
    1 FORMAT(3F10.2)
      WRITE(3,12)A,B,C
   12 FORMAT(//,F10.2,'X**2+',F10.2,'X+',F10.2,'=0.0')
      IF(A)6,5,6
    5 WRITE(3,2)
    2 FORMAT(2X,'NOT A QUADRATIC EQUATION.')
      GO TO 10
    6 DIS=B**2-4.0*A*C
      IF(DIS)7,8,8
    8 X1=(-B+SQRT(DIS))/(2.0*A)
      X2=(-B-SQRT(DIS))/(2.0*A)
      WRITE(3,3)X1,X2
    3 FORMAT(2X,'X1=',F10.2,7X,'X2=',F10.2)
      GO TO 10
    7 REAL=(-B)/(2.0*A)
      EMAG=SQRT(ABS(DIS))
      WRITE(3,4)REAL,EMAG,REAL,EMAG
    4 FORMAT(2X,'X1=',F10.2,'+',F10.2,'I',/,'X2=',F10.2,'-',F10.2,'I')
      GO TO 10
  100 CALL EXIT
      END
```

LOCATION OF REAL ROOTS (SUBROUTINE)

The subroutine finds real roots between 2 points (XO,f(XO)) and (XI,f(XI)). This is accomplished by bisecting the interval XI,XO, and evaluating at the new X3. If X3 is within .00005 unit, it is declared to be the real root. If not, a product testing follows. A positive product for XI*X3 indicates that the real root lies between X0 and X3. Therefore the new XI is X3, and the bisection is repeated. If a negative product for XI*X3 is determined, then the real root lies between these 2 values and the new XO is X3.

VARIABLES

N - number of terms in the polynomial
A(N) - coefficients of polynomial
XO - upper limit of interval containing real root
XI - lower limit of interval containing real root
X3 - an evaluated intermediate term between XO and XI
J - a counter index
Value (A,XI,N) - another subroutine which evaluates the polynomial at XI


From Main Program
Call LOCAT (N,A,XO, XI,RX,J)
Subroutine LOCAT (N,A,XO,XI, RX,J)
Continue in Program
J=0
X3= (XO+X1) 2
3
X3
±
VAL3 = VALUE(A,X3,N)
VAL2 = VALUE(A,XI,N)
o
XO= XO+.00001
J= J+1
End
Return
RX= 500
+ o
J-500
-
RX= X3
o
|VAL3|-.00005
+
3
XO= X3
o
VAL3xVAL2
+
3
XI= X3

IBM FORTRAN Coding Form

PROGRAM SUBROUTINE LOCAT DATE

STATEMENT NUMBER	FORTRAN STATEMENT
	SUBROUTINE LOCAT(N,A,XO,XI,RX,J)
	J=0
30	X3=(XO+XI)/2.
	IF(X3)25,20,25
20	XO=XO+.0001
	GO TO 30
25	VAL3=VALUE(A,X3,N)
	VAL2=VALUE(A,X1,N)
	J=J+1
	IF(J-500)55,50,50
55	IF(ABS(VAL3)-.00005)35,35,15
15	IF(VAL3*VAL2)40,40,45
40	XO=X3
	GO TO 30
45	XI=X3
	GO TO 30
35	RX=X3
	GO TO 60
50	RX=500
60	RETURN
	END

PROGRAM GRAPHING

Although a curve plotter is available for the 1130 computer, this program is included as a demonstration for plotting. Double subscripted variables (IP) are used to identify points of a 41 x 41 array. This matrix is first filled with blanks, then positions which will represent the x and y axes are filled with dots. Integral pairs which satisfy the equation are shown on the graph as asterisks. For this program, coefficients of the polynominal (quadratic) $ax^2+bx+c=0$ are read into the program. This equation is evaluated for integral values of x from -20 to +20. Results of this evaluation are compared with each of the y coordinates between -20 and +20 which might be paired with the x's. If the result of the comparison is 0, the pair represents a point on the graph of the equation and an asterisk is placed in storage for this point. Finally, using a self-indexed WRITE statement, all values now occupying the matrix are printed. By changing the arithmetic statement which evaluates the equation, and changing the statement which reads the coefficients accordingly, other equations could be plotted.

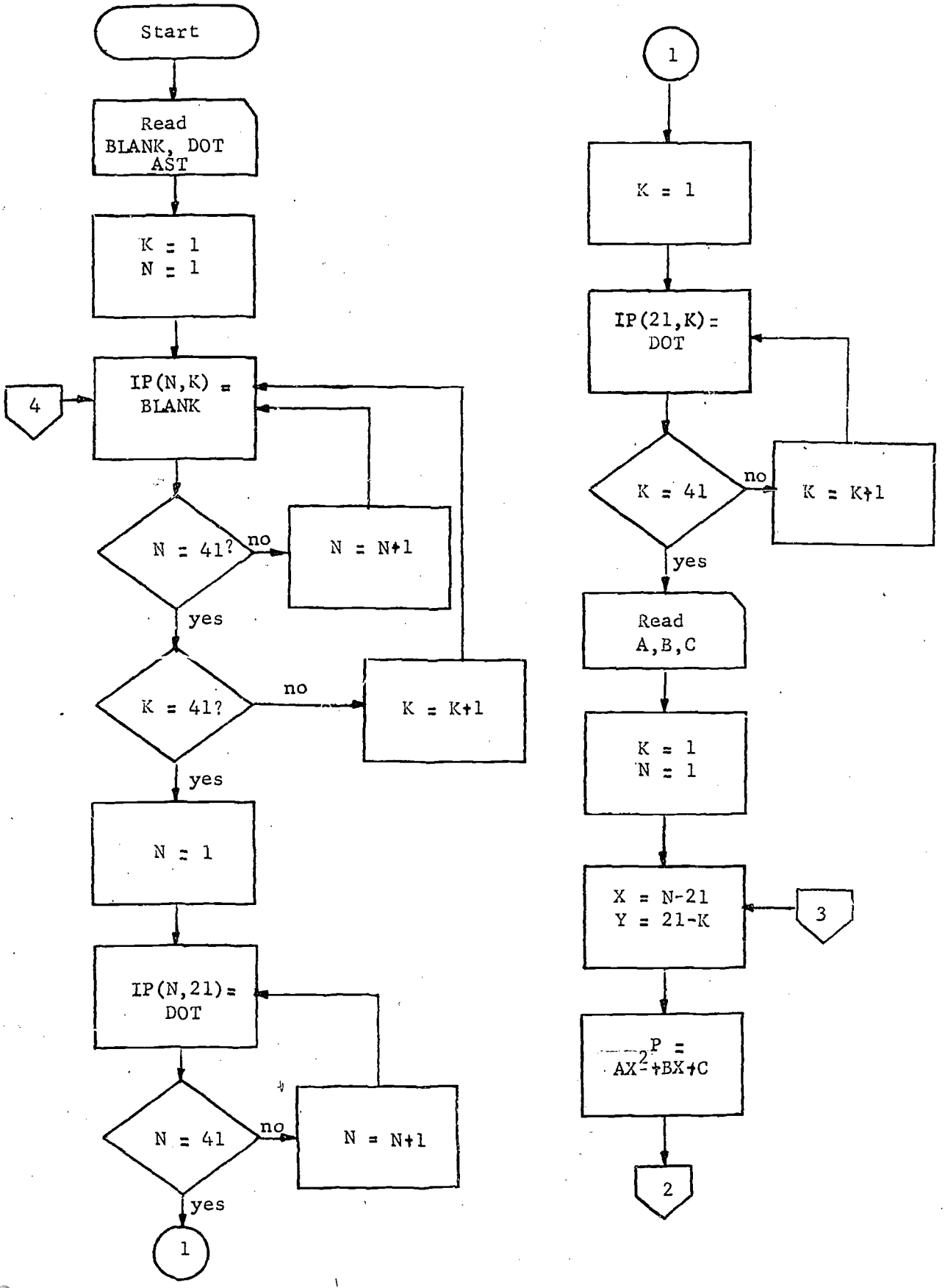
The input card contains a blank in column 1, a period in column 2, and an asterisk in column 3.

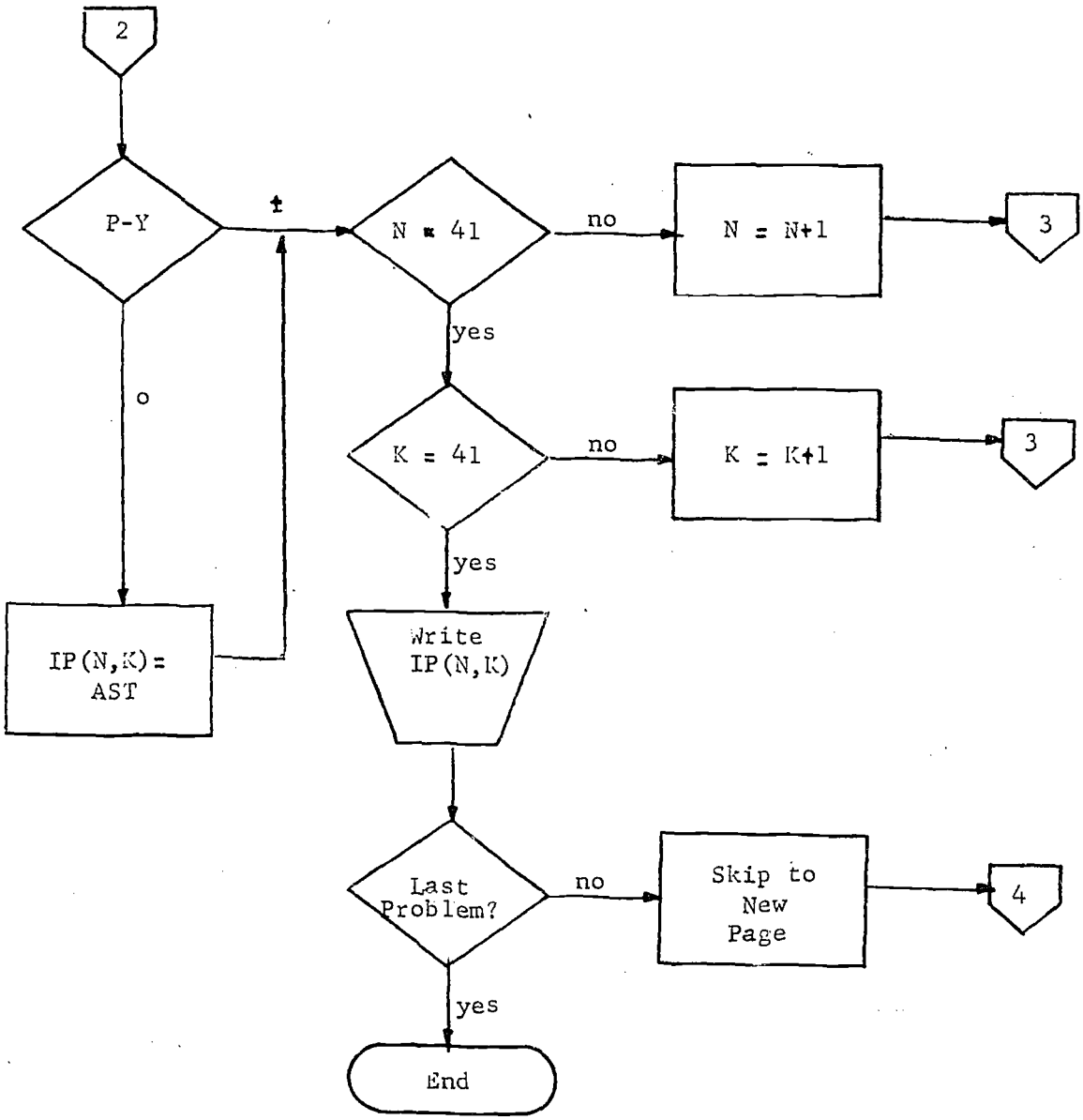
Separate graphs, each on a fresh sheet of paper, will be produced if several cards are used for different values of a, b, and c.

Space for the array IP is reserved by the INTEGER statement rather than the usual DIMENSION statement.

VARIABLES

A,B,C - coefficients of a quadratic equation
X,Y - coordinates of points on the graph
BLANK - an alphabetic blank
DOT - a period
AST - an asterisk
IP - points of a grid
P - value of the equation


Start
Read BLANK, DOT AST
K = 1
N = 1
IP(N,K) = BLANK
4
N = 41?
no
N = N+1
yes
K = 41?
no
K = K+1
yes
N = 1
IP(N,21) = DOT
N = 41
no
N = N+1
yes
1
1
K = 1
IP(21,K) = DOT
K = 41
no
K = K+1
yes
Read A,B,C
K = 1
N = 1
X = N-21
Y = 21-K
3
P = AX^2+BX+C
2


2
P-Y
t
N ≤ 41
no
N = N+1
3
yes
o
K = 41
no
K = K+1
3
yes
IP(N,K) = AST
Write IP(N,K)
Last Problem?
no
Skip to New Page
4
yes
End

IBM

FORTRAN Coding Form

```
* ONE WORD INTEGERS
C GRAPH OF A QUADRATIC EQUATION
      INTEGER IP(41,41),BLANK,DOT,AST
   80 READ (2,3) BLANK,DOT,AST
    3 FORMAT (3A1)
  800 DO 4 K=1,41
      DO 4 N=1,41
    4 IP(N,K) = BLANK
      DO 5 N=1,41
    5 IP(N,21) = DOT
      DO 6 K=1,41
    6 IP(21,K) = DOT
   50 READ (2,1) A,B,C
    1 FORMAT (3F7.3)
      DO 10 K=1,41
      DO 10 N=1,41
      X = N-21
      Y = 21-K
      P = A*X**2+B*X+C
      IF (P-Y) 10,20,10
   20 IP(N,K) = AST
   10 CONTINUE
      WRITE (3,2) ((IP(N,K),K=1,41),N=1,41)
    2 FORMAT (2X,41A1)
      WRITE (3,7) A,B,C
    7 FORMAT (2X,'GRAPH OF',2X,'Y=',F7.3,'X**2+',F7.3,'X+',F7.3)
      WRITE (3,8)
    8 FORMAT (1H1)
      GO TO 800
   11 CALL EXIT
      END
```

TEST

A useful demonstration for any type of class, this program is presented because of its versatility. It is not actually designed for serious testing, but rather as a demonstration that is easily understood, but is of a serious nature. It was first written for mathematics classes, but later was used in social studies, english, and music. Substitution of appropriate questions for the discipline desired, in the correct order is all that is needed to adapt this test to another subject matter area.

```
PAGE.  1
// JOB
LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0002         0002         0000

// FOR
*IOCS(CARD,TYPEWRITER,KEYBOARD,1132 PRINTER,DISK,PAPER TAPE)
*LIST SOURCE PROGRAM
C       PROGRAM TEST
  777   WRITE(1,1)
    1   FORMAT(' ENTER YOUR NAME')
        READ (6,40) NAME
   40   FORMAT (A30)
        WRITE (1,41)
   41   FORMAT (' ENTER YOUR PERIOD NUMBER')
        READ(5,42)NO
   42   FORMAT (I2)
        WRITE (1,2)
    2   FORMAT('ALGEBRA TEST, NOTE-- **2=SQUARE',//,
     1' THIS SECTION OF THIS TEST CONSISTS OF A SET OF STATEMENTS '
     2'TO BE CLASSIFIED AS TRUE OR FALSE')
        WRITE (1,25)
   25   FORMAT (' IF YOUR ANSWER IS TRUE, ENTER 1., IF FALSE ENTER 2.')
        WRITE(1,3)
    3   FORMAT(' SAMPLE QUESTION',//,
     1' GIVEN THE FUNCTION F(X)=3(X-1)**2+2,'/,'THE GRAPH HAS A MINIMUM
     2POINT.'//,'ENTER YOUR ANSWER'/,)
        READ (6,30) ANS
   30   FORMAT (F4.2)
        IF(ANS-1.)61,60,61
   60   SCORE=2.
        WRITE(1,4)SCORE
    4   FORMAT(' YOUR ANSWER IS CORRECT.YOU HAVE SCORED',F7.2//)
        GO TO 16
   61   SCORE=0.
        WRITE(1,5)SCORE
    5   FORMAT(' YOU GOOFED.YOUR SCORE IS',F7.2//)
   16   WRITE (1,6)
    6   FORMAT(' THE REAL TEST BEGINS HERE.',//' THE EQUATION OF THE AXIS
     1OF SYMMETRY IS X=1.',//,' ANSWER 1. OR 2.')
        READ (6,30) ANS
        SCORE=0.
        IF (ANS-1.)63,62,63
   62   SCORE=SCORE+2.
   63   WRITE (1,7)
    7   FORMAT('THE COORDINATES OF THE VERTEX ARE (-1,2)'//)
        READ (6,30) ANS
        IF (ANS-1.)64,65,64
   64   SCORE=SCORE+2.
   65   WRITE (1,8)
    8   FORMAT('THE GRAPH IS A HYPERBOLA'//)
        READ (6,30) ANS
        IF (ANS-1.) 66,67,66
   66   SCORE=SCORE+2.
```

```
   67 WRITE (1,9)
    9 FORMAT('THE MAXIMUM VALUE OF THE FUNCTION IS 2'//)
      READ (6,30) ANS
      IF (ANS-1.)68,69,68
   68 SCORE=SCORE+2.
   69 WRITE (1,10)
   10 FORMAT('THIS GRAPH IS SYMMETRIC WITH RESPECT TO THE Y-AXIS'//)
      READ(6,30)ANS
      IF (ANS-1.)70,71,70
   70 SCORE=SCORE+2.
   71 WRITE (1,11)SCORE
   11 FORMAT(' YOUR SCORE ON THIS TEST IS',F7.2//)
      IF (SCORE-7.)72,73,74
   72 WRITE (1,12)
   12 FORMAT(' YOU NEED TO STUDY THIS AGAIN')
      GO TO 80
   73 WRITE (1,13)
   13 FORMAT (' YOU BARELY MADE IT, STUDY MORE')
      GO TO 80
   74 WRITE (1,14)
   14 FORMAT (' YOU PASSED, KEEP UP THE GOOD WORK')
   80 WRITE (1,81)
   81 FORMAT ('   IF THIS IS THE LAST PERSON TO TAKE THIS TEST ENTER THE
     1NUMBER 88.88,IF NOT TYPE IN 0.')
      READ (6,82)Z
   82 FORMAT (F5.2)
      IF (Z-88.88)777,555,777
  555 CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     10  PROGRAM    764

 END OF COMPILATION

// XEQ
```

PROBLEM SUPPLEMENT

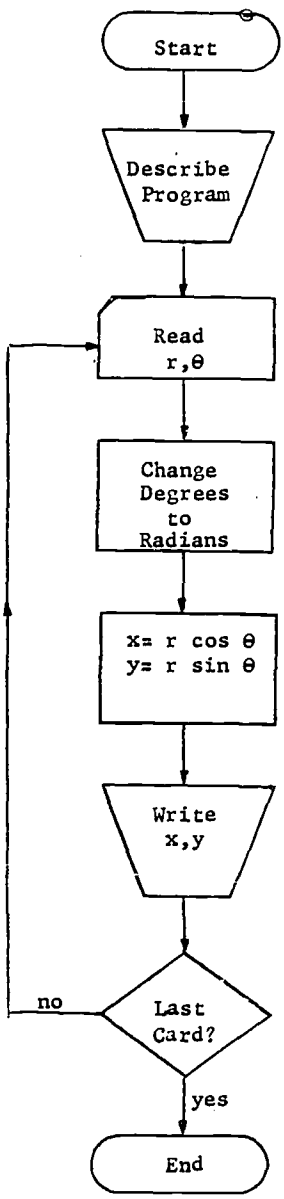
Trigonometry Section II

POTRE

R and THETA must be given and the X and Y coordinates are found. This simple program is easily understood by beginning trigonometry students, and can be programmed with only a minimum of practice in FORTRAN. 9 FORMAT provides for spacing between answers.

VARIABLES

R - polar distance
THETA - direction angle in degrees
RDINS - radians in theta
X - abcissa
Y - ordinate



IBM FORTRAN Coding Form

POTRE

FORTRAN STATEMENT

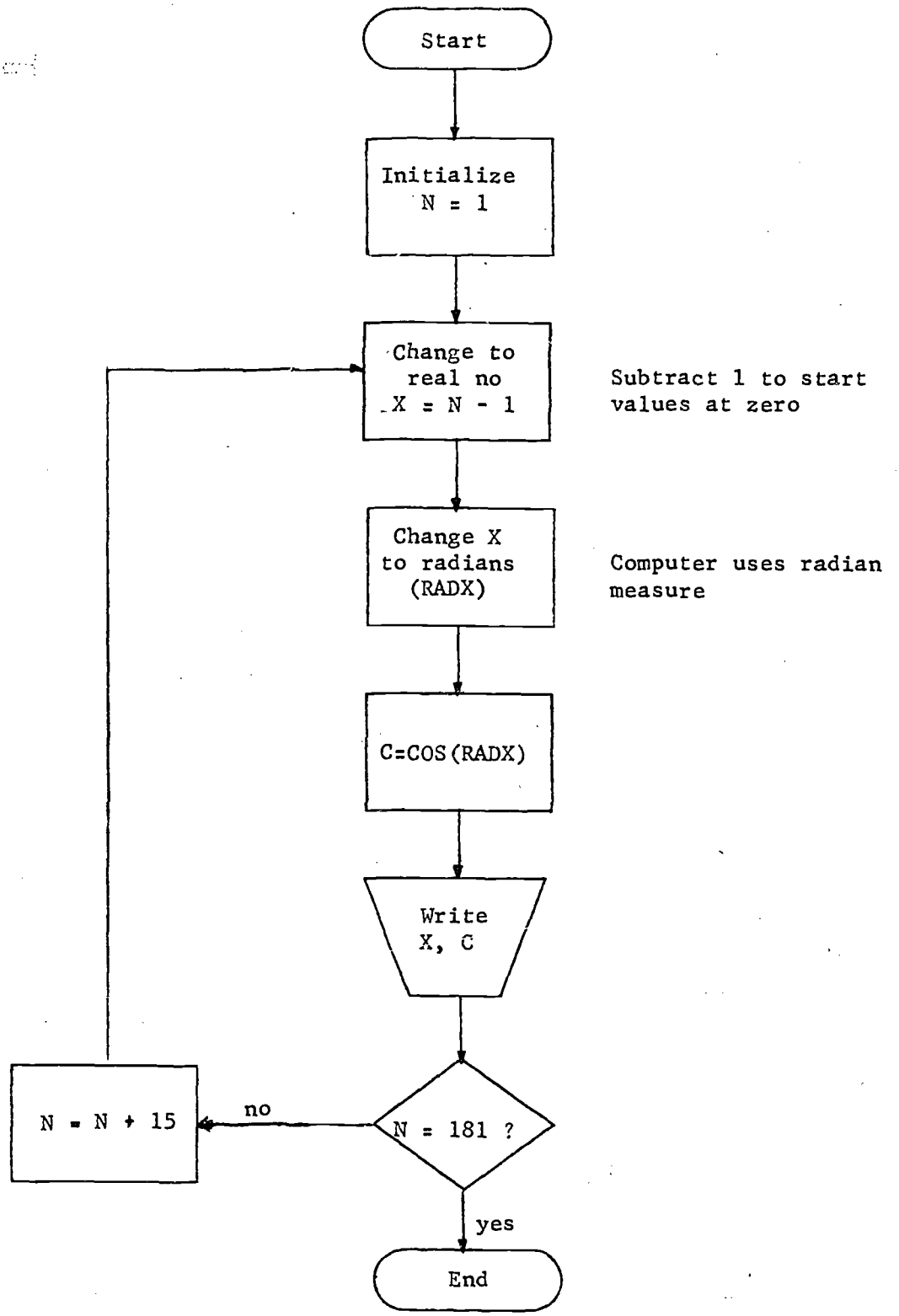
```
C       PROGRAM POTRE
   10   WRITE(3,1)
    1   FORMAT(' THIS PROGRAM WILL CHANGE POLAR COORDINATES TO RECTANGULAR
       1 COORDINATES.')
   20   READ(2,2)R,THETA
    2   FORMAT(F10.4,F5.1)
        RDINS=THETA*3.1416/180.
        X=R*COS(RDINS)
        Y=R*SIN(RDINS)
        WRITE(3,3)X,Y
    3   FORMAT(' THE RECTANGULAR COORDINATES ARE (',F10.4,',',F5.1,')')
        WRITE(3,9)
    9   FORMAT(/////)
        GO TO 20
   30   CALL EXIT
        END
  //    XEQ
```

LISCO

This program illustrates input by a DO-loop. No data cards are required. The library function COS is employed. All the trigonometry required is knowledge of the meaning of the cosine and changing from degrees to radians. A good point to emphasize is the fact that the word COS cannot be used as a variable name, because it is the name of a library function.

VARIABLES

N - number of degrees in angle
X - real number for number of degrees in angle
RANG - radians in angle
C - cosine of the angle


Start
Initialize
N = 1
Change to
real no
X = N - 1
Subtract 1 to start
values at zero
Change X
to radians
(RADX)
Computer uses radian
measure
C=COS(RADX)
Write
X, C
N = 181 ?
no
N = N + 15
yes
End

IBM FORTRAN Coding Form

FORTRAN STATEMENT

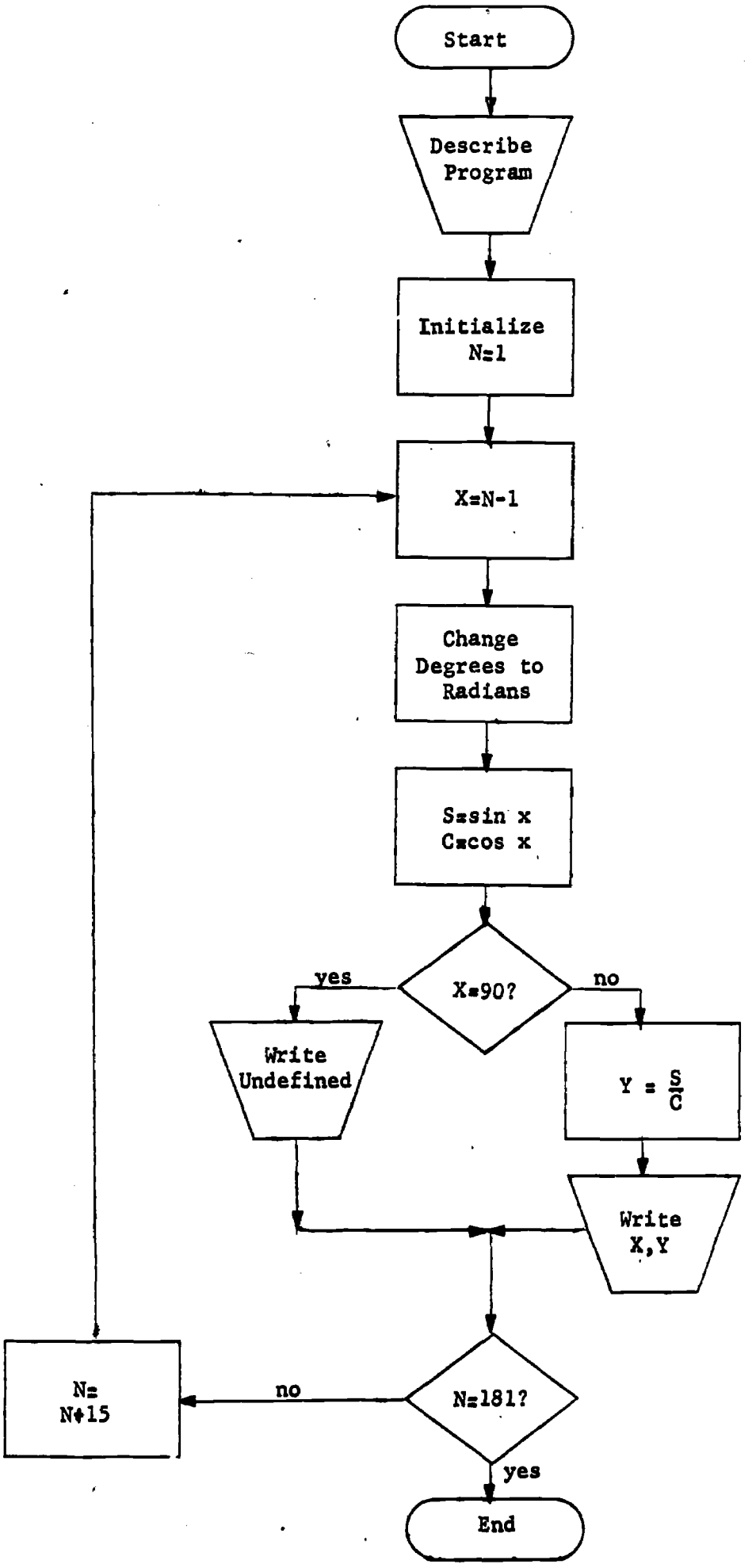
```
C PROGRAM LISCO THIS PROGRAM WILL LIST THE COSINES OF ANGLES FROM 0° TO
C 180 DEGREES AT INTERVALS OF 15 DEGREES.
      Do 30 N=1,181,15
      X=N-1
      RANG=X*3.1416/180.
      C=COS(RANG)
      WRITE (3,1)X,C
    1 FORMAT (2X,'X=',F5.1,10X,'COS(X)=',F10.5)
   30 CONTINUE
      CALL EXIT
      END
```

TANCR

This program is similar to PROGRAM LISCO but since the tangent is undefined at 90°, provision for a message to this effect is necessary. This is an easy program and could be assigned to be written by students after they have been shown PROGRAM LISCO in class.

VARIABLES

X - number of degrees in the angle
RANG - number of radians in the angle
S - sine x
C - cosine x
Y - tangent x


Start
Describe Program
Initialize N=1
X=N-1
Change Degrees to Radians
S=sin x
C=cos x
X=90?
yes
no
Write Undefined
Y = S/C
Write X,Y
N=181?
no
N=N+15
yes
End

IBM

FORTRAN Coding Form

FORTRAN STATEMENT

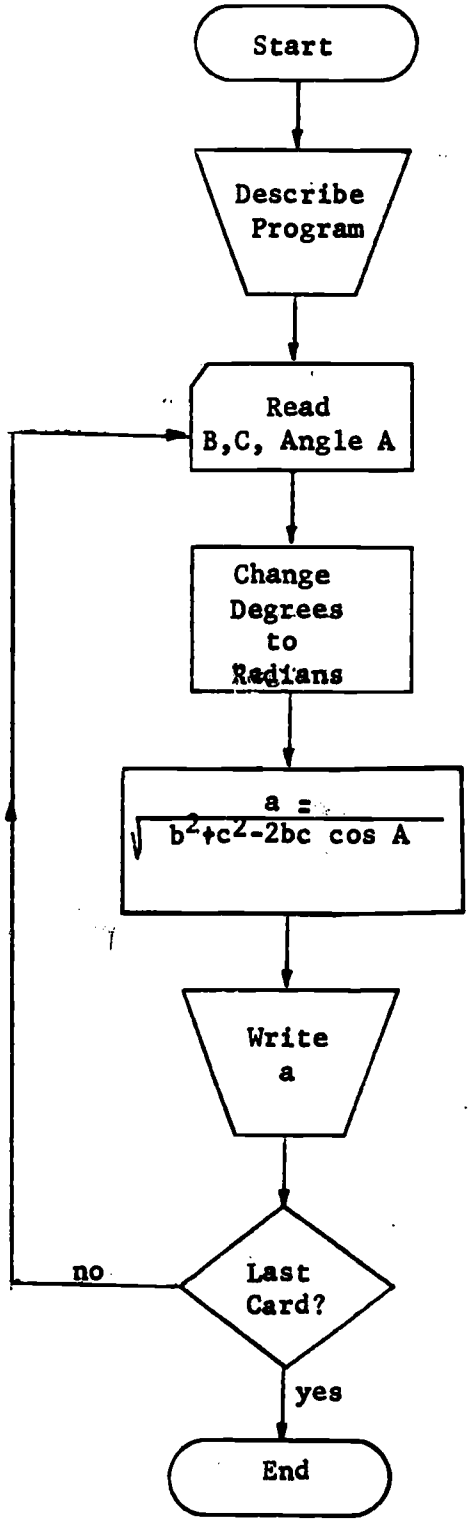
```
C PROGRAM TANCR
      WRITE (3, 1)
    1 FORMAT (2X,'THIS PROGRAM WILL GIVE VALUES FOR X AND Y=TAN(X)')
      DO 30 N = 1, 181, 15
      X = N-1
      RANG = X*3.1416/180.
      S = SIN(RANG)
      C = COS(RANG)
      IF (X-90.0) 4,2,4
    2 WRITE (3,3) X
    3 FORMAT (2X,'X=',F5.1,20X,'Y= UNDEFINED')
      GO TO 30
    4 Y = S/C
      WRITE (3,5) X,Y
    5 FORMAT (2X,'X=',F5.1,20X,'Y=',F10.5)
   30 CONTINUE
      CALL EXIT
      END
```

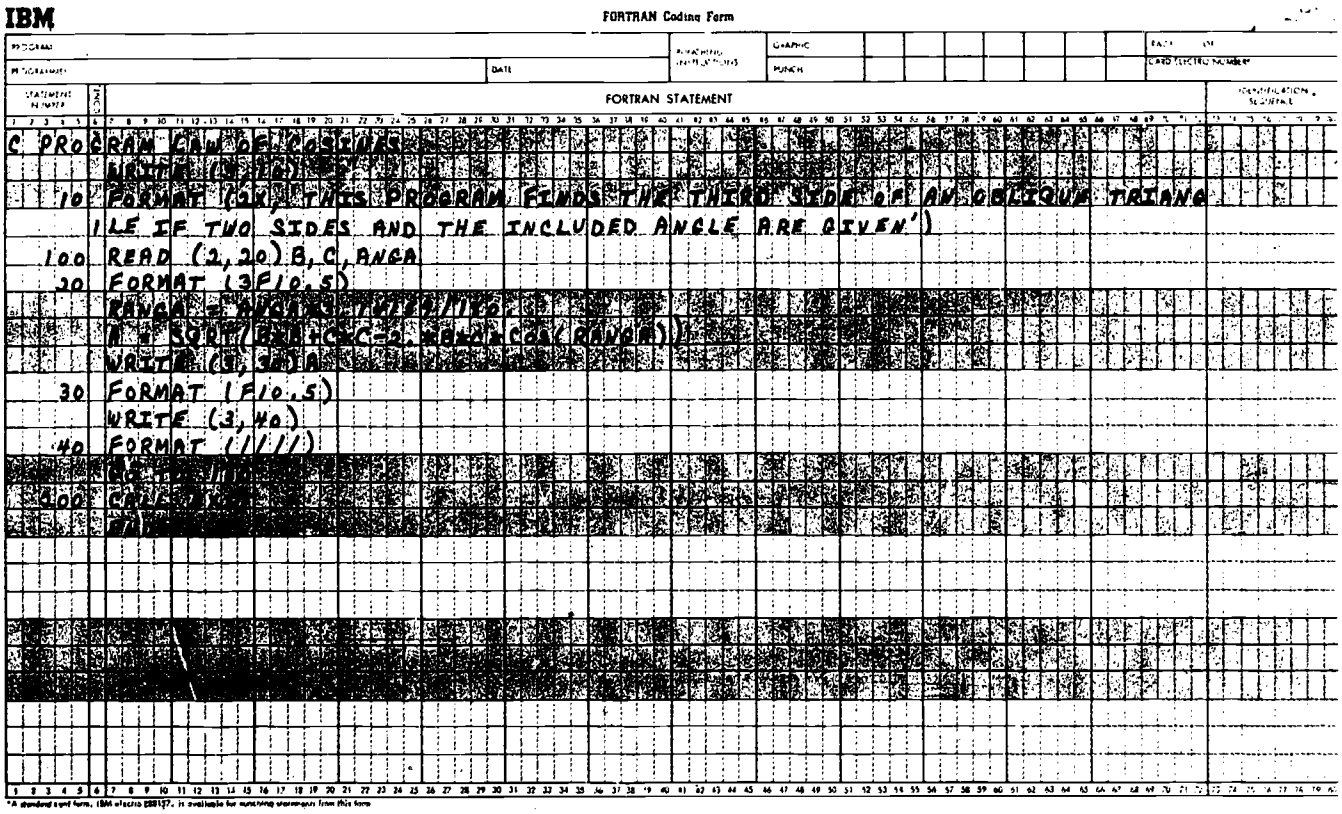
LAW OF COSINES

This is an extremely simple program using simple arithmetic statements. However, the introductory statement in 10 FORMAT requires more spaces than can be used on one card, so that the use of column 6 for continuation of longer statements is illustrated. 40 FORMAT illustrates a method of spacing between problems.

VARIABLES

B, C - given sides
ANGA - given angle in degrees
RANGA - radians in given angle
A - missing side


Start
Describe Program
Read B,C, Angle A
Change Degrees to Radians
a = √(b²+c²-2bc cos A)
Write a
Last Card?
no
yes
End

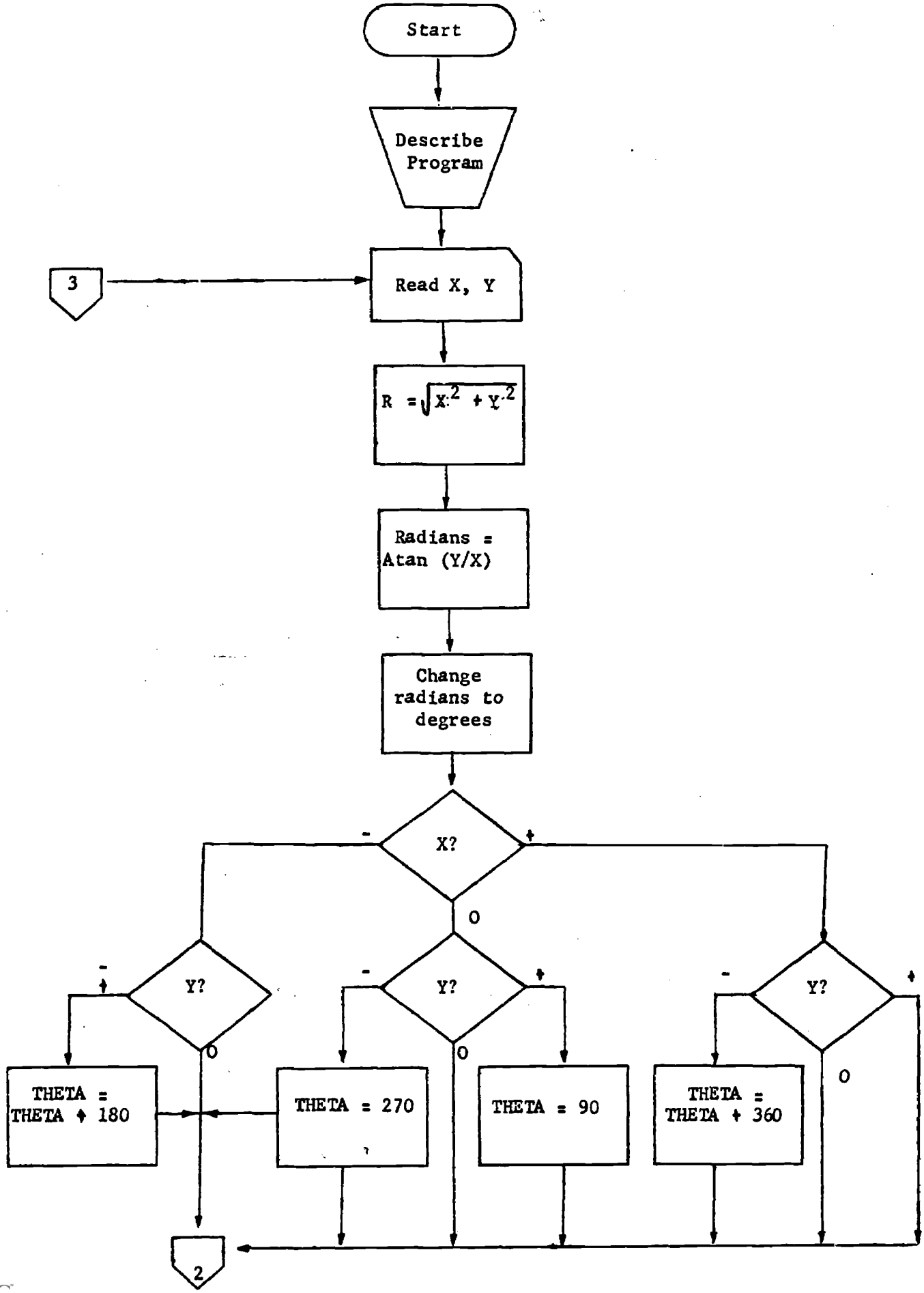


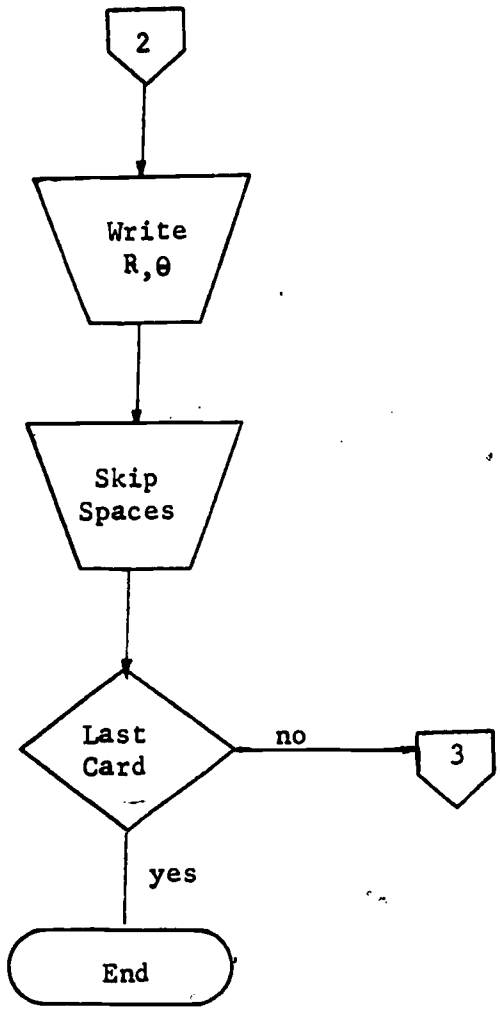
RETOP

The X and Y coordinates of a point are read from cards and the polar distance (R) and direction angle (θ) are computed. Most students are familiar with angles measured in degrees in the early stages of the study of trigonometry. However, the ATAN function produces results in radians and only for the principal values of the inverse tangent ($-\pi$ to $+\pi$). An excellent opportunity for practice is realized by having the program convert the values of the ATAN to the familiar measurements in each of the four quadrants. The signs of the coordinates in each of the 4 quadrants are checked and appropriate quantities added to produce angles between 0° and 360°. Quadrantal angles can be handled in the same sequence.

VARIABLES

X, Y - rectangular coordinates
R - polar distance
THETA - direction angle


Start
Describe Program
3
Read X, Y
R = √X^2 + Y^2
Radians = Atan (Y/X)
Change radians to degrees
X?
-
+
0
Y?
-
+
0
THETA = THETA + 180
THETA = 270
THETA = 90
THETA = THETA + 360
2



IBM FORTRAN Coding Form

RETOP

		FORTRAN STATEMENT
C		PROGRAM RETOP
		WRITE(3,1)
	1	FORMAT(' THIS PROGRAM WILL CHANGE RECTANGULAR COORDINATES TO POLAR
	1	COORDINATES')
	10	READ(2,2)X,Y
	2	FORMAT(2F10.4)
		R=SQRT(X**2+Y**2)
		RDIMS=ATAN(Y/X)
		THETA=180.*RDIMS/3.1416
		IF(X)4,5,6
	4	IF(Y)7,40,7
	5	IF(Y)9,40,90
	6	IF(Y)8,40,40
	7	THETA=THETA+180.
		GO TO 40
	8	THETA=THETA+360.
		GO TO 40
	9	THETA=270.
		GO TO 40
	90	THETA=90.
		GO TO 40
	100	THETA=180.

IBM

RETOP (CONC.)

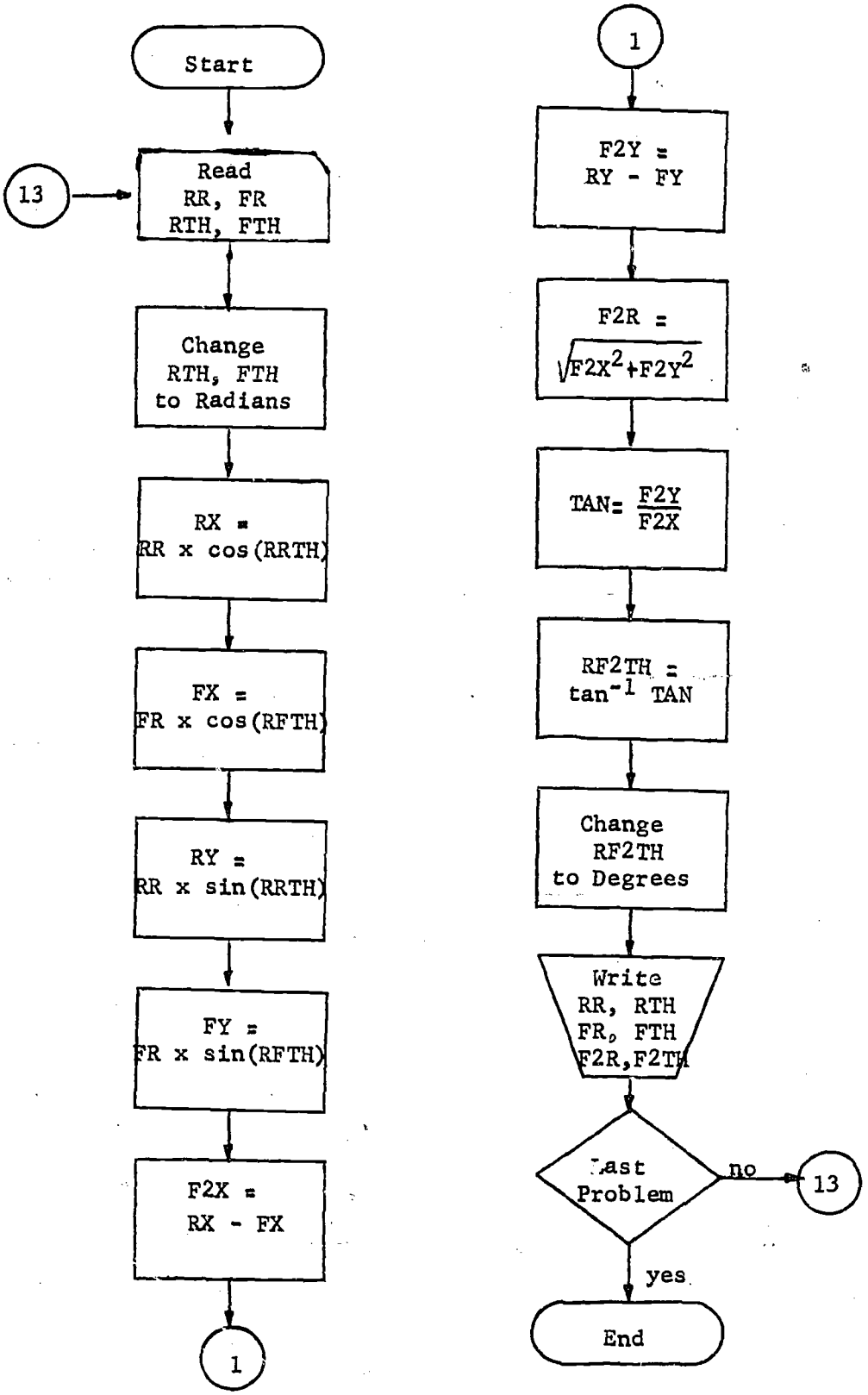
```
      GO TO 40
   40 WRITE(3,3)R,THETA
    3 FORMAT(' THE POLAR COORDINATES ARE (',F10.4,',',F10.4,')')
      WRITE(3,19)
   19 FORMAT(/////)
      GO TO 10
   20 CALL EXIT
      END
//    XEQ
```

PROGRAM MISFOR

This program is suitable for use in the early stages of both trigonometry and programming. It is similar to the program REFOR, but unlike it, in that data must be entered on cards instead of on the console keyboard. It could be useful as an assignment — after the demonstration program REFOR has been used. One ambitious student combined MISFOR and REFOR so that the new program could handle both types of problems. The radius vectors and direction angles are first changed to complex numbers and then subtracted. The missing force is found as a complex number and then changed to polar form.

VARIABLES

RR - the polar distance of the resultant
FR - the polar distance of the known force
RTH - the direction angle of the resultant in degrees
FTH - the direction angle of the known force in degrees
RRTH - RTH in radians
RFTH - FTH in radians
RX - X component of the resultant
RY - Y component of the resultant
FX - X component of the known force
FY - Y component of the known force
F2X - X component of the missing force
F2Y - Y component of the missing force
F2R - the polar distance of the missing force
F2TH - the direction angle of the missing force in degrees
RF2TH - F2TH in radians


Start
13
Read
RR, FR
RTH, FTH
Change
RTH, FTH
to Radians
RX =
RR x cos(RRTH)
FX =
FR x cos(RFTH)
RY =
RR x sin(RRTH)
FY =
FR x sin(RFTH)
F2X =
RX - FX
1
1
F2Y =
RY - FY
F2R =
$\sqrt{F2X^2+F2Y^2}$
TAN= $\frac{F2Y}{F2X}$
RF2TH =
$\tan^{-1}$ TAN
Change
RF2TH
to Degrees
Write
RR, RTH
FR, FTH
F2R, F2TH
Last
Problem
no
13
yes
End

IBM FORTRAN Coding Form

```
C PROGRAM MISFOR
C ENTER RADIUS TO THE NEAREST HUNDRETH AND THETA TO THE NEAREST HUND
C RETH OF A DEGREE
   13 READ (2,1) RR, FR, RTH, FTH
    1 FORMAT (4F10.2)
      RRTH=RTH*3.1416/180.
      RFTH=FTH*3.1416/180.
      RX=RR*COS(RRTH)
      FX=FR*COS(RFTH)
      RY=RR*SIN(RRTH)
      FY=FR*SIN(RFTH)
      F2X=RX-FX
      F2Y=RY-FY
      F2R=SQRT(F2X**2+F2Y**2)
      TAN=F2Y/F2X
      RF2TH=ATAN(TAN)
      F2TH=RF2TH*180./3.1416
      WRITE(3,2)
    2 FORMAT(2X,' MAG OF RES',2X,'DIR OF RES',2X,'MAG OF FOR',2X,'DIR
     1OF FOR',8X,'MAG OF FOR2',2X,'DIR OF FOR2')
      WRITE(3,3) RR, RTH, FR, FTH, F2R, F2TH
    3 FORMAT (F10.2,3X,F10.2,3X,F10.2,3X,F10.2,3X,F10.3,3X,F10.2)
      GO TO 13
   23 CALL EXIT
```

RITRI

To practice the technique of using subroutines, students wrote program RITRI. Most of the techniques had already been covered and this was the culminating activity for some students. Others who were more ambitious wrote similar programs for the solution of any triangle. This program clearly illustrates the need for flowcharting, as well as control cards needed for naming the main program, and for storing and deleting the program and the subroutines.

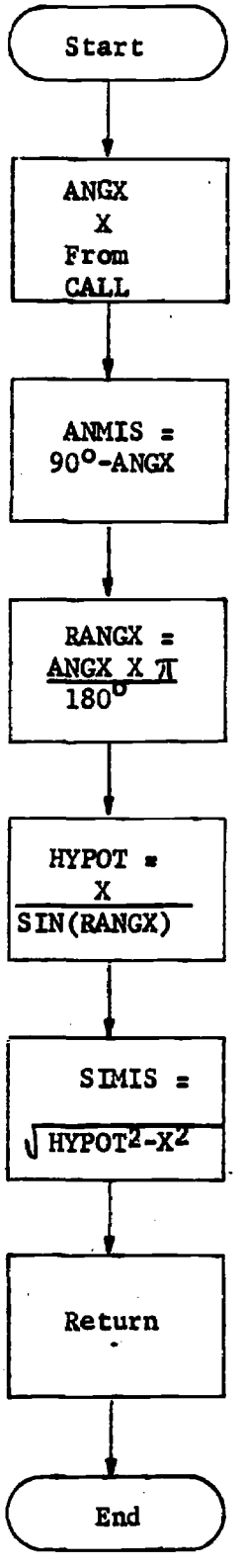
SUBROUTINES

A. AOPSI

Subroutine AOPSI solves for the missing parts when an angle (ANGX) and the side (X) are given. The values which are used in the subroutine are those values which are listed in the CALL statement. The results are returned to the main program so that they can be labeled properly in the printed output.

VARIABLES

ANGX - given angle (A or B)
X - side opposite the given angle (a or b)
ANMIS - missing angle (B or A)
RANGX - radians in angle X
HYPOT - hypotenuse (c)
SIMIS - missing side (b or a)

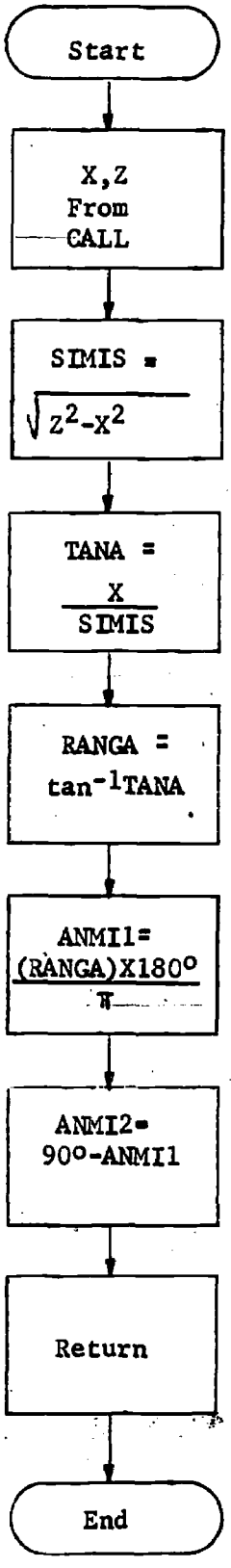


B. SIHYP

Subroutine SIHYP solves for the missing parts when a side (X) and the hypotenuse (Z) are known. The dummy variables X and Z are replaced by the values used in the CALL statement.

VARIABLES

- X - given side (a or b)
- Z - hypotenuse (c)
- SIMIS - missing side (b or a)
- TANA - tangent of angle opposite the given side (A or B)
- RANGA - radians in angle (A or B)
- ANMI1 - degrees in the angle opposite the given side (A or B)
- ANMI2 - degrees in other missing angle (B or A)

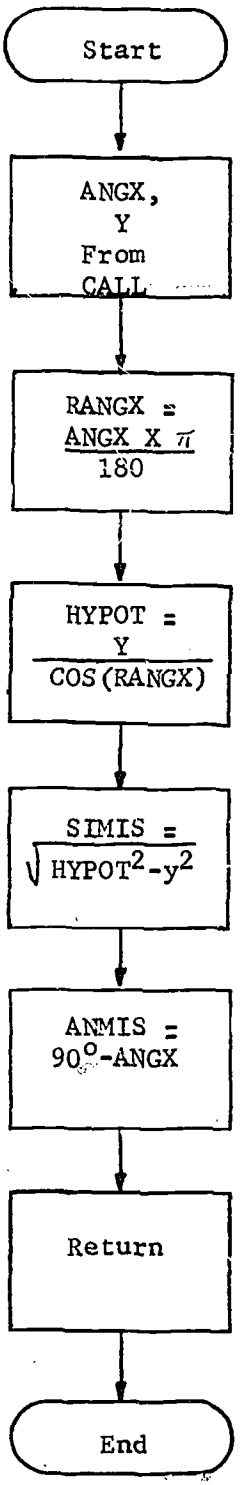


C. ANADS

Subroutine ANADS solves for the missing parts when an angle (ANGX) and the adjacent side (Y) are given.

VARIABLES

ANGX - given angle (A or B)
Y - side adjacent to the given side (b or a)
RANGX - radians in given angle
HYPOT - hypotenuse (c)
SIMIS - missing side (a or b)
ANMIS - missing angle (B or A)

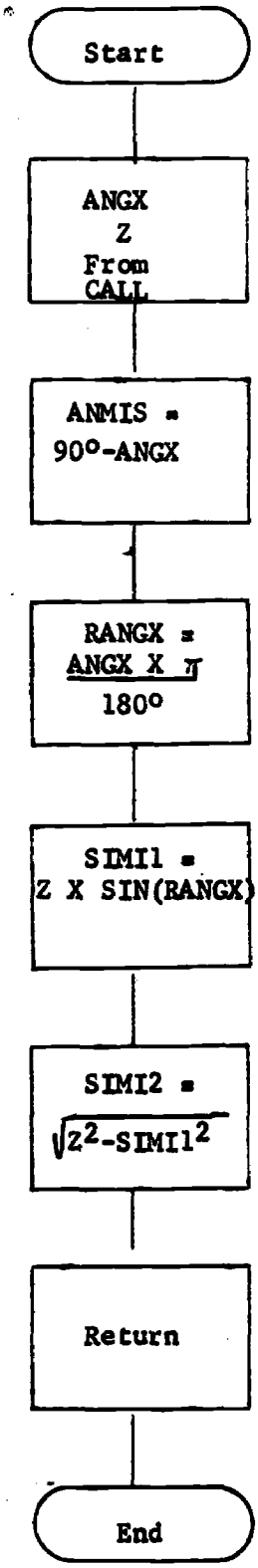


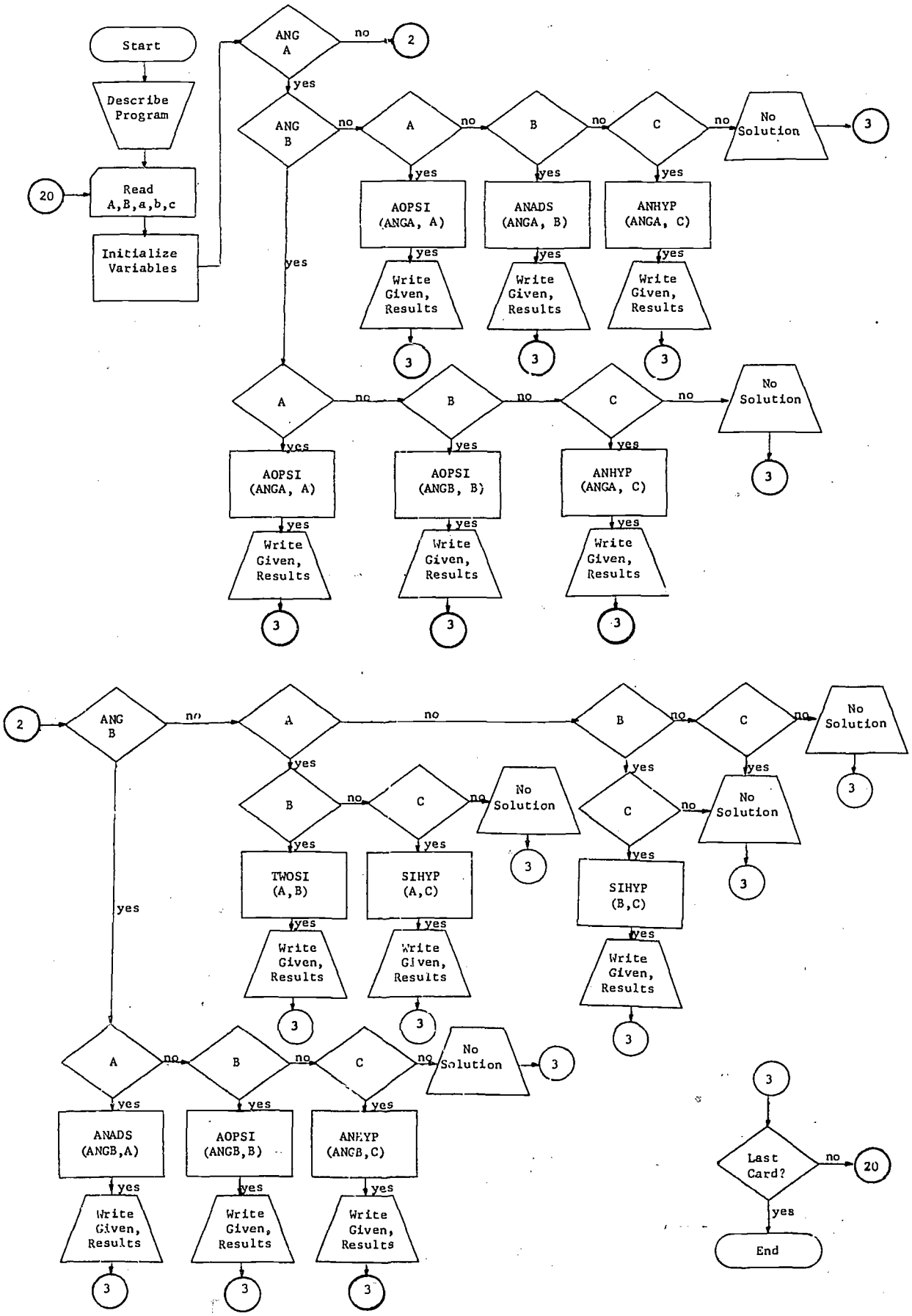
D. ANHYP(ANGX,Z)

Subroutine ANHYP solves for the missing parts when an angle (ANGX) and the hypotenuse (Z) are given.

VARIABLES

ANGX - given angle (A or B)
Z - hypotenuse (c)
ANMIS - missing angle (B or A)
RANGX - radians in given angle
SIMI1 - side opposite the given angle (a or b)
SIMI2 - side adjacent to the given angle (b or a)




Start
Describe Program
20
Read A,B,a,b,c
Initialize Variables
ANG A
no
2
yes
ANG B
no
A
no
B
no
C
no
No Solution
3
yes
AOPSI (ANGA, A)
ANADS (ANGA, B)
ANHYP (ANGA, C)
Write Given, Results
A
no
B
no
C
no
No Solution
AOPSI (ANGA, A)
AOPSI (ANGB, B)
ANHYP (ANGA, C)
Write Given, Results
2
ANG B
no
A
no
B
no
C
no
No Solution
yes
B
no
C
no
No Solution
C
no
No Solution
TWOSI (A,B)
SIHYP (A,C)
SIHYP (B,C)
Write Given, Results
A
no
B
no
C
no
No Solution
ANADS (ANGB,A)
AOPSI (ANGB,B)
ANHYP (ANGB,C)
Write Given, Results
3
Last Card?
no
20
yes
End

IBM

FORTRAN Coding Form

PROGRAM RITRI

1 6

FORTRAN STATEMENT

```
*NAME  RITRI
       COMMON ANMIS,SIMIS,HYPOT,ANMI1,ANMI2,SIMI1,SIMI2
C THIS PROGRAM WILL SOLVE ANY RIGHT TRIANGLE IF TWO PARTS ARE KNOWN ONE
C OF WHICH MUST BE A SIDE.
       WRITE (3,1)
     1 FORMAT (2X,'THIS PROGRAM WILL SOLVE FOR THE MISSING PARTS OF A RIG
      1HT TRIANGLE IF TWO PARTS ARE GIVEN OTHER THAN TWO ANGLES.THE',/,2X
      2,'DATA IS ENTERED ON CARDS,ANGLES TO THE NEAREST HUNDREDTH OF A DE
      3GREE AND UNKNOWN PARTS AS THE CONSTANT 999.')
   100 READ (2,2)ANGA,ANGB,A,B,C
     2 FORMAT (5F7.2)
       ANMIS =0.
       HYPOT =0.
       SIMIS =0.
       ANMI1 =0.
       ANMI2 =0.
       SIMI1 =0.
       SIMI2 =0.
       IF (ANGA-999.) 10,20,10
    10 IF (ANGB-999.) 30,15,30
    30 IF (A-999.) 71,40,71
    71 CALL AOPSI (ANGA,A)
       WRITE (3,72)ANGA,A,ANMIS,SIMIS,HYPOT
```

IBM

FORTRAN Coding Form

2 6

FORTRAN STATEMENT

```
    72 FORMAT (2X,'GIVEN',4X,'ANGLE A=',F7.2,2X,'A=',F7.2,/,
      1 2X,'ANSWERS',2X,'ANGLE B=',F7.2,2X,'B=',F7.2,2X,'C=',F7.2)
       GO TO 100
    40 IF (B-999.) 70,50,70
    70 CALL AOPSI (ANGB,B)
       WRITE (3,73)ANGB,B,ANMIS,SIMIS,HYPOT
    73 FORMAT (2X,'GIVEN',4X,'ANGLE B=',F7.2,2X,'B=',F7.2,/,
      1 2X,'ANSWERS',2X,'ANGLE A=',F7.2,2X,'A=',F7.2,2X,'C=',F7.2)
       GO TO 100
    50 IF (C-999.) 90,60,90
    90 CALL ANHYP (ANGA,C)
       WRITE (3,91)ANGA,C,ANMIS,SIMI2,SIMI1
    91 FORMAT (2X,'GIVEN',4X,'ANGLE A=',F7.2,2X,'C=',F7.2,/,
      1 2X,'ANSWERS',2X,'ANGLE B=',F7.2,2X,'B=',F7.2,2X,'A=',F7.2)
       GO TO 100
    15 IF (A-999.) 71,16,71
    16 IF (B-999.) 81,17,81
    81 CALL ANADS (ANGA,B)
       WRITE (3,82)ANGA,B,ANMIS,SIMIS,HYPOT
    82 FORMAT (2X,'GIVEN',4X,'ANGLE A=',F7.2,2X,'B=',F7.2,/,
      1 2X,'ANSWERS',2X,'ANGLE B=',F7.2,2X,'A=',F7.2,2X,'C=',F7.2)
       GO TO 100
```

IBM

FORTRAN Coding Form

FORTRAN STATEMENT

```
17  IF (C-999.)90,60,90
20  IF (ANGB-999.) 31,18,31
31  IF (A-999.) 80,41,80
80  CALL ANADS (ANGB,A)
    WRITE (3,32) ANGB,A,ANMIS,SIMIS,HYPOT
32  FORMAT (2X,'GIVEN',4X,'ANGLE B=',F7.2,2X,'A=',F7.2,/,
   1 2X,'ANSWERS',2X,'ANGLE A=',F7.2,2X,'B=',F7.2,2X,'C=',F7.2)
    GO TO 100
41  IF (B-999.) 70,51,70
51  IF (C-999.) 92,60,92
92  CALL ANHYP (ANGB,C)
    WRITE (3,93) ANGB,C,ANMIS,SIMI2,SIMI1
93  FORMAT (2X,'GIVEN',4X,'ANGLE B=',F7.2,2X,'C=',F7.2,/,
   1 2X,'ANSWERS',2X,'ANGLE A=',F7.2,2X,'A=',F7.2,2X,'B=',F7.2)
    GO TO 100
18  IF (A-999.) 19,21,19
19  IF (B-999.) 112,24,112
112 RANGA =ATAN(A/B)
    ANGA =RANGA*180./3.1416
    ANGB =90.-ANGA
    C =SQRT(A**2+B**2)
    WRITE (3,113) A,B,ANGB,ANGA,C
```

IBM

FORTRAN Coding Form

FORTRAN STATEMENT

```
113 FORMAT (2X,'GIVEN',4X,'A=',F7.2,2X,'B=',F7.2,/,
   1 2X,'ANSWERS',2X,'ANGLE A=',F7.2,2X,'ANGLE B=',F7.2,2X,'C=',F7.2)
    GO TO 100
24  IF (C-999.) 111,60,111
111 CALL SIHYP (A,C)
    WRITE (3,114) A,C,ANMI1,ANMI2,SIMIS
114 FORMAT (2X,'GIVEN',4X,'A=',F7.2,2X,'C=',F7.2,/,
   1 2X,'ANSWERS',2X,'ANGLE A=',F7.2,2X,'ANGLE B=',F7.2,2X,'B=',F7.2)
    GO TO 100
21  IF (B-999.) 22,60,22
22  IF (C-999.) 110,60,110
110 CALL SIHYP (B,C)
    WRITE (3,115) B,C,ANMI2,ANMI1,SIMIS
115 FORMAT (2X,'GIVEN',4X,'B=',F7.2,2X,'C=',F7.2,/,
   1 2X,'ANSWERS',2X,'ANGLE A=',F7.2,2X,'ANGLE B=',F7.2,2X,'A=',F7.2)
    GO TO 100
60  WRITE (3,61)
61  FORMAT (2X,'CHECK YOUR INFORMATION GIVEN,THERE CAN BE NO SOLUTION
   1 WITH THE DATA YOU HAVE ENTERED.')
    GO TO 100
9999 CALL EXIT
    END
```

IBM

FORTRAN Coding Form

SUBROUTINES for PROGRAM RITRI

5 6

FORTRAN STATEMENT

```
      SUBROUTINE ANADS(ANGX,Y)
      COMMON ANMIS,SIMIS,HYPOT,ANMI1,ANMI2,SIMI1,SIMI2
      RANGX =ANGX*3.1416/180.
      HYPOT =Y/COS(RANGX)
      SIMIS =SQRT(HYPOT**2-Y**2)
      ANMIS =90.-ANGX
      RETURN
      END

      SUBROUTINE ANHYP(ANGX,Z)
      COMMON ANMIS,SIMIS,HYPOT,ANMI1,ANMI2,SIMI1,SIMI2
      ANMIS =90.-ANGX
      RANGX =ANGX*3.1416/180.
      SIMI1 =Z*SIN(RANGX)
      SIMI2 =SQRT(Z**2-SIMI1**2)
      RETURN
      END
```

IBM

FORTRAN Coding Form

SUBROUTINES for RITRI (CONT'D)

6 6

FORTRAN STATEMENT

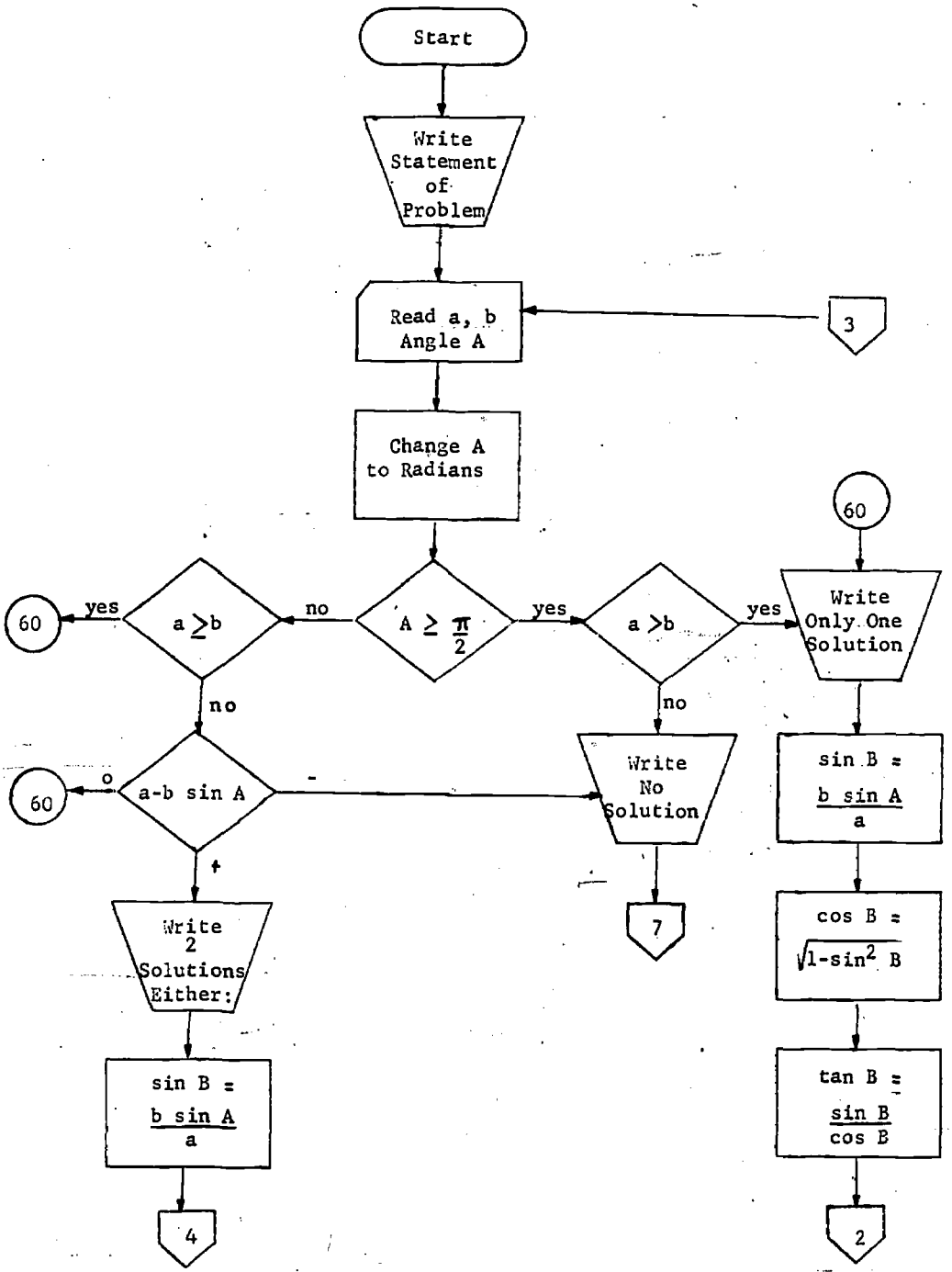
```
      SUBROUTINE SIHYP(X,Z)
      COMMON ANMIS,SIMIS,HYPOT,ANMI1,ANMI2,SIMI1,SIMI2
      SIMIS =SQRT(Z**2-X**2)
      TANA  =X/SIMIS
      RANGA =ATAN(TANA)
      ANMI1 =RANGA*180./3.1416
      ANMI2 =90.-ANMI1
      RETURN
      END

      SUBROUTINE AOPSI(ANGX,X)
      COMMON ANMIS,SIMIS,HYPOT,ANMI1,ANMI2,SIMI1,SIMI2
      ANMIS =90.-ANGX
      RANGX =ANMIS*3.1416/180.
      HYPOT =X/SIN(RANGX)
      SIMIS =SQRT(HYPOT**2-X**2)
      RETURN
      END
```

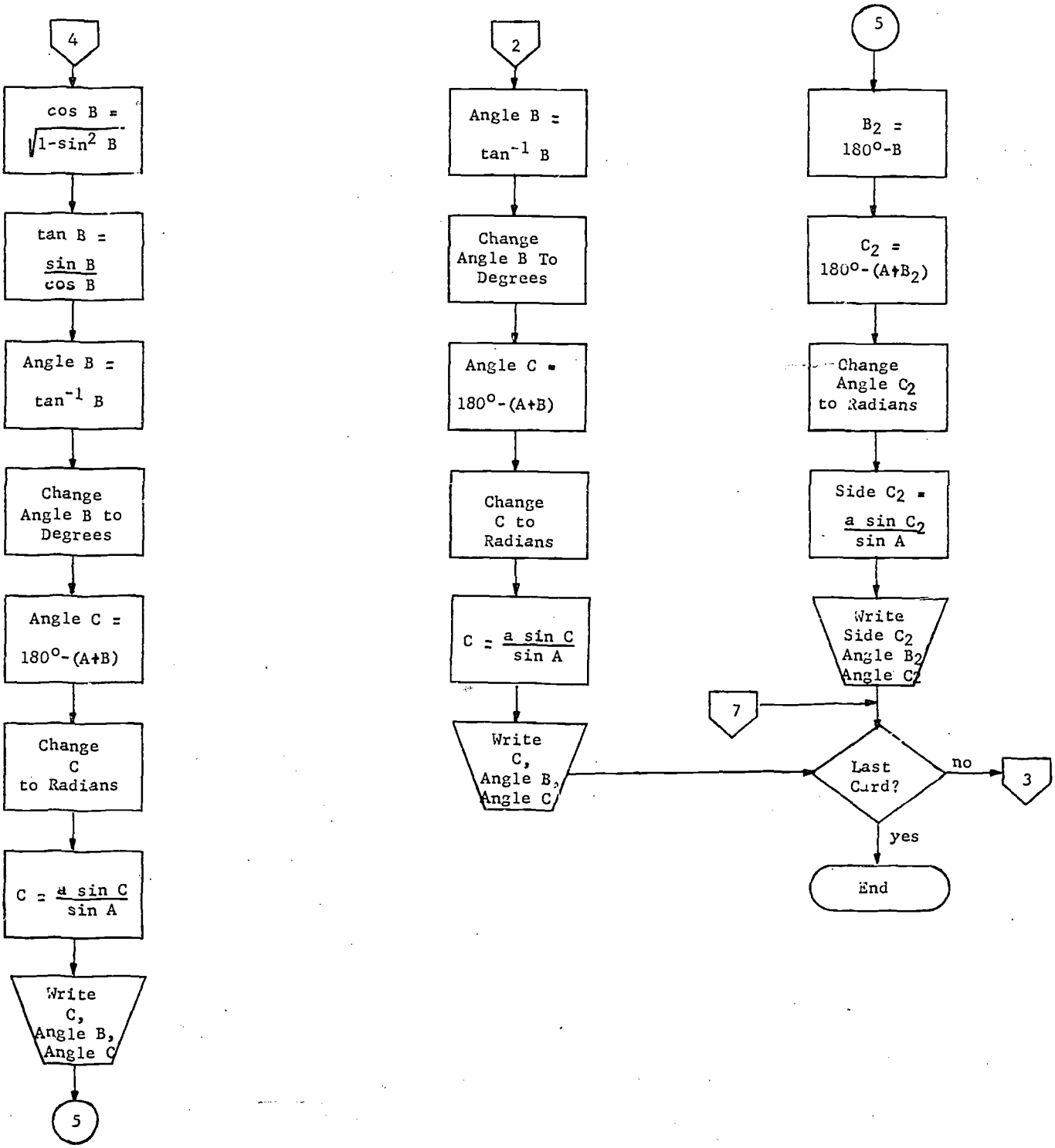
THE AMBIGUOUS CASE (S.S.A.)

This program will give a complete solution for a triangle if twn sides and the angle npposite one of these sides are given. The sides must be less than 1000 and space for 4 decimal places is provided. The known parts are entered in the order a, b, angle A. Angles are in degrees to 4 decimal places.

VARIABLES

a - side a
b - side b
ANGA - angle A
XANGA - radians in angle A




4
cos B = √1-sin² B
tan B = sin B / cos B
Angle B = tan⁻¹ B
Change Angle B to Degrees
Angle C = 180°-(A+B)
Change C to Radians
C = a sin C / sin A
Write C, Angle B, Angle C
5
2
Angle B = tan⁻¹ B
Change Angle B To Degrees
Angle C = 180°-(A+B)
Change C to Radians
C = a sin C / sin A
Write C, Angle B, Angle C
5
B2 = 180°-B
C2 = 180°-(A+B2)
Change Angle C2 to Radians
Side C2 = a sin C2 / sin A
Write Side C2 Angle B2 Angle C2
7
Last Card?
no
3
yes
End

IBM

FORTRAN Coding Form

1 3

FORTRAN STATEMENT

```
C THE AMBIGUOUS CASE, S.S.A.
      WRITE(3,1)
    1 FORMAT(2X,'GIVEN TWO SIDES OF A TRIANGLE AND AN ANGLE OPPOSITE ONE
     1OF THEM SUCH THAT')
    2 READ(2,3)A,B,ANGA
    3 FORMAT(3F10.4)
      WRITE(3,4)A,B,ANGA
    4 FORMAT(2X,'A=',F10.4,2X,'B=',F10.4,2X,'ANGA=',F10.4,2X,'DEGREES,')
      XANGA=ANGA*3.1416/180.
      IF(XANGA-1.5708)30,40,40
   40 IF(A-B)50,50,60
   50 WRITE(3,5)
    5 FORMAT(2X,'THEN THERE IS NO SOLUTION.THIS TRIANGLE CANNOT EXIST.')
      GO TO 2
   60 WRITE(3,6)
    6 FORMAT(2X,'THEN THERE IS ONE AND ONLY ONE SOLUTION.')
      SINB=B*SIN(XANGA)/A
      COSB=SQRT(1-SINB**2)
      TANB=SINB/COSB
      XANGB=ATAN(TANB)
      ANGB=XANGB*180./3.1416
      ANGC=180.-(ANGA+ANGB)
```

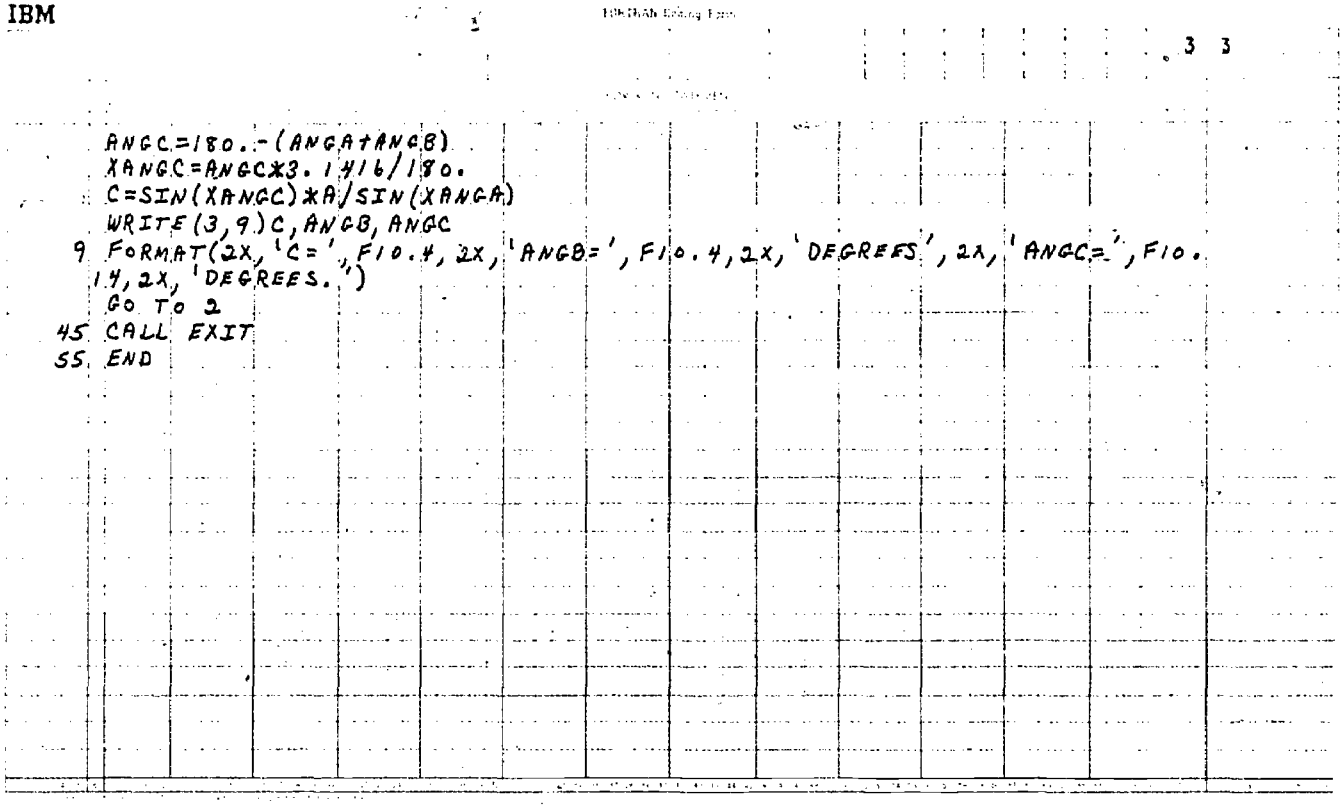
IBM

FORTRAN Coding Form

2 3

FORTRAN STATEMENT

```
      XANGC=ANGC*3.1416/180.
      C=SIN(XANGC)*A/SIN(XANGA)
      WRITE(3,7)C,ANGB,ANGC
    7 FORMAT(2X,'C=',F10.4,2X,'ANGB=',F10.4,2X,'DEGREES',2X,'ANGC=',F10.
     14,2X,'DEGREES.')
      GO TO 2
   30 IF(A-B)70,60,60
   70 IF(A-B*SIN(XANGA))50,60,90
   90 WRITE(3,8)
    8 FORMAT(2X,'THERE ARE TWO SOLUTIONS.EITHER')
      SINB=B*SIN(XANGA)/A
      COSB=SQRT(1-SINB**2)
      TANB=SINB/COSB
      XANGB=ATAN(TANB)
      ANGB=XANGB*180./3.1416
      ANGC=180.-(ANGA+ANGB)
      XANGC=ANGC*3.1416/180.
      C=SIN(XANGC)*A/SIN(XANGA)
      WRITE(3,10)C,ANGB,ANGC
   10 FORMAT(2X,'C=',F10.4,2X,'ANGB=',F10.4,2X,'DEGREES',2X,'ANGC=',F10.
     14,2X,'DEGREES,OR')
      ANGB=180.-ANGB
```



```
IBM                                                          3 3
      ANGC=180.-(ANGA+ANGB)
      XANGC=ANGC*3.1416/180.
      C=SIN(XANGC)*A/SIN(XANGA)
      WRITE(3,9)C,ANGB,ANGC
    9 FORMAT(2X,'C=',F10.4,2X,'ANGB=',F10.4,2X,'DEGREES',2X,'ANGC=',F10.
     1 4,2X,'DEGREES.')
      GO TO 2
   45 CALL EXIT
   55 END
```

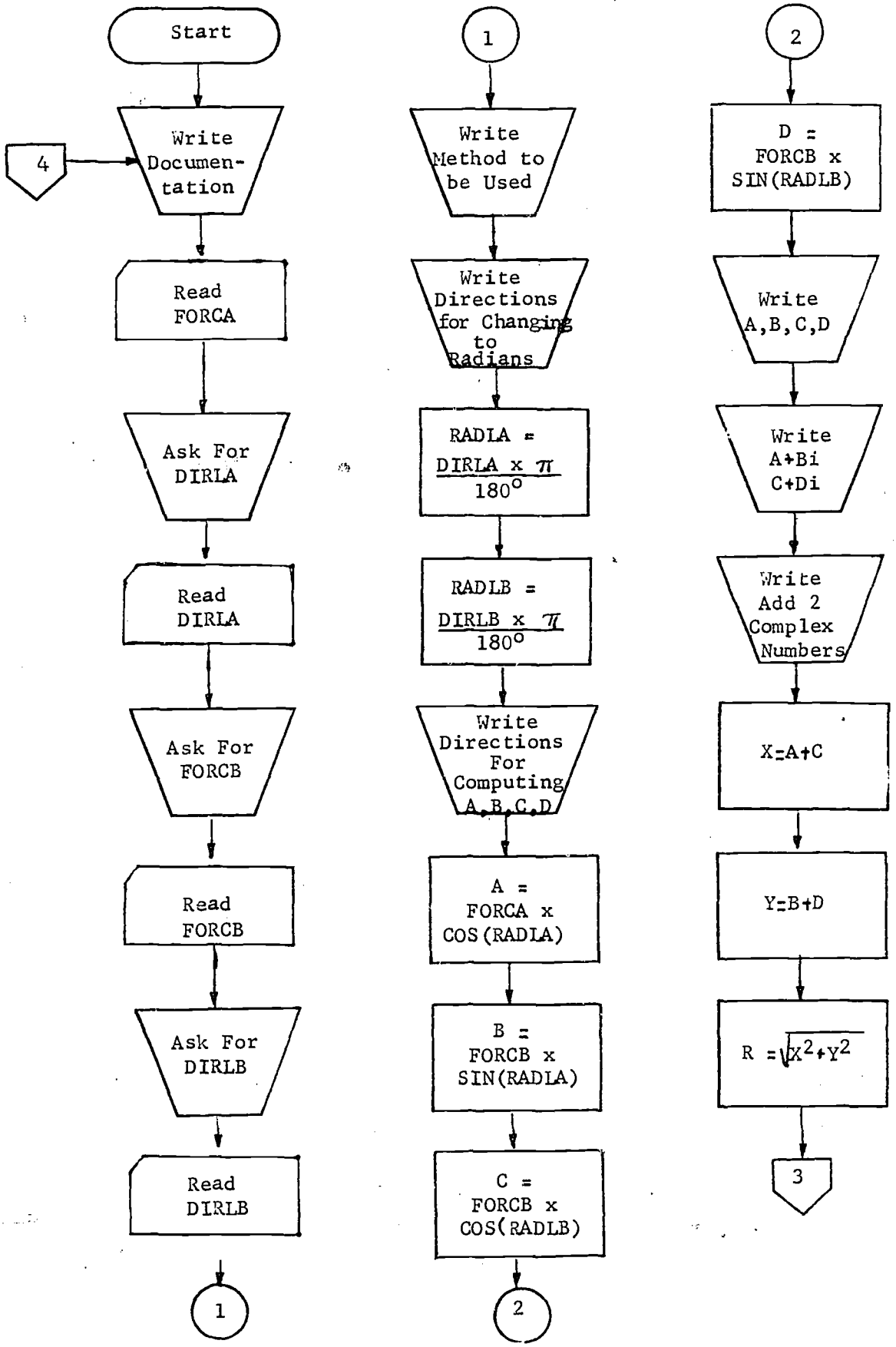
REFOR

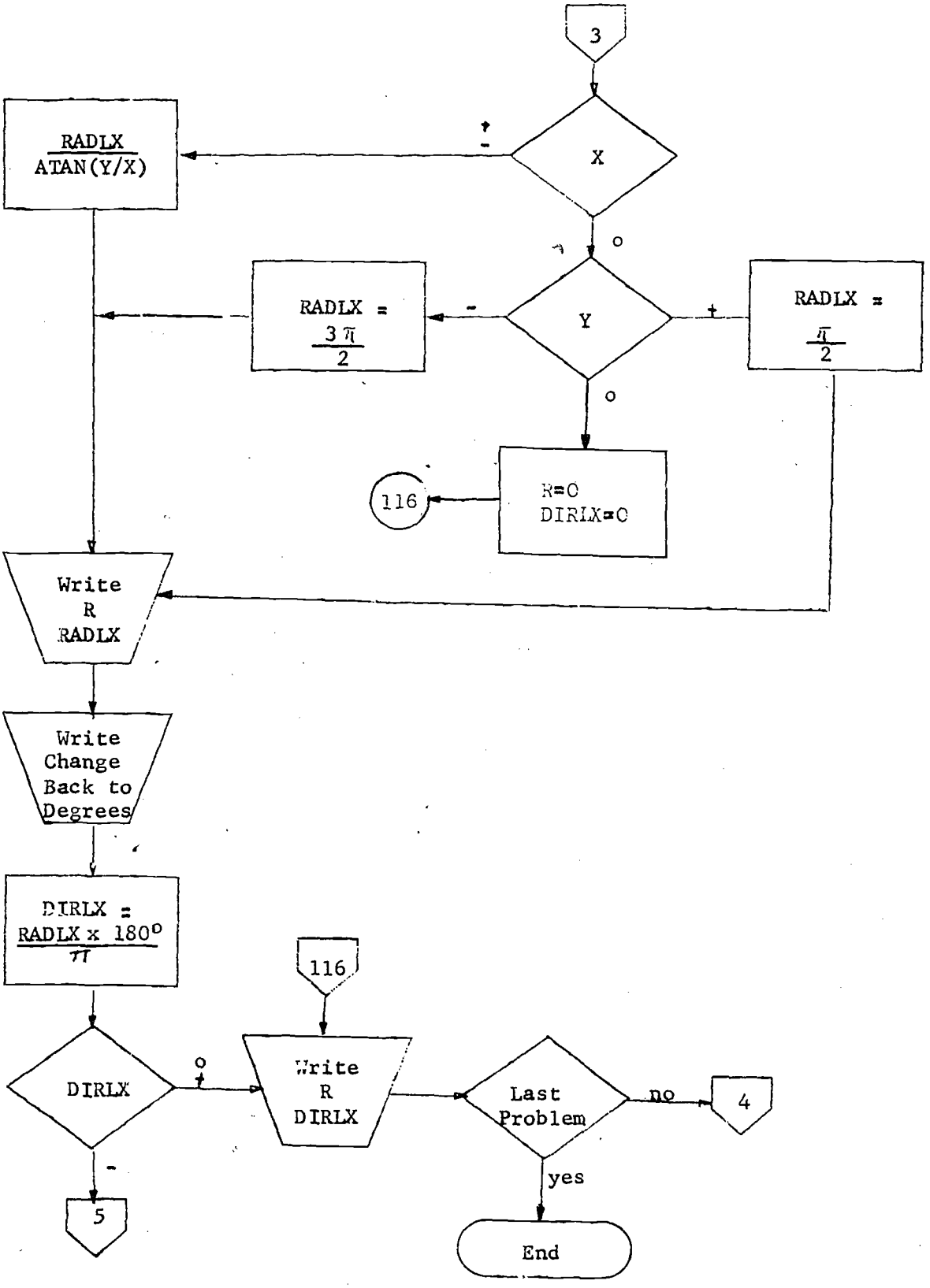
This program was written as an experiment in using the computer to teach the lesson. Not only are results computed, but the method of computation is explained, so that the student can use the output as a study guide. Since the student must enter the values on the typewriter, some additional interest is generated. Though the teacher who originated this program used it frequently as a demonstration program, she felt that even more improvement should be made by allowing for even more student participation.

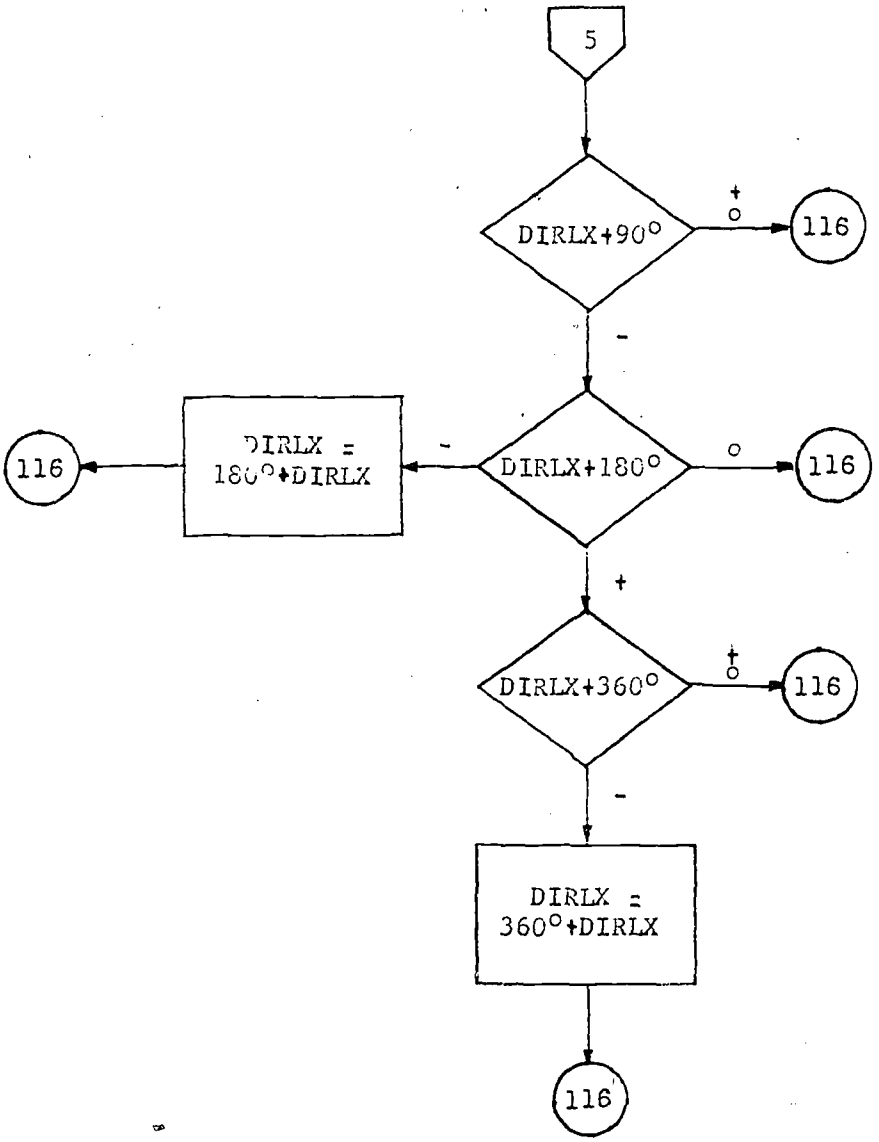
The magnitude and direction of each of two forces are entered. The angles are in degrees. Both angles and magnitude may contain 2 decimal places. The method of solution is addition of the complex numbers which represent the forces.

VARIABLES

FORCA - magnitude of first force
DIRLA - direction angle of first force
FORCB - magnitude of second force
DIRLB - direction angle of second force
RADLA - radians in first angle
RADLB - radians in second angle
A - X component of first force
B - X component of second force
C - Y com onent of first force
D - Y component of second force
X - X component of resultant
Y - Y component of resultant
R - magnitude of resultant
DIRLX - direction angle of resultant


Start
4
Write Documen-tation
Read FORCA
Ask For DIRLA
Read DIRLA
Ask For FORCB
Read FORCB
Ask For DIRLB
Read DIRLB
1
1
Write Method to be Used
Write Directions for Changing to Radians
RADLA = DIRLA x π / 180°
RADLB = DIRLB x π / 180°
Write Directions For Computing A,B,C,D
A = FORCA x COS(RADLA)
B = FORCB x SIN(RADLA)
C = FORCB x COS(RADLB)
2
2
D = FORCB x SIN(RADLB)
Write A,B,C,D
Write A+Bi C+Di
Write Add 2 Complex Numbers
X=A+C
Y=B+D
R = √(X²+Y²)
3


3
X
+
-
RADLX
ATAN(Y/X)
0
Y
-
RADLX =
3π
2
+
RADLX =
π
2
0
116
R=0
DIRLX=0
Write
R
RADLX
Write
Change
Back to
Degrees
DIRLX =
RADLX x 180°
π
116
DIRLX
0
+
Write
R
DIRLX
Last
Problem
no
4
yes
-
5
End


5
DIRLX+90°
+
0
116
-
DIRLX+180°
-
DIRLX =
180°+DIRLX
116
0
116
+
DIRLX+360°
+
0
116
-
DIRLX =
360°+DIRLX
116

```
PAGE    1
// JOB
LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0001         0001         0000
V2 M05    ACTUAL  8K  CONFIG  8K
// FOR
*IOCS(CARD,TYPEWRITER,KEYBOARD,1132 PRINTER,DISK,PAPER TAPE)
*LIST  SOURCE PROGRAM
C       PROGRAM REFOR
  222   WRITE (1,1)
      1 FORMAT ('THIS PROGRAM WILL FIND THE RESULTANT OF TWO FORCES WHICH
     1ACT SIMULTANEOUSLY AT THE   SAME POINT ON A BODY'//,
     2'ENTER THE MAGNITUDE OF THE FIRST FORCE TO THE NEAREST HUNDREDTH')
        READ(6,2)FORCA
      2 FORMAT (F7.2)
        WRITE(1,3)
      3 FORMAT (' ENTER THE DIRECTION ANGLE OF THE FIRST FORCE TO THE NEAR
     1EST HUNDREDTH')
        READ (6,2)DIRLA
        WRITE(1,4)
      4 FORMAT (' ENTER THE MAGNITUDE OF THE SECOND FORCE TO THE  NEAREST
     1HUNDREDTH')
        READ (6,2)FORCB
        WRITE (1,5)
      5 FORMAT (' ENTER THE DIRECTION ANGLE OF THE SECOND FORCE TO THE NEA
     1REST HUNDREDTH')
        READ (6,2)DIRLB
        WRITE (1,6)FORCA,DIRLA,FORCA,DIRLA,FORCB,DIRLB,FORCB,DIRLB
      6 FORMAT( 'THE FIRST FORCE CAN BE REPRESENTED BY THE COMPLEX NUMBER
     1A+BI,A=',F7.2,'COS',F7.2,'AND B=',F7.2,'SIN',F7.2,'AND'/,
     2'THE SECOND FORCE CAN BE REPRESENTED BY THE COMPLEX NUMBER C+DI,
     3C=',F7.2,'COS',F7.2,'AND D=',F7.2,'SIN',F7.2)
        WRITE (1,7)DIRLA,DIRLB
      7 FORMAT ( 'THE TABLES IN THIS COMPUTER ARE IN RADIANS SO IT IS NECE
     1SSARY TO CHANGE DEGREES TO'//,'RADIANS. WE DO THIS BY MULTIPLYING
     2 THE NUMBER OF DEGREES BY PI AND DIVIDING BY 180.'//,
     3'RADLA=',F7.2,' X 3.1416/180.',///,
     4'RADLB=',F7.2,' X 3.1416/180.'//)
        RADLA=DIRLA*3.1416/180.
        RADLB=DIRLB*3.1416/180.
        WRITE (1,8)RADLA,RADLB
      8 FORMAT (' RADLA=',F7.4,5X,'RADLB=',F7.4 //)
        WRITE(1,9) FORCA,RADLA, FORCA,RADLA,FORCB,RADLB,FORCB,RADLB
      9 FORMAT (' A=',F7.2,'COS',F7.4,/,
     1'B=',F7.2,'SIN',F7.4,/,
     2'C=',F7.2,'COS',F7.4,/,
     3 'D=',F7.2,'SIN',F7.4//)
        A=FORCA*COS(RADLA)
        B=FORCA*SIN(RADLA)
        C=FORCB*COS(RADLB)
        D=FORCB*SIN(RADLB)
        WRITE(1,10)A,B,C,D
     10 FORMAT ( 'A=',F9.4,/,
     1 'B=',F9.4,/,
     2'C=',F9.4,/,
     3'D=',F9.4//)
        WRITE (1,11)A,B,C,D
     11 FORMAT ('A+BI=',F7.2,'+',F7.2,'I',/,
     1'C+DI=',F7.2,'+',F7.2,'I'//)
        WRITE (1,12)
     12 FORMAT (' TO FIND THE RESULTANT WE MUST ADD THESE TWO COMPLEX NUMB
     1ERS')
        X=A+C
        Y=B+D
        WRITE(1,13)X,Y
     13 FORMAT(' THE COMPLEX NUMBER THAT REPRESENTS THE RESULTANT IS',//,
     1F7.2,'+',F7.2,'I'//)
        WRITE (1,14)
     14 FORMAT ( 'TO CHANGE THIS COMPLEX NUMBER WE USE THE FORMULA',/,
     1'R=SQRT(X**2+Y**2),AND TAN(RADLX)=Y/X'//)
        R=SQRT(X**2+Y**2)
```

```
      IF (X) 115,25,115
   25 IF(Y)26,127,28
  127 R=0.
      DIRLX=0.
      GO TO 116
   26 RADLX=3*3.1416/2.
      GO TO 126
   28 RADLX=3.1416/2.
      GO TO 126
  115 RADLX=ATAN(Y/X)
  126 WRITE (1,15)R,RADLX
   15 FORMAT( 'R=',F7.2,'RADLX=',F9.4,//,
     1'CHANGING RADIANS BACK TO DEGREES WE MULTIPLY RADIANS BY 180. AND
     2DIVIDE BY PI')
      DIRLX=RADLX*180./3.1416
      IF (DIRLX)35,116,116
   35 IF (DIRLX+90.)36,116,116
   36 IF (DIRLX+180.)38,116,37
   37 DIRLX=180.+DIRLX
      GO TO 116
   38 IF (DIRLX+360.)39,116,116
   39 DIRLX=360.+DIRLX
      GO TO 116
  116 WRITE (1,16)R,DIRLX
   16 FORMAT ( 'THE RESULTANT FORCE IS',F7.2,/,
     1' THE DIRECTION ANGLE IS',F7.2,'DEGREES')
      WRITE (1,17)
   17 FORMAT (' IF YOU HAVE ANOTHER PROBLEM TO SOLVE ENTER .9999, IF NO
     1T ENTER 0.')
      READ (6,18)Z
   18 FORMAT (F7.4)
      IF (Z-.9999)111,222,111
  111 CALL EXIT
      END

FEATURES SUPPORTED
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES     38  PROGRAM    1152

 END OF COMPILATION
```

PROBLEM SUPPLEMENT

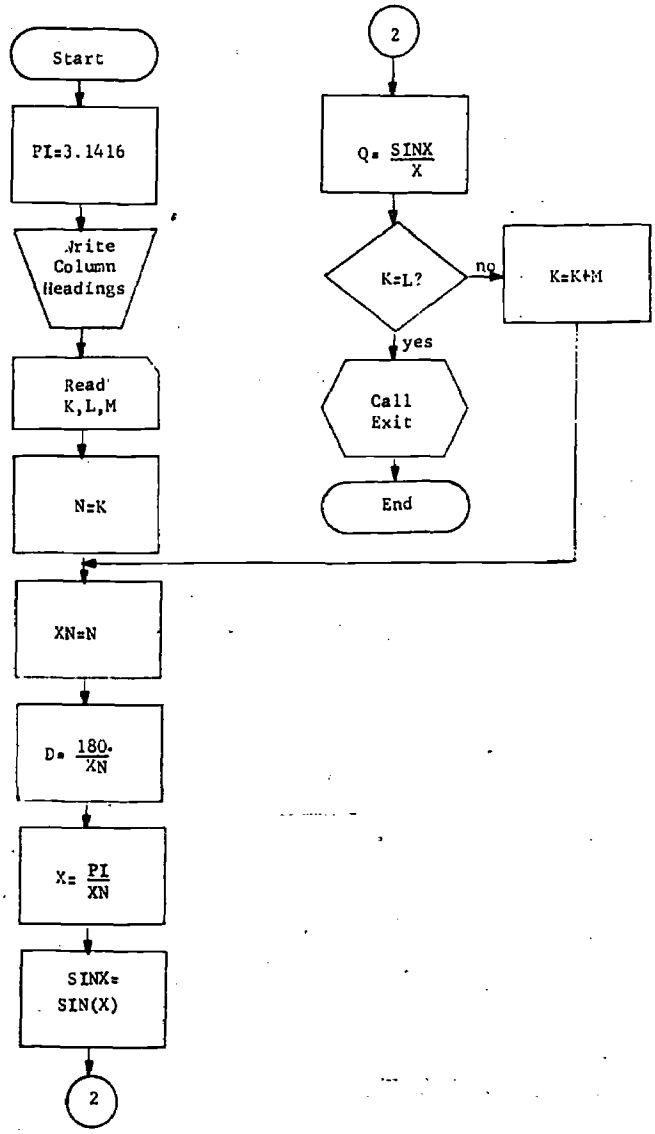
Advanced Mathematics Section III

LIMIT OF $\frac{SIN\ X}{X}$

This problem shows that the limit of SIN(X)/X approaches the value of one as X approaches zero.

INPUT
- K - lower limit of X
- L - upper limit of X
- M - interval

OUTPUT
- PI/N - notation for X in terms of PI
- D - degrees in X
- X - radians in X
- SINX — sine of X
- Q - SINX/X



IBM FORTRAN Coding Form

Limit of SINX/X

FORTRAN STATEMENT

```
      WRITE(3,1)
1     FORMAT(1X,'LIMIT OF SINX/X')
      PI=3.1416
      WRITE(3,2)
2     FORMAT(10X,'DEG.',7X,'RAD.',9X,'SINX',7X,'SINX/X')
      READ(2,3)K,L,M
3     FORMAT(3I3)
      DO 4 N=K,L,M
      XN=N
      D=180./XN
      X=PI/XN
      SINX=SIN(X)
      Q=SINX/X
4     WRITE(3,5)N,D,X,SINX,Q
5     FORMAT(1X,'PI/',I3,3X,F5.2,3X,F9.6,3X,F9.6,3X,F9.6)
6     CALL EXIT
      END
```

AREA UNDER A CURVE
(TRAPEZOIDAL RULE)

This program finds the area under the curve by the trapezoidal rule in the form:

$$AREA = ((X(2)-X(1))/2)*(Y(1)+2.*SSUM+Y(M))$$

INPUT

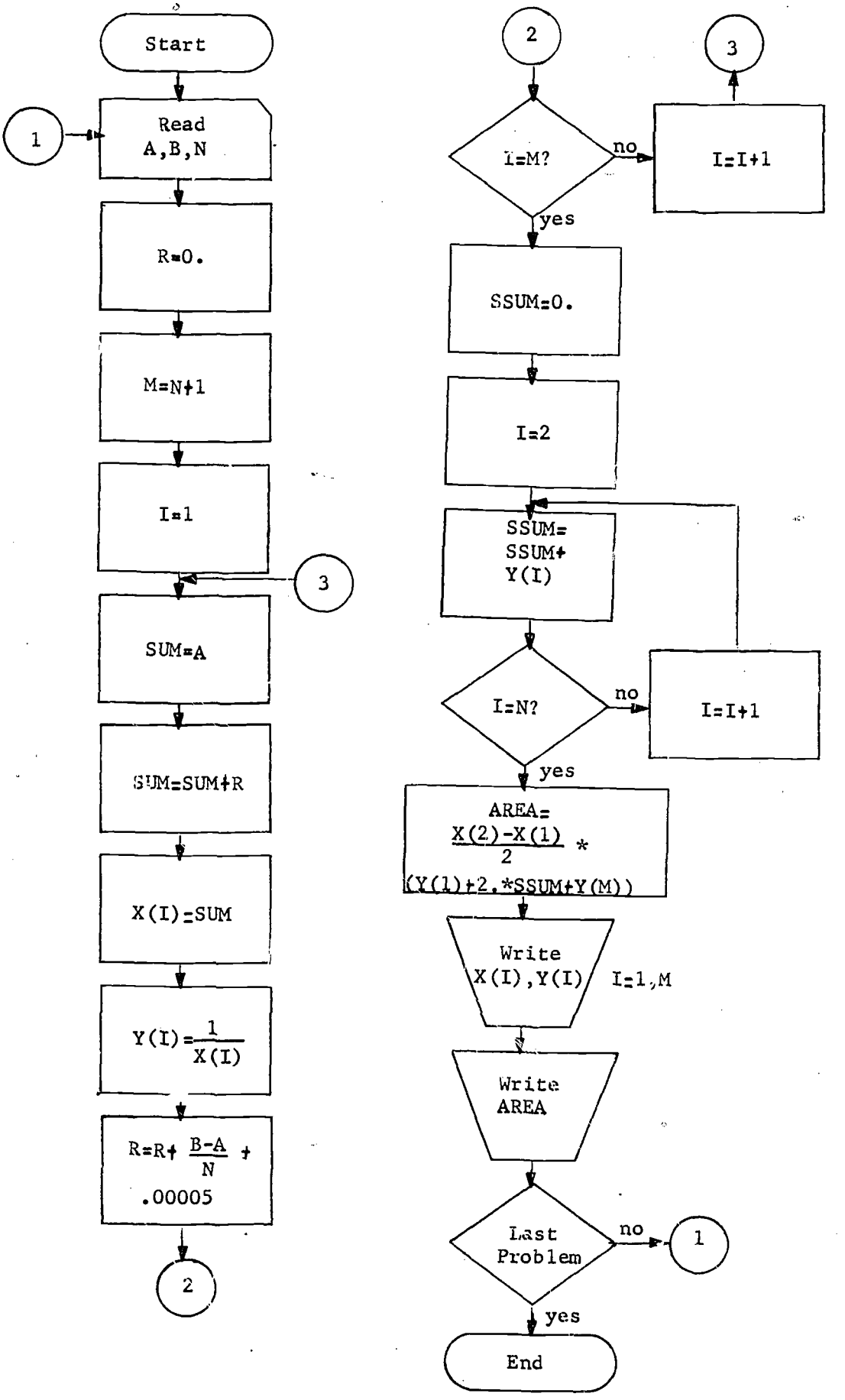
- A - lower limit(1)
- B - upper limit (2)
- N - number of intervals

FUNCTION: $y = 1/x$

OUTPUT:

- X - X coordinate
- Y - Y coordinate
- AREA

R is used to reduce error due to binary system (it is not necessary).


Start
1
Read
A,B,N
R=0.
M=N+1
I=1
3
SUM=A
SUM=SUM+R
X(I)=SUM
Y(I)=1/X(I)
R=R+ (B-A)/N + .00005
2
2
3
I=M?
no
I=I+1
yes
SSUM=0.
I=2
SSUM=
SSUM+
Y(I)
I=N?
no
I=I+1
yes
AREA=
(X(2)-X(1))/2 *
(Y(1)+2.*SSUM+Y(M))
Write
X(I),Y(I)
I=1,M
Write
AREA
Last
Problem
no
1
yes
End

IBM

FORTRAN Coding Form

1 1

FORTRAN STATEMENT

```
**INTEGRATION BY THE TRAPEZOIDAL RULE
      DIMENSION X(50), Y(50)
   40 READ(2,1)A,B,N
    1 FORMAT(2F5.2,I5)
      R=0.0
      M=N+1
      DO 20 I=1,M
      SUM=A
      SUM=SUM+R
      X(I)=SUM
      Y(I)=1./X(I)
      R=R+(B-A)/FLOAT(N)+.00005
   20 CONTINUE
      SSUM=0.0
      DO 30 I=2,N
   30 SSUM=SSUM+Y(I)
      AREA=(X(2)-X(1))/2.*(Y(1)+2.*SSUM+Y(M))
      WRITE(3,2) (X(I),Y(I),I=1,M)
    2 FORMAT(1X,'X=',F6.3,1X,'Y=',F6.3)
      WRITE(3,3) AREA
    3 FORMAT(1X,'AREA=',F6.3)
      GO TO 40
   50 CALL EXIT
      END
```

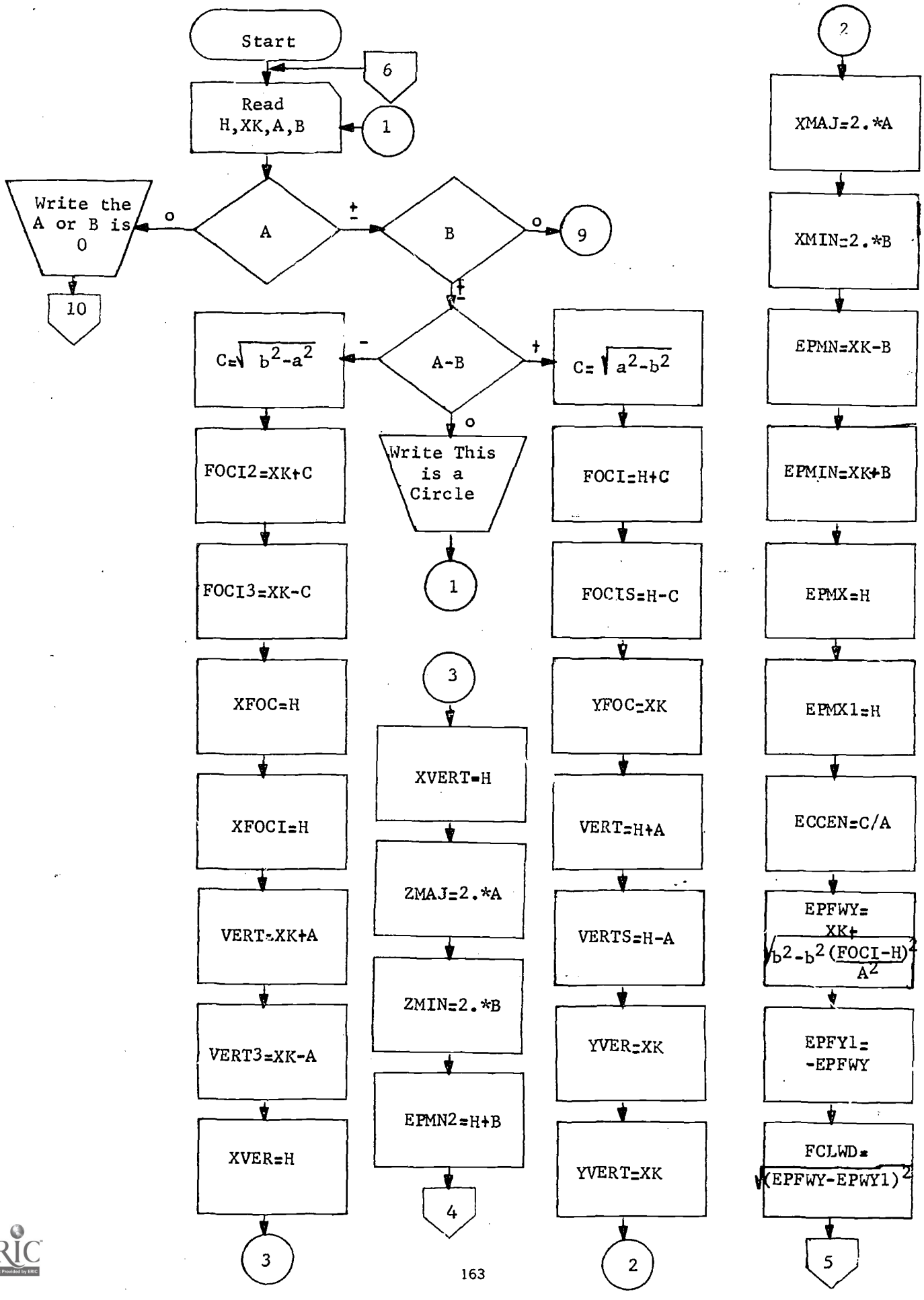
ELLIPSE

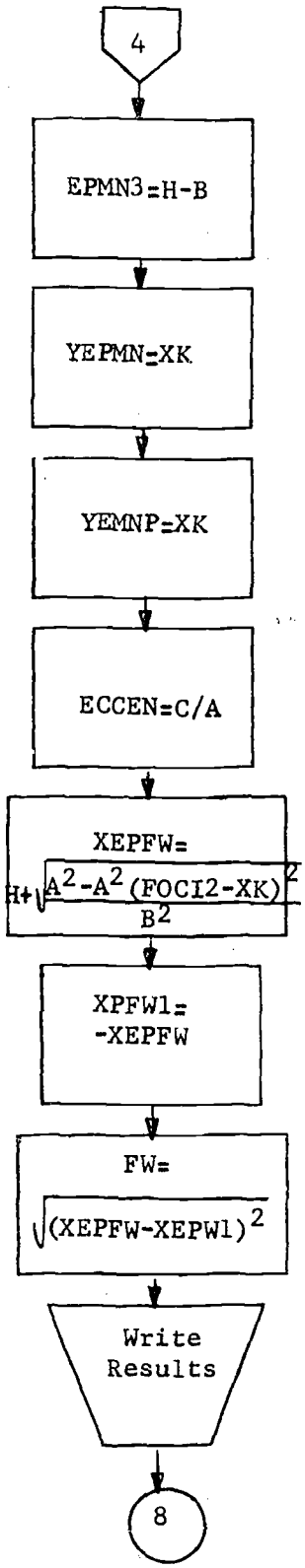
This program will find a list of facts about the ellipse of the form:

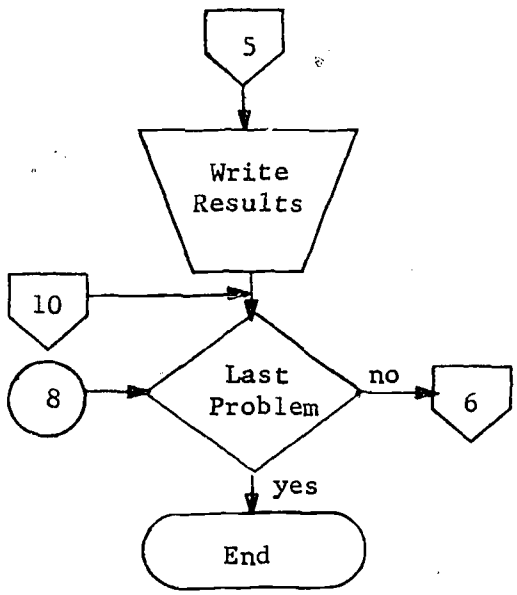
$$\frac{(X-H)^2}{A^2} + \frac{(Y-K)^2}{B^2} = 1$$

INPUT: H,K,A,B

OUTPUT:
- Focal point (x,y)
- Vertex (x,y)
- Length of major axis
- Length of minor axis
- End points of minor axis (x_1,y_1) and (x_2,y_2)
- Eccentricity
- Focal width end points
- Focal width


Start
6
Read H,XK,A,B
1
A
Write the A or B is 0
10
B
9
A-B
C=√(b²-a²)
FOCI2=XK+C
FOCI3=XK-C
XFOC=H
XFOCI=H
VERT=XK+A
VERT3=XK-A
XVER=H
3
Write This is a Circle
1
3
XVERT=H
ZMAJ=2.*A
ZMIN=2.*B
EPMN2=H+B
4
C=√(a²-b²)
FOCI=H+C
FOCIS=H-C
YFOC=XK
VERT=H+A
VERTS=H-A
YVER=XK
YVERT=XK
2
2
XMAJ=2.*A
XMIN=2.*B
EPMN=XK-B
EPMIN=XK+B
EPMX=H
EPMX1=H
ECCEN=C/A
EPFWY=XK+√(b²-b²(FOCI-H)²/A²)
EPFY1=-EPFWY
FCLWD=√(EPFWY-EPWY1)²
5


4
EPMN3=H-B
YEPMN=XK
YEMNP=XK
ECCEN=C/A
XEPFW=
H+√(A²-A²(FOCI2-XK)²/B²)
XPFW1=
-XEPFW
FW=
√(XEPFW-XEPW1)²
Write
Results
8


5
Write
Results
10
8
Last
Problem
no
6
yes
End

IBM FORTRAN Coding Form

ELLIPSE 1 3

FORTRAN STATEMENT

```
C       THIS PROGRAM FINDS TERMS OF AN ELLIPSE
C       (X-H)**2/A**2 + (Y-K)**2/B**2=1.
  600   READ(2,1)H,XK,A,B
    1   FORMAT(F7.3)
        IF(A)5,9,5
    5   IF(B)6,9,6
    9   WRITE(3,2)
    2   FORMAT(2X,'THE A OR B TERM IS 0')
        GO TO 600
    6   IF(A-B)7,8,11
    8   WRITE(3,12)A
   12   FORMAT(2X,'THIS IS A CIRCLE OF RADIUS',2X,F7.3)
        GO TO 600
   11   C=SQRT(A**2-B**2)
        FOCI=H+C
        FOCIS=H-C
        YFOC=XK
        VERT=H+A
        VERTS=H-A
        YVER=XK
        YVERT=XK
        XMAJ=2.*A
        XMIN=2.*B
        EPMN=XK-B
```

IBM FORTRAN Coding Form

ELLIPSE 2 3

FORTRAN STATEMENT

```
        EPMIN=XK+B
        EPMX=H
        EPMX1=H
        ECCEN=C/A
        EPFWY=XK+(SQRT(B**2-B**2*(FOCI-H)**2/A**2))
        EPWY1=-EPFWY
        FCLWD=SQRT((EPFWY-EPWY1)**2)
        WRITE(3,13)FOCI,YFOC,FOCIS,YFOCI,VERT,YVER,VERTS,YVERT
   13   FORMAT('FOCUS AT ','(',F5.1,F5.1,')','AND','(',F5.1,F5.1,')',//,'VE
       1RTEX AT ','(',F5.1,F5.1,')','AND','(',F5.1,F5.1,')')
        WRITE(3,14)XMAJ,XMIN,EPMIN,EPMX,EPMN,EPMX1,ECCEN
   14   FORMAT('LENGTH MAJOR AXIS ',F4.1,/,'LENGTH MINOR AXIS',F4.1,/,'END
       1 PTS. MINOR AXIS','(',F5.1,F5.1,')','AND','(',F5.1,F5.1,')',//,2X,
       2'ECCENTRICITY',F5.1)
        WRITE(3,15)FOCI,EPFWY,FOCI,EPWY1,FOCIS,EPFWY,FOCIS,EPWY1,FCLWD
   15   FORMAT('FOCAL WIDTH END POINTS ARE',/,18X,'(',F5.1,F5.1,')','(',F
       15.1,F5.1,')','AND','(',F5.1,F5.1,')','(',F5.1,F5.1,')',//,'FOCAL
       2 WIDTH IS',F5.1)
        GO TO 600
    7   C=SQRT(B**2-A**2)
        FOCI2=XK+C
        FOCI3=XK-C
        XFOC=H
        XFOCI=H
        VERT2=XK+A
```

IBM FORTRAN Coding Form

ELLIPSE 3 3

FORTRAN STATEMENT

```
      VERT3=XK-A
      XVER=H
      XVERT=H
      ZMAJ=2.*A
      ZMIN=2.*B
      EPMN2=H+B
      EPMN3=H-B
      YEPMW=XK
      YEMNP=XK
      ECCEN=C/A
      XEPFW=H+(SQRT(A**2-A**2*(FOCI2-XK)**2/B**2))
      XPFW1=-XEPFW
      FW=SQRT((XEPFW-XPFW1)**2)
      WRITE(3,13)FOCI2,XFOC,FOCI3,XFOC,VERT2,XVER,VERT3,XVERT
      WRITE(3,14)ZMAJ,ZMIN,EPMN2,YEPMW,EPMN3,YEMNP,ECCEN
      WRITE(3,15)FOCI2,XEPFW,FOCI2,XPFW1,FOCI3,XEPFW,FOCI3,XPFW1,FW
      GO TO 600
9999  CALL EXIT
      END
// XEQ
```

INVERSE OF ANY MATRIX
(GAUSS - JORDAN)

This program finds the inverse of any square matrix using the GAUSS - JORDAN transformation method.

INPUT

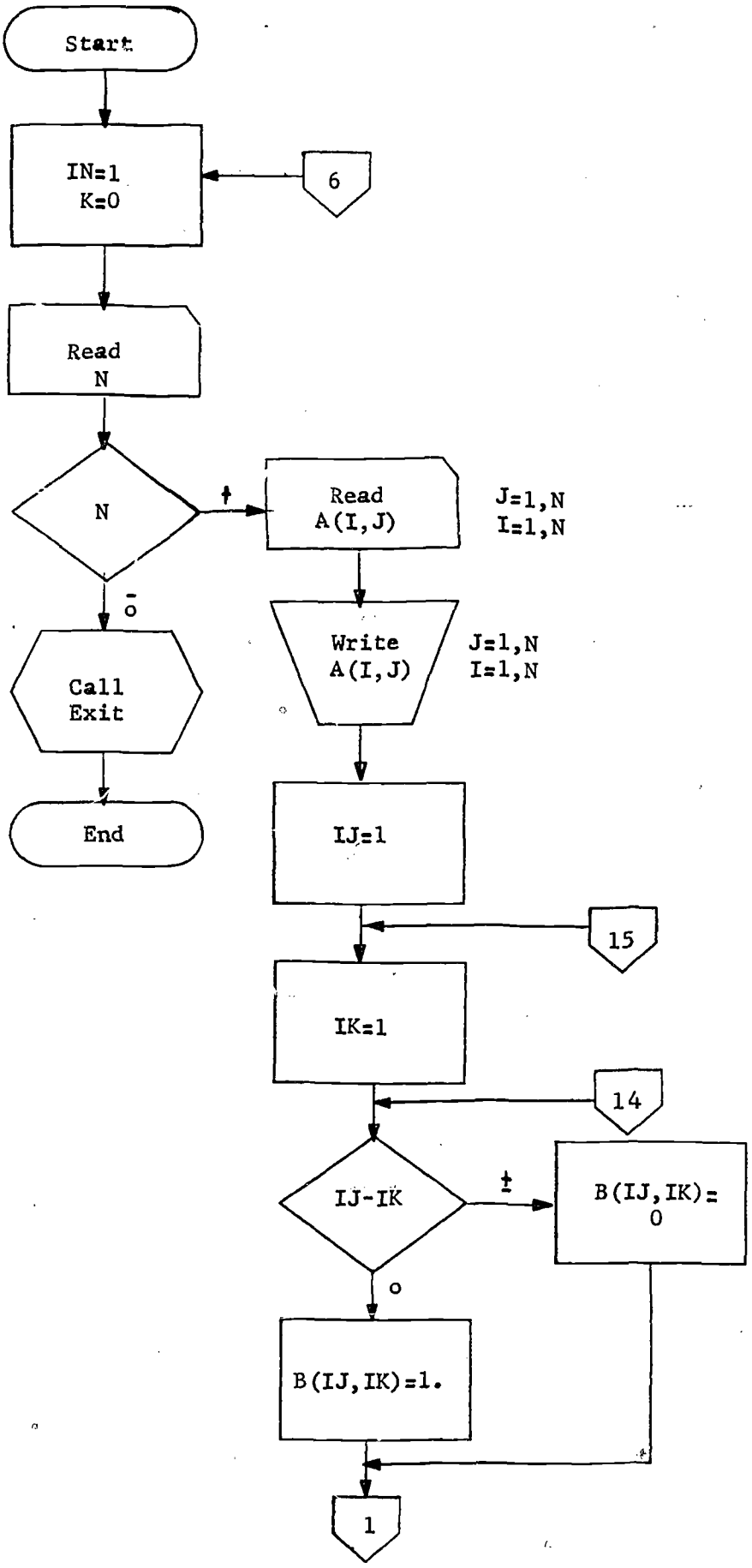
A matrix

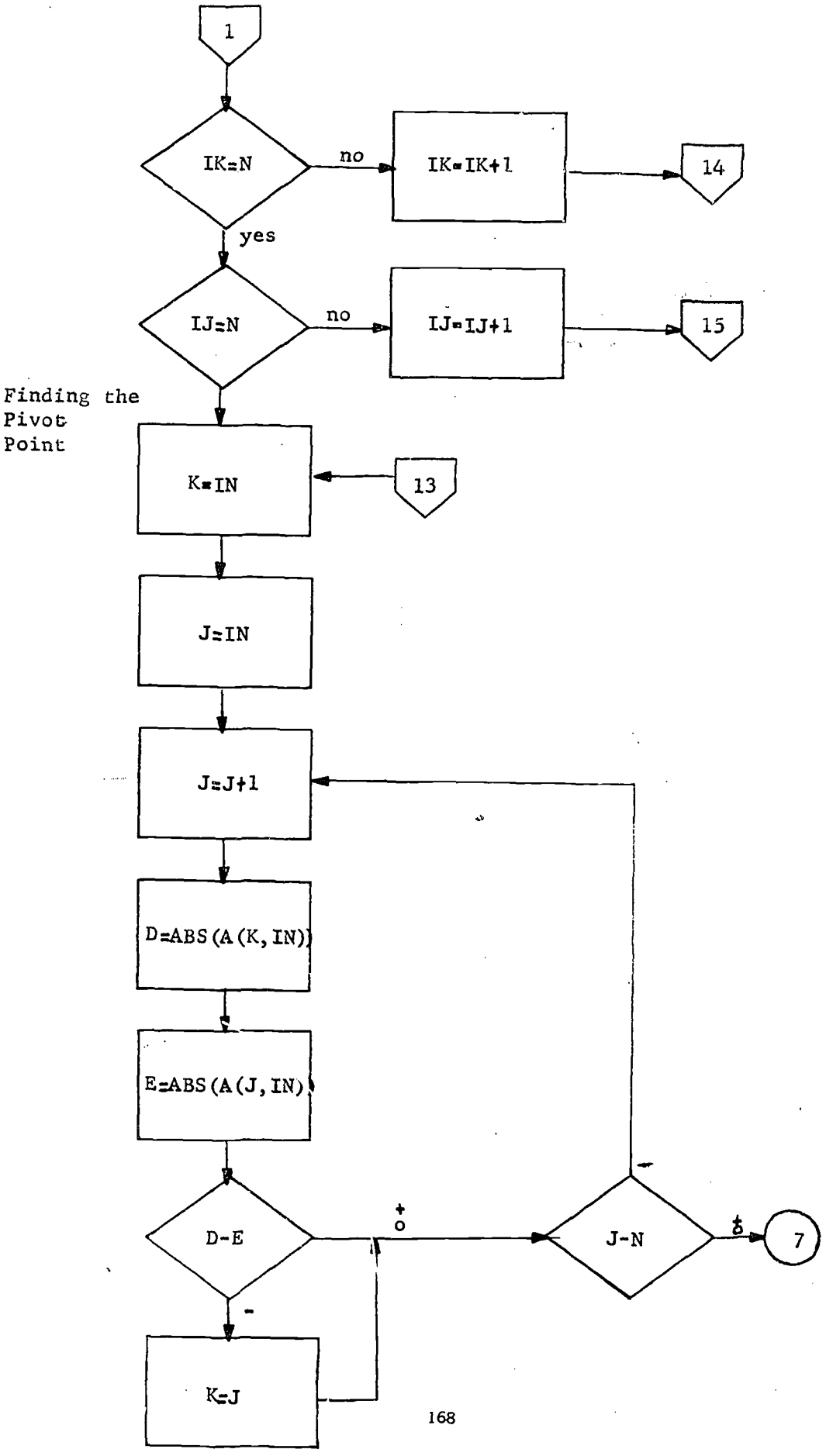
OUTPUT

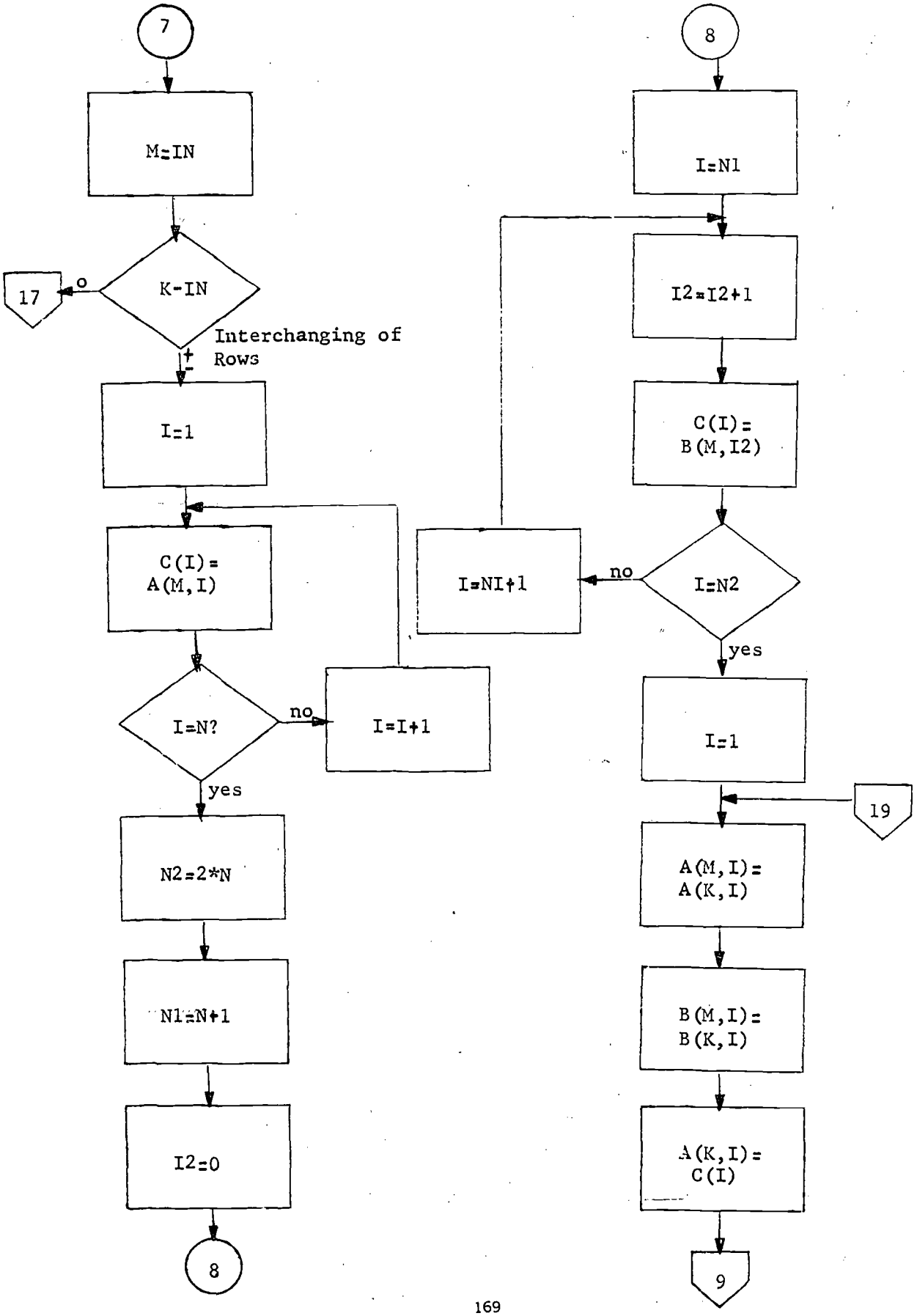
A matrix

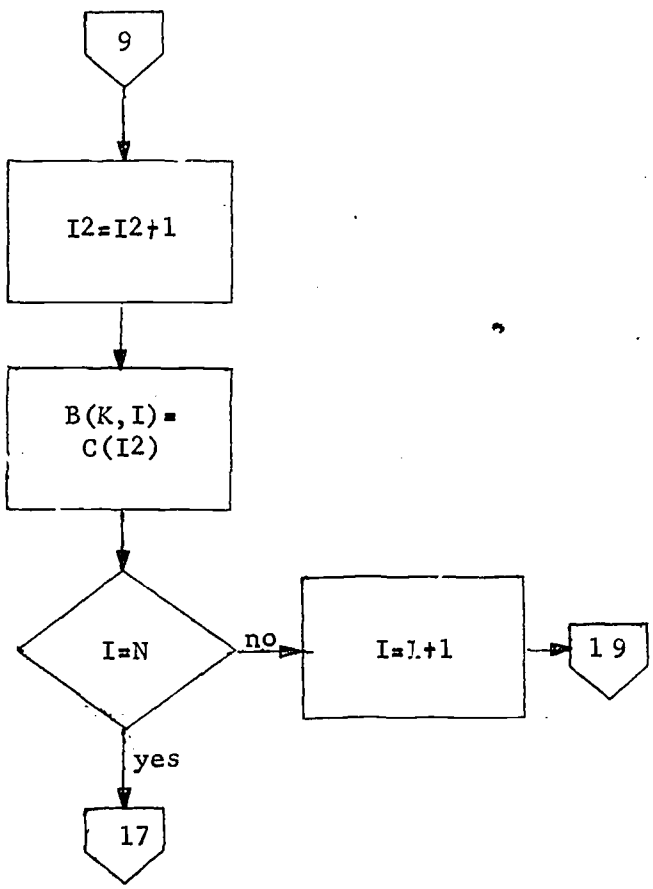
Elements of inverse

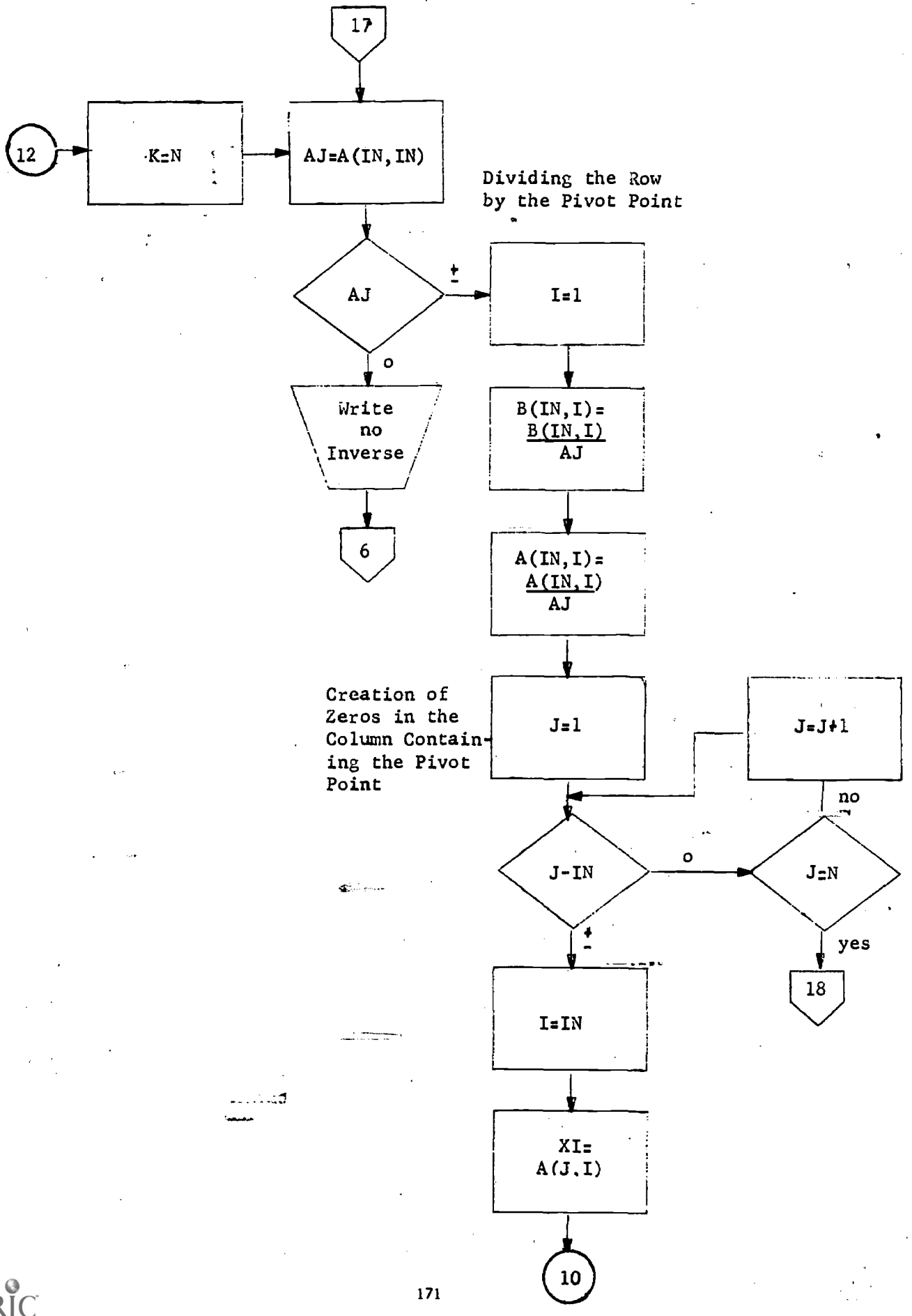
If data switch 0 is up the program will output each transformed matrix relative to A and to B (Identity Matrix)

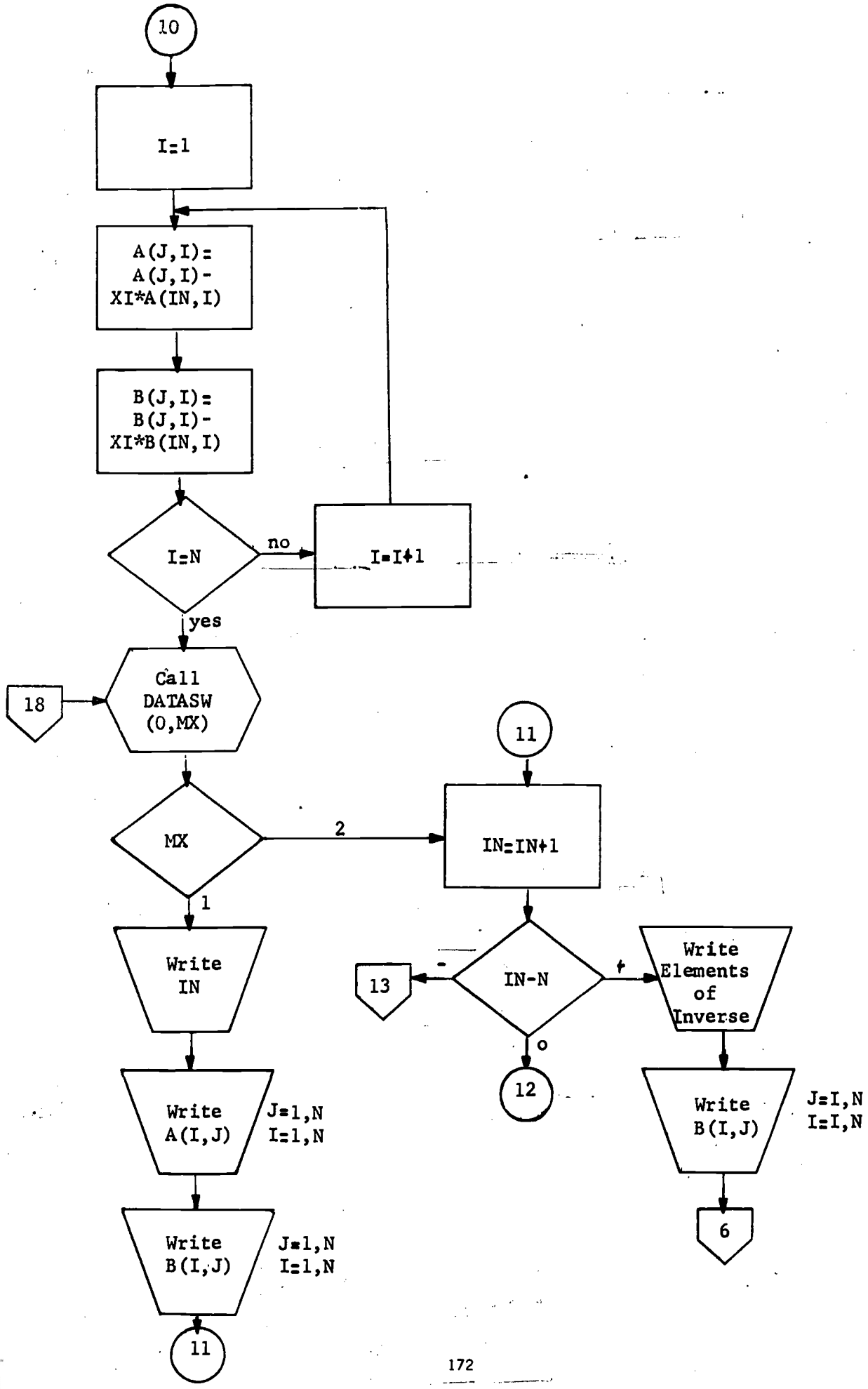

Start
IN=1
K=0
6
Read
N
N
+
Read
A(I,J)
J=1,N
I=1,N
-
0
Write
A(I,J)
J=1,N
I=1,N
Call
Exit
End
IJ=1
15
IK=1
14
IJ-IK
±
B(IJ,IK)=
0
0
B(IJ,IK)=1.
1


1
IK=N
no
IK=IK+1
14
yes
IJ=N
no
IJ=IJ+1
15
Finding the Pivot Point
K=IN
13
J=IN
J=J+1
D=ABS(A(K, IN))
E=ABS(A(J, IN))
D-E
+ 0
J-N
+ 0
7
-
K=J


7
M=IN
K-IN
o
17
+
-
Interchanging of Rows
I=1
C(I)= A(M,I)
I=N?
no
I=I+1
yes
N2=2*N
N1=N+1
I2=0
8
8
I=N1
I2=I2+1
C(I)= B(M,I2)
I=N2
no
I=NI+1
yes
I=1
19
A(M,I)= A(K,I)
B(M,I)= B(K,I)
A(K,I)= C(I)
9


9
I2=I2+1
B(K,I)=
C(I2)
I=N
no
I=I+1
19
yes
17


17
12
K=N
AJ=A(IN,IN)
Dividing the Row by the Pivot Point
AJ
±
I=1
o
Write no Inverse
6
B(IN,I)= B(IN,I) / AJ
A(IN,I)= A(IN,I) / AJ
Creation of Zeros in the Column Containing the Pivot Point
J=1
J=J+1
no
J-IN
o
J=N
yes
±
18
I=IN
XI= A(J,I)
10


10
I=1
A(J,I)=
A(J,I)-
XI*A(IN,I)
B(J,I)=
B(J,I)-
XI*B(IN,I)
I=N
no
I=I+1
yes
18
Call
DATASW
(0,MX)
MX
2
1
11
IN=IN+1
IN-N
-
+
0
13
12
Write
IN
Write
A(I,J)
J=1,N
I=1,N
Write
B(I,J)
J=1,N
I=1,N
11
Write
Elements
of
Inverse
Write
B(I,J)
J=I,N
I=I,N
6

IBM

FORTRAN Coding Form

INVERSE OF ANY MATRIX

1 4

FORTRAN STATEMENT

```
C       FINDS THE INVERSE OF ANY SQUARE MATRIX(GAUSS-JORDAN)
        DIMENSION A(30,30),B(20,20),C(60)
    1   IN=1
        K=0
        READ(2,2)N
    2   FORMAT(I4)
        IF(N)3,1,4
    4   READ(2,5)((A(I,J),J=1,N),I=1,N)
    5   FORMAT(10F8.2)
        WRITE(3,121)
        WRITE(3,20)((A(I,J),J=1,N),I=1,N)
C       CREATION OF IDENTITY MATRIX
        DO 28 IJ=1,N
        DO 26 IK=1,N
        IF(IJ-IK)27,30,27
   30   B(IJ,IK)=1.
        GO TO 26
   27   B(IJ,IK)=0.
   26   CONTINUE
   28   CONTINUE
C       FINDING OF PIVOT POINT
   17   K=IN
        J=IN
   99   J=J+1
```

IBM

FORTRAN Coding Form

INVERSE OF ANY MATRIX

2 4

FORTRAN STATEMENT

```
        D=ABS(A(K,IN))
        E=ABS(A(J,IN))
        IF(D-E)6,8,8
    6   K=J
    8   IF(J-N)99,98,98
   97   K=N
        GO TO 102
   98   M=IN
        IF(K-IN)103,102,103
C       INTERCHANGING OF ROWS
  103   DO 9 I=1,N
    9   C(I)=A(M,I)
        N2=2*N
        N1=N+1
        I2=0
        DO 10 I=N1,N2
        I2=I2+1
   10   C(I)=B(M,I2)
        DO 11 I=1,N
        A(M,I)=A(K,I)
        B(M,I)=B(K,I)
        A(K,I)=C(I)
```

IBM FORTRAN Coding Form

INVERSE OF ANY MATRIX 3 4

FORTRAN STATEMENT

```
      I2=I2+1
   11 B(K,I)=C(I2)
C     DIVIDING THE ROW BY THE PIVOT POINT
  102 AJ=A(IN,IN)
      IF(AJ)96,31,96
   96 DO 12 I=1,N
      B(IN,I)=B(IN,I)/AJ
   12 A(IN,I)=A(IN,I)/AJ
C     CREATION OF ZEROS IN THE COLUMN CONTAIMING THE PIVOT POINT
  101 DO 13 J=1,N
      IF(J-IN)14,13,14
   14 I=IN
      XI=A(J,I)
      DO 15 I=1,N
      A(J,I)=A(J,I)-XI*A(IN,I)
   15 B(J,I)=B(J,I)-XI*B(IN,I)
   13 CONTINUE
      CALL DATSW(0,MX)
      GO TO(122,123),MX
  122 WRITE(3,2)IN
      WRITE(3,121)
  121 FORMAT(' A MATRIX')
      WRITE(3,20)((A(I,J),J=1,N),I=1,N)
      WRITE(3,124)
```

IBM FORTRAN Coding Form

INVERSE OF ANY MATRIX 4 4

FORTRAN STATEMENT

```
  124 FORMAT(' B MATRIX')
      WRITE(3,20)((B(I,J),J=1,N),I=1,N)
  123 IN=IN+1
      IF(IN-N)17,97,18
   31 WRITE(3,32)
   32 FORMAT(' THIS MATRIX HAS NO INVERSE')
      GO TO 1
   18 WRITE(3,19)
   19 FORMAT(' ELEMENTS OF INVERSE')
      WRITE(3,20)((B(I,J),J=1,N),I=1,N)
   20 FORMAT(10F12.3)
      GO TO 1
    3 CALL EXIT
      END
```

PROBLEM SUPPLEMENT

Chemistry Section IV

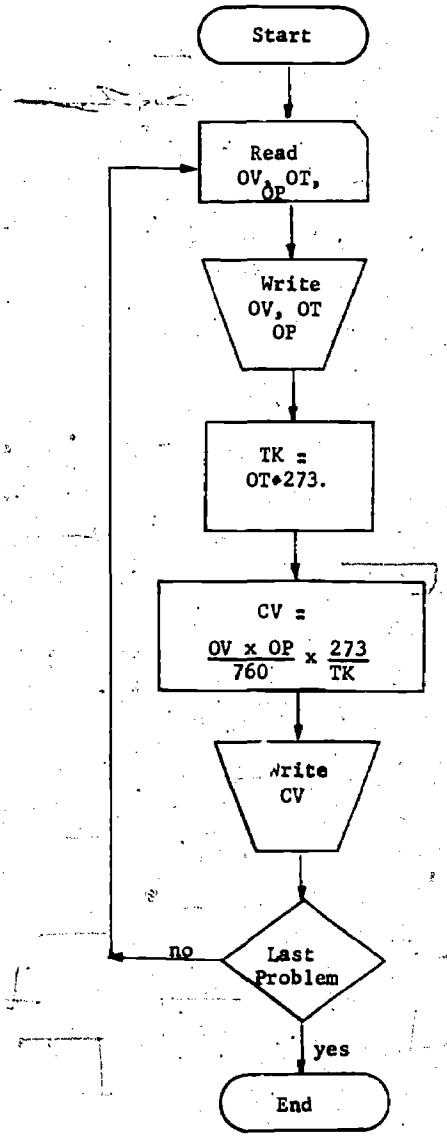
STP

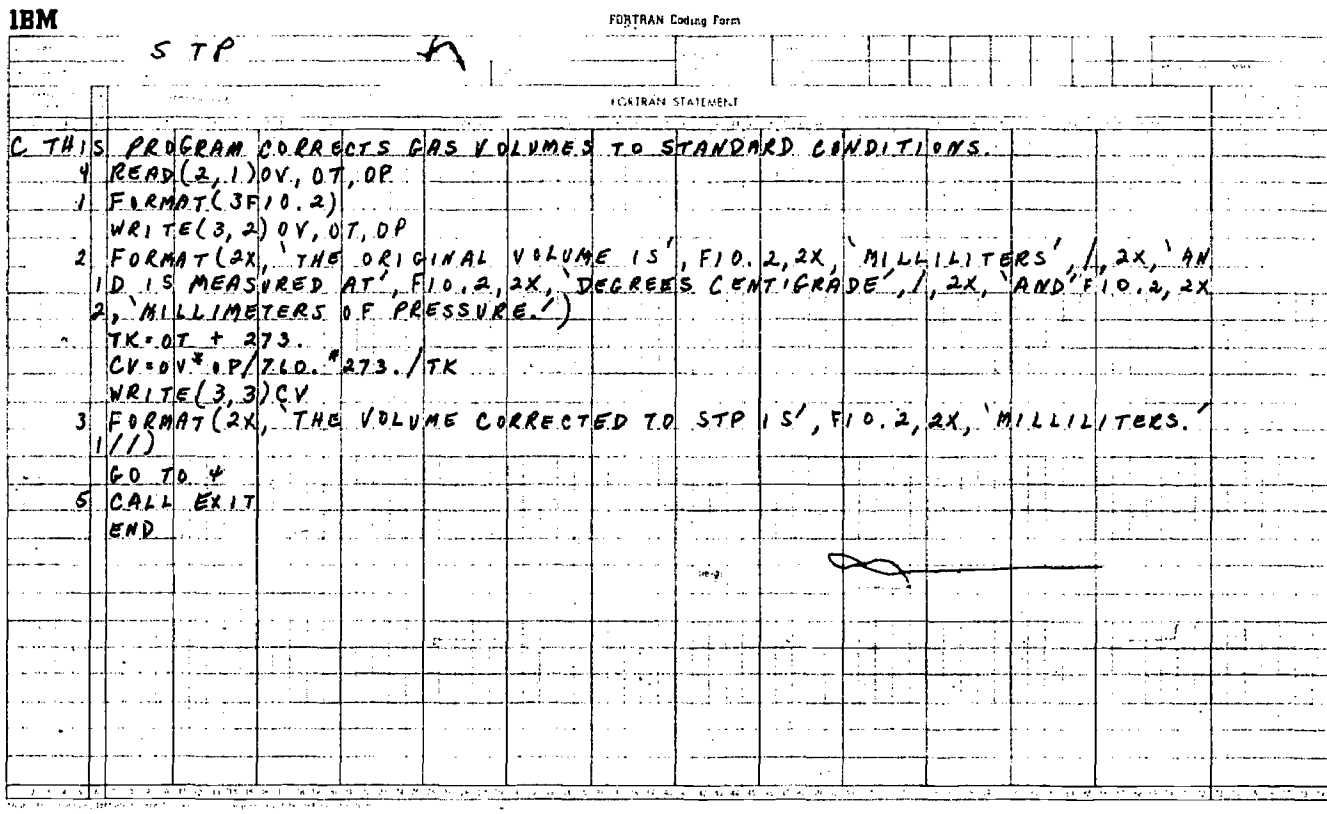
This program takes any known volume of gas in milliliters at any known pressure in millimeters and any known temperature in degrees Centigrade, and corrects this volume to standard conditions.

VARIABLES

INPUT: OV - Original volume.
OT - Original temperature.
OP - Original pressure.

Output: The program writes out the values of OV, OT, and OP and also the corrected volume for each set of values.





```
IBM                                   FORTRAN Coding Form
      STP
                                      FORTRAN STATEMENT
C THIS PROGRAM CORRECTS GAS VOLUMES TO STANDARD CONDITIONS.
    4 READ(2,1)OV,OT,OP
    1 FORMAT(3F10.2)
      WRITE(3,2)OV,OT,OP
    2 FORMAT(2X,'THE ORIGINAL VOLUME IS',F10.2,2X,'MILLILITERS',/,2X,'AN
     1D IS MEASURED AT',F10.2,2X,'DEGREES CENTIGRADE',/,2X,'AND',F10.2,2X
     2,'MILLIMETERS OF PRESSURE.')
      TK=OT + 273.
      CV=OV*OP/760.*273./TK
      WRITE(3,3)CV
    3 FORMAT(2X,'THE VOLUME CORRECTED TO STP IS',F10.2,2X,'MILLILITERS.'
     1//)
      GO TO 4
    5 CALL EXIT
      END
```

C TO F TO K

This program converts Centigrade temperatures to Fahrenheit and also to Kelvin degrees.

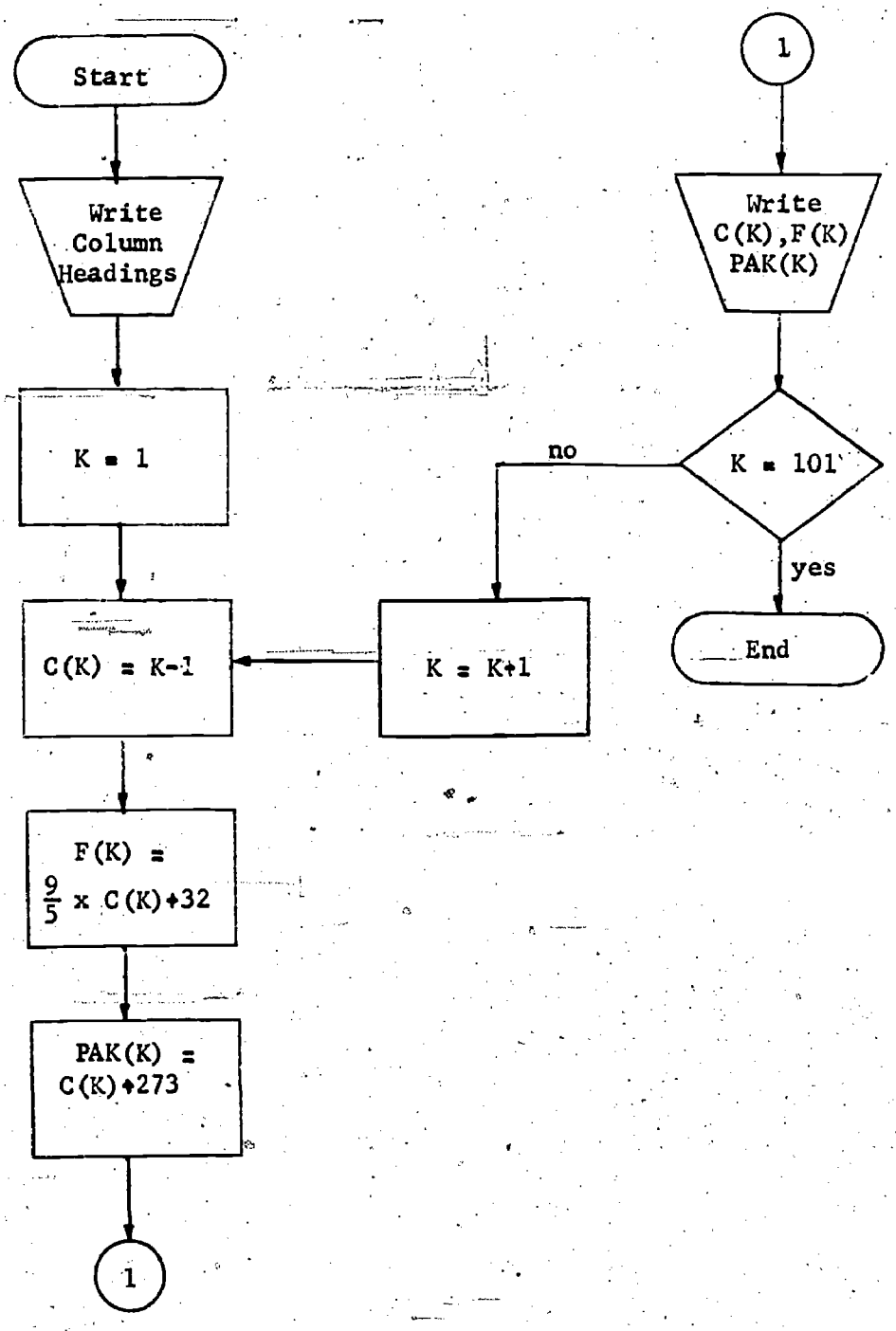
Input: None

Output: All Centigrade temperatures between 1 and 100 degrees with the corresponding Fahrenheit and Kelvin equivalents.

This program could be easily modified to produce Centigrade readings when Fahrenheit is available, or any other combination.

VARIABLES

C - temperature in Centigrade
F - temperature in Fahrenheit
PAK - temperature in Kelvin


Start
Write Column Headings
K = 1
C(K) = K-1
F(K) = 9/5 x C(K)+32
PAK(K) = C(K)+273
1
1
Write C(K),F(K) PAK(K)
K = 101
no
yes
End
K = K+1

IBM FORTRAN Coding Form

C TO F TO K

FORTRAN STATEMENT

```
C THIS PROGRAM WILL CONVERT CENTIGRADE DEGREES TO FAHRENHEIT AND KELVIN
C DEGREES
      DIMENSION C(101), F(101), PAK(101)
      WRITE (3,31)
   31 FORMAT (2X, 'CENTIGRADE', 6X, 'FAHRENHEIT', 6X, 'KELVIN')
      DO 71 K=1, 101
      C(K)=K-1
      F(K)=((9./5.)*C(K)+32.)
      PAK(K)=C(K)+273.
      WRITE (3,51) C(K), F(K), PAK(K)
   51 FORMAT (2X, F7.2, 6X, F8.2, 6X, F9.2)
   71 CONTINUE
   61 CALL EXIT
      END
```

PH

This program converts PH values to the corresponding hydrogen ion concentrations.

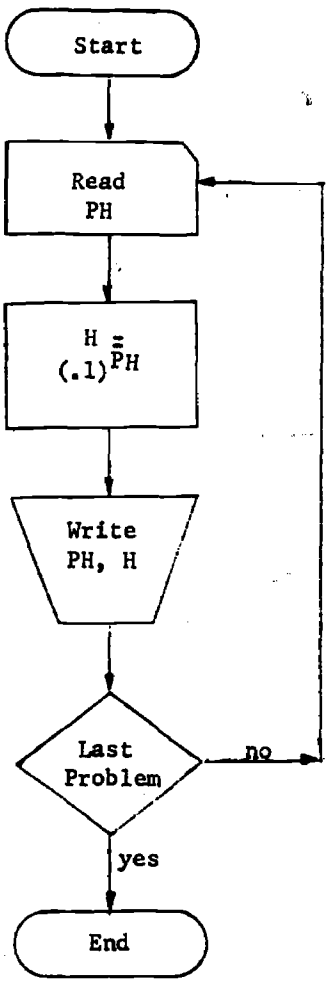
Input: PH values between 1 and 14.

Output: The corresponding hydrogen ion concentrations.

This program could be modified to produce PH values from hydrogen ion concentrations.

VARIABLES

PH - Scale of acidity and basicity
H+ - hydrogen ion concentrations



IBM

FORTRAN Coding Form

PROGRAM pH

PROGRAMMER DATE

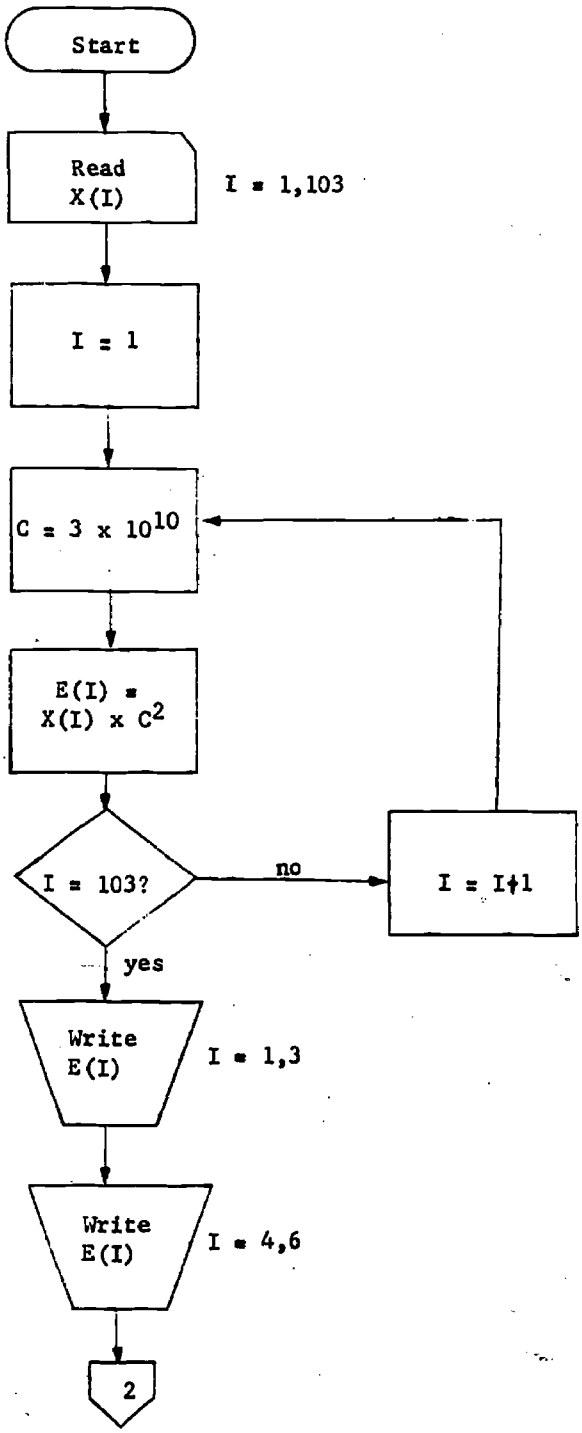
STATEMENT NUMBER	FORTRAN STATEMENT	IDENTIFICATION SEQUENCE
C	THIS PROGRAM CONVERTS PH VALUES TO THE CORRESPONDING HYDROGEN	
C	ION CONCENTRATIONS	
10	READ (2,15)PH	
15	FORMAT (4F10.2)	
	H=0.1**PH	
	WRITE (3,20) PH,H	
20	FORMAT(1H ,'PH=',F5.1,5X,'(H+)=',F10.8)	
	GO TO 10	
26	STOP	
	END	

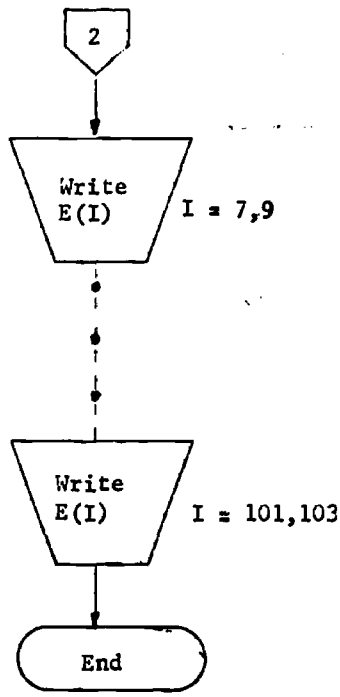
EINSTEIN'S EQUATION

This program will solve the Einstein equation, finding the energy involved when each of the 103 elements is converted into energy.

VARIABLES

X - atomic weight of each element
E - the list of elements
C - speed of light in cm.





IBM

FORTRAN Coding Form

EINSTEIN'S EQUATION

```
C THIS PROGRAM CALCULATES THE VALUE OF E IN THE EINSTEIN EQUATION FOR
C ALL ELEMENTS IN THE PERIODIC TABLE.
      DIMENSION X(150),E(103)
      READ (2,11) (X(I),I=1,103)
   11 FORMAT (3F9.3)
      DO 12 I=1,103
      C=3.0*10.**10
      E(I)=X(I)*C**2
   12 CONTINUE
      WRITE (3,13)(E(I),I=1,3)
   13 FORMAT (2X,'H=',E14.3,2X,'HE=',E14.3,2X,'LI=',E14.3)
      WRITE (3,14)(E(I),I=4,6)
   14 FORMAT (1X,'BE=',E14.3,3X,'B=',E14.3,3X,'C=',E14.3)
      WRITE (3,15) (E(I),I=7,9)
   15 FORMAT (2X,'N=',E14.3,3X,'O=',E14.3,3X,'F=',E14.3/)
      WRITE (3,16) (E(I),I=10,12)
   16 FORMAT (1X,'NE=',E14.3,2X,'NA=',E14.3,2X,'MG=',E14.3)
      WRITE (3,17) (E(I),I=13,15)
   17 FORMAT (1X,'AL=',E14.3,2X,'SI=',E14.3,3X,'P=',E14.3)
      WRITE (3,18) (E(I),I=16,18)
   18 FORMAT (2X,'S=',E14.3,2X,'CL=',E14.3,2X,'AR=',E14.3/)

            (THE PROGRAM CONTINUES IN THE SAME FASHION WITH WRITE
             AND FORMAT STATEMENTS UNTIL ALL 103 ELEMENTS ARE INCLUDED.)
```

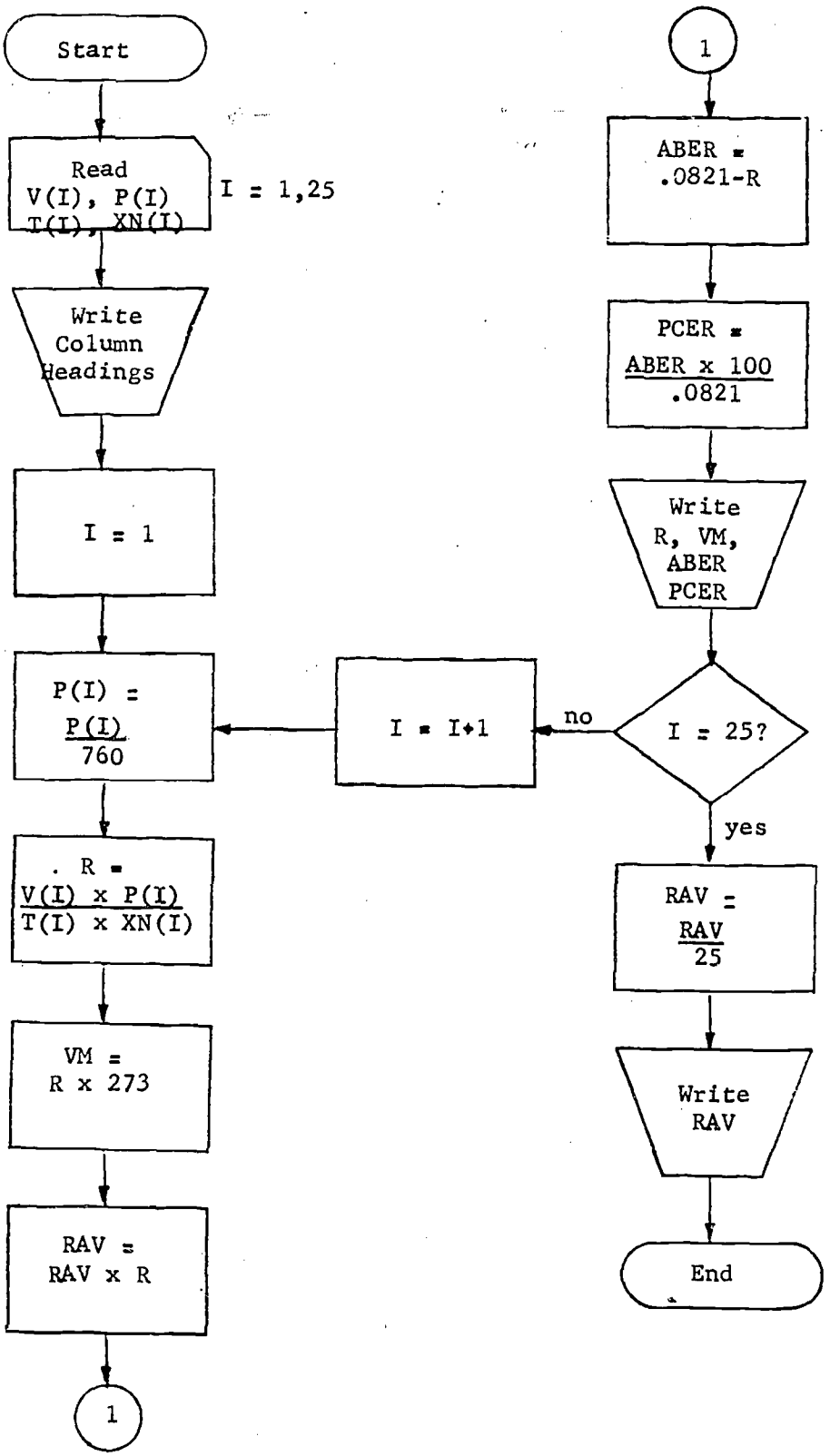
REXP

This program will find the experimental value of the Universal Gas Constant, compare it to the actual value, calculate the average and percent of error for each member of a class of 25 students.

VARIABLES

Input: V(I) - Volume of gas in liters.
P(I) - Pressure of gas in millimeters.
T(I) - Temperature of gas in Kelvin degrees.
XN(I) - Number of moles of gas.

Output: R - Experimental value of gas constant.
VM - Molar volume constant.
ABER - Deviation of R from accepted value.
RAV - Average value of R.
PCER - Percent of error of ABER from the accepted value.


Start
Read V(I), P(I) T(I), XN(I)
I = 1,25
Write Column Headings
I = 1
P(I) = P(I) / 760
. R = V(I) x P(I) / T(I) x XN(I)
VM = R x 273
RAV = RAV x R
1
1
ABER = .0821-R
PCER = ABER x 100 / .0821
Write R, VM, ABER PCER
I = 25?
no
I = I+1
yes
RAV = RAV / 25
Write RAV
End

IBM FORTRAN Coding Form

REXP

```
C THIS PROGRAM FINDS THE EXPERIMENTAL VALUE OF THE UNIVERSAL GAS CONSTANT
C T AND COMPARES IT TO THE ACTUAL VALUE AND CALCULATES THE VALUE AND
C THE PERCENT OF ERROR FOR A CLASS OF AS STUDENTS.
      DIMENSION V(25), P(25), T(25), XN(25)
      RAV=0.
      READ (2,10) (V(I), P(I), T(I), XN(I), I=1,25)
   10 FORMAT (4F10.3)
      WRITE (3,12)
   12 FORMAT (9X, 'REXP', 6X, 'MOL VOL', 6X, 'RACT=REXP', 6X, 'PCER ERROR')
      DO 15 I=1,25
      P(I)=P(I)/760.
      R=(V(I)*P(I))/T(I)*XN(I))
      VM=R*273.
      RAV=RAV+R
      ABER=.0821-R
      PCER=ABER*100./.0821
   20 WRITE (3,21)R, VM, ABER, PCER
   21 FORMAT (4X, F7.4, 6X, F6.2, 6X, F7.4, 6X, F6.2)
   15 CONTINUE
      RAV=RAV/25.
      WRITE (3,79)RAV
   79 FORMAT (F7.4)
      CALL EXIT
      END
```

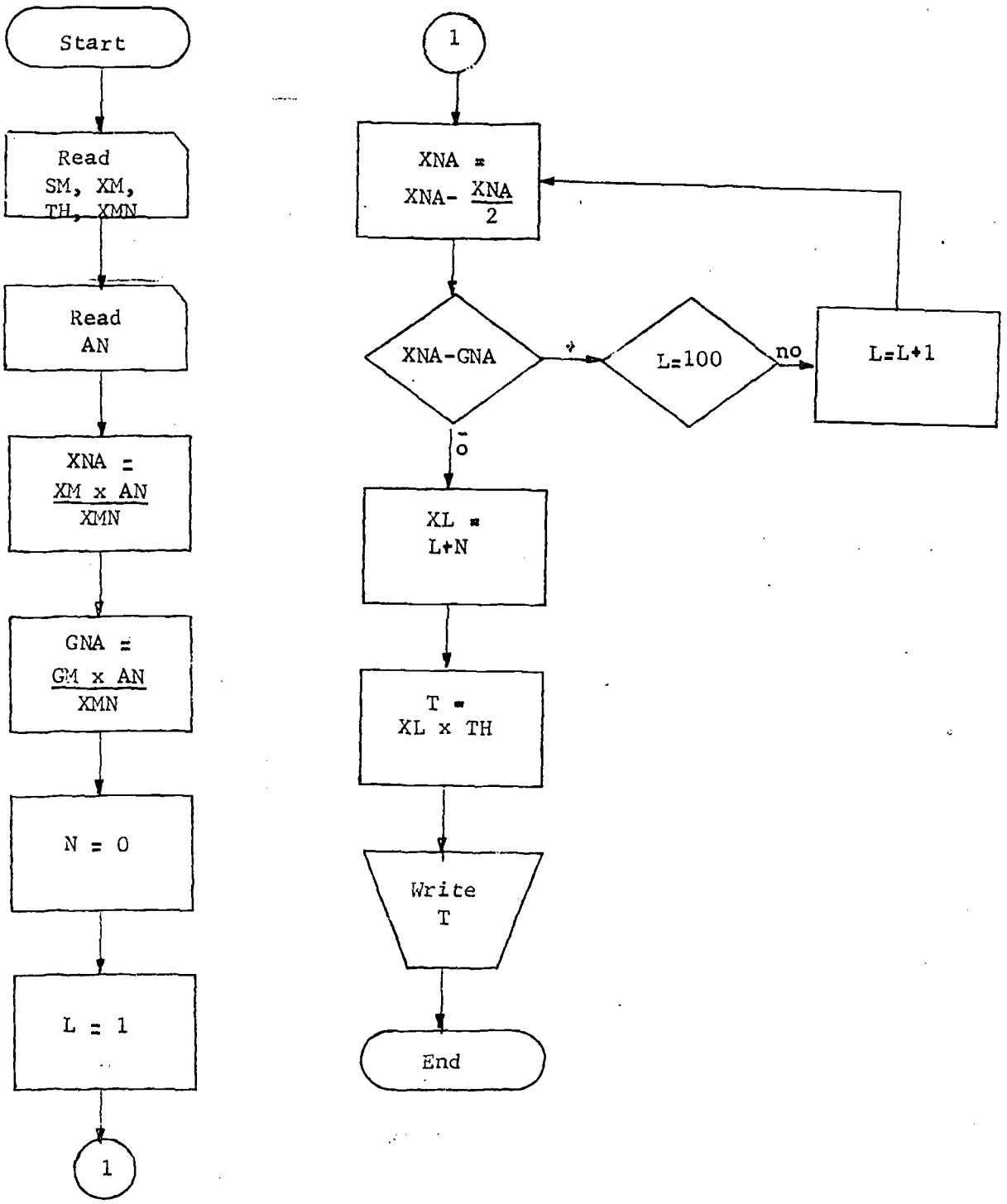
HALF LIFE

This program will determine how long it will take a radioactive sample of known mass to decay to some desired mass, or if the desired mass is 0, to decay to one atom.

VARIABLES

Input: XM - Mass of sample in grams.
GM - Mass after reduction in grams.
AN - Avogadro's Number - 6.023×10^{23} ,(Note) AN is in E Format - .6023 E 24
TH - The halflife of the particular isotope.
XMN - The mass number.

Output: The time required in years (or in same units as TH).


Start
Read SM, XM, TH, XMN
Read AN
XNA = XM x AN / XMN
GNA = GM x AN / XMN
N = 0
L = 1
1
1
XNA = XNA- XNA/2
XNA-GNA
+
L=100
no
L=L+1
-
o
XL = L+N
T = XL x TH
Write T
End

IBM FORTRAN Coding Form

HALF LIFE

FORTRAN STATEMENT

```
C THIS PROGRAM WILL DETERMINE HOW LONG IS REQUIRED FOR A SAMPLE TO DECAY
C TO A PARTICULAR WEIGHT.
      READ(2,1) GM,XM,TH,XMN
    1 FORMAT(4F12.4)
      READ(2,2) AN
    2 FORMAT(E20.5)
      XNA=XM*AN/X
      GNA=GM*AN/XM
      N=0
      DO 3 L=1,1000
      XNA=XNA-.5*XNA
      IF(XNA-GNA)5,5,3
    5 XL=L+N
      T=XL*TH
      WRITE(3,6)T
    6 FORMAT(2X,'THE TOTAL TIME IS',F14.3,2X,'YEARS.')
      GO TO 10
    3 CONTINUE
   10 CALL EXIT
      END
```

GAS ANALYSIS

Problem: A mixture of hydrogen, carbon monoxide, methane and nitrogen has a volume of 20.0 milliliters. To this is added 25.0 mls. of oxygen and the mixture is ignited. The resulting cooled gas is found to have a volume of 23.9 mls. and to contain 6.2 mls. of carbon dioxide and 12.1 mls. of residual oxygen. This program calculates the percent by volume composition of the original gas by the use of matrix algebra.

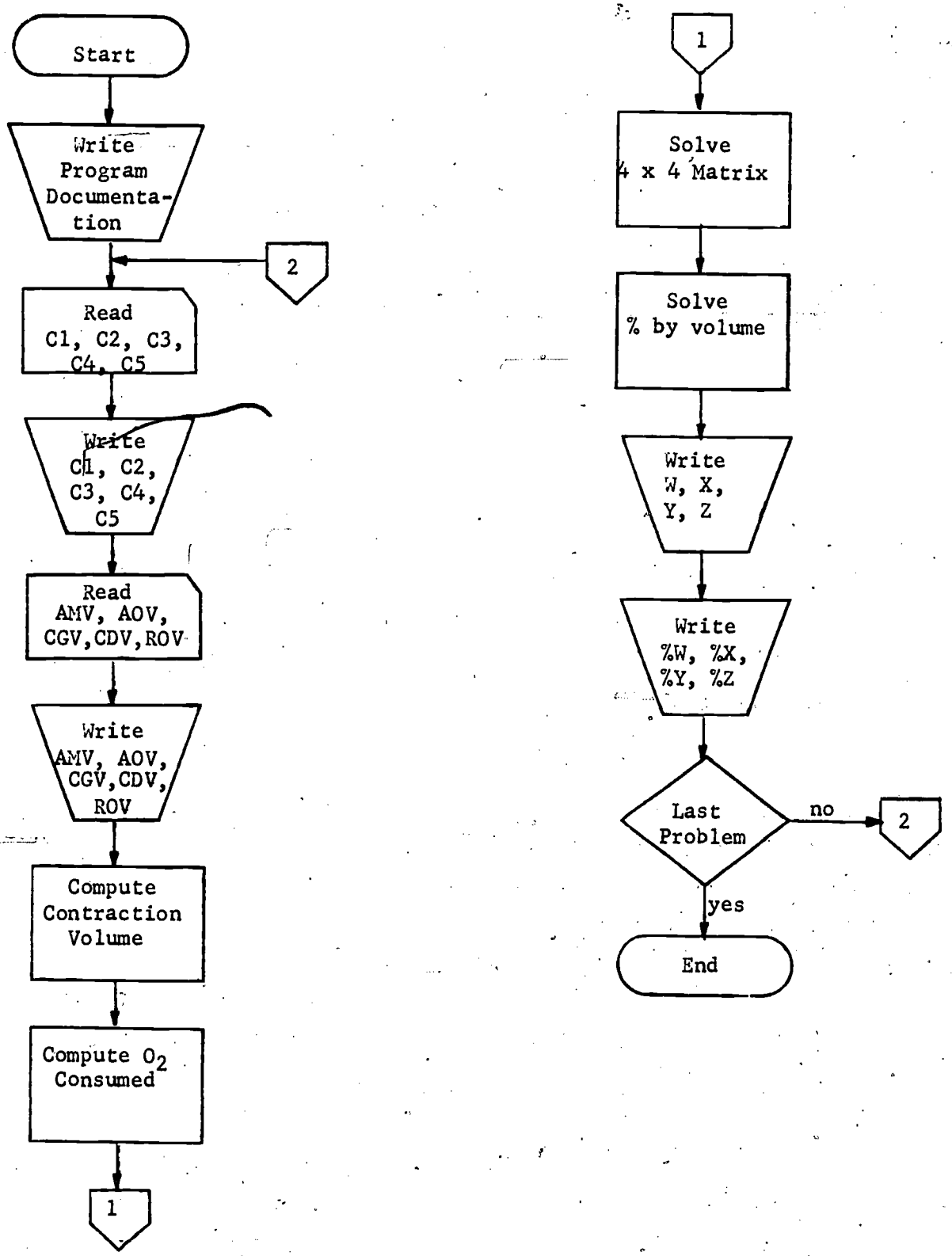
VARIABLES

Input:

C1 = 0.0	AMV - mixture volume
C2 - 0.5	AOV - oxygen volume
C3 - 1.0	CGV - cooled gas volume
C4 - 1.5	CDV - carbon dioxide volume
C5 - 2.0	ROV - residual oxygen volume

Output:

W - volume of hydrogen
X - volume of carbon monoxide
Y - volume of methane
Z - volume of nitrogen


Start
Write Program Documenta-tion
2
Read C1, C2, C3, C4, C5
Write C1, C2, C3, C4, C5
Read AMV, AOV, CGV, CDV, ROV
Write AMV, AOV, CGV, CDV, ROV
Compute Contraction Volume
Compute O_2 Consumed
1
1
Solve 4 x 4 Matrix
Solve % by volume
Write W, X, Y, Z
Write %W, %X, %Y, %Z
Last Problem
no
2
yes
End

IBM FORTRAN Coding Form

GAS ANALYSIS 1 5

```
C THIS PROGRAM FINDS A PERCENT-BY-VOLUME COMPOSITION OF A GAS MIXTURE
C USING MATRIX ALGEBRA.
      DIMENSION A(4,5)
   10 READ (2,100) C1,C2,C3,C4,C5
  100 FORMAT (5F14.8)
      A(1,1) = C3
      A(1,2) = C3
      A(1,3) = C3
      A(1,4) = C3
      A(3,3) = C3
      A(3,2) = C3
      A(2,4) = C1
      A(3,1) = C1
      A(3,4) = C1
      A(4,4) = C1
      A(2,2) = C2
      A(4,1) = C2
      A(4,2) = C2
      A(2,1) = C4
      A(2,3) = C5
      A(4,3) = C5
      WRITE (3,110) C1,C2,C3,C4,C5
  110 FORMAT (5F14.8)
    1 READ (2,120) AMV,AOV,CGV,CDV,ROV
```

IBM FORTRAN Coding Form

GAS ANALYSIS (CONT.) PAGE 2 5

```
  120 FORMAT (5F14.8)
      A(1,5) = AMV
      WRITE (3,130) AMV,AOV,CGV,CDV,ROV
  130 FORMAT (5F14.8)
      A(2,5) = AOV+AMV-CGV
      A(4,5) = AOV-ROV
      A(3,5) = CDV
      N = 0
    7 IF(A(1,1))2,3,2
    3 N = N+1
      IF(N-3)5,5,6
    5 DO 4 I = 2,4
      DO 4 J = 1,6
      TS1 = A(I,J)
      A(I,J) = A(I-1,J)
    4 A(I-1,J) = TS1
      GO TO 7
    6 WRITE (3,101)
  101 FORMAT (2X,'NO SOLUTION')
      GO TO 10
    2 TS1 = A(1,1)
      DO 9 I = 1,5
    8 A(1,I) = A(1,I)/TS1
      DO 9 J = 2,4
```

IBM FORTRAN Coding Form

GAS ANALYSIS (CONT.) 3 5

FORTRAN STATEMENT

```
      TS1=A(J,1)
      DO 9 I=1,5
    9 A(J,I)=A(J,I)-TS1*A(1,I)
      N=0
   15 IF(A(2,2))11,12,11
   12 N=N+1
      IF(N-2)13,13,6
   13 DO 14 I=3,4
      DO 14 J=2,5
      TS1=A(I,J)
      A(I,J)=A(I-1,J)
   14 A(I-1,J)=TS1
      GO TO 15
   11 TS1=A(2,2)
      DO 16 I=2,5
   16 A(2,I)=A(2,I)/TS1
      DO 17 J=3,4
      TS1=A(J,2)
      DO 17 I=2,5
   17 A(J,I)=A(J,I)-TS1*A(2,I)
      TS1=A(2,2)
      DO 18 I=2,5
   18 A(1,I)=A(1,I)-TS1*A(2,I)
      N=0
```

IBM FORTRAN Coding Form

GAS ANALYSIS (CONT.) 4 5

FORTRAN STATEMENT

```
   23 IF(A(3,3))19,20,19
   20 N=N+1
      IF(N-1)21,21,6
   21 DO 22 I=3,5
      TS1=A(4,I)
      A(4,I)=A(3,I)
   22 A(3,I)=A(4,I)
      GO TO 23
   19 TS1=A(3,3)
      DO 28 I=3,5
   28 A(3,I)=A(3,I)/TS1
      DO 24 I=3,5
   24 A(4,I)=A(4,I)-TS1*A(3,I)
      DO 25 J=1,2
      TS1=A(J,3)
      DO 25 I=3,5
   25 A(J,I)=A(J,I)-TS1*A(3,I)
      IF(A(4,4))29,27,29
   27 IF(A(4,5))6,26,6
   29 TS1=A(4,4)
      DO 30 I=4,5
   30 A(4,I)=A(4,I)/TS1
      DO 31 J=1,3
      TS1=A(J,4)
```

IBM FORTRAN Coding Form

GAS ANALYSIS (CONC.) 5 5

FORTRAN STATEMENT

```
      DO 31 I=4,5
   31 A(J,I)=A(J,I)-TS1*A(4,I)
   24 WRITE (3,102)A(1,5),A(2,5),A(3,5),A(4,5)
  102 FORMAT (2X,'ML-H2=',F14.8,4X,'ML-CO=',F14.8,4X,'ML-CH4=',E14.8,4X,
     1'ML-N2=',F14.8)
      W=A(1,5)/AMV*100.
      X=A(2,5)/AMV*100.
      Y=A(3,5)/AMV*100.
      Z=A(4,5)/AMV*100.
      WRITE(3,103)
  103 FORMAT(2X,'PERCENT BY VOLUME')
      WRITE(3,150)W,X,Y,Z
  150 FORMAT(2X,'PCT-H2=',F14.8,4X,'PCT-CO=',F14.8,4X,'PCT-CH4=',F14.8,
     14X,'PCT-N2=',F14.8)
      CALL EXIT
      END
```

PROBLEM SUPPLEMENT

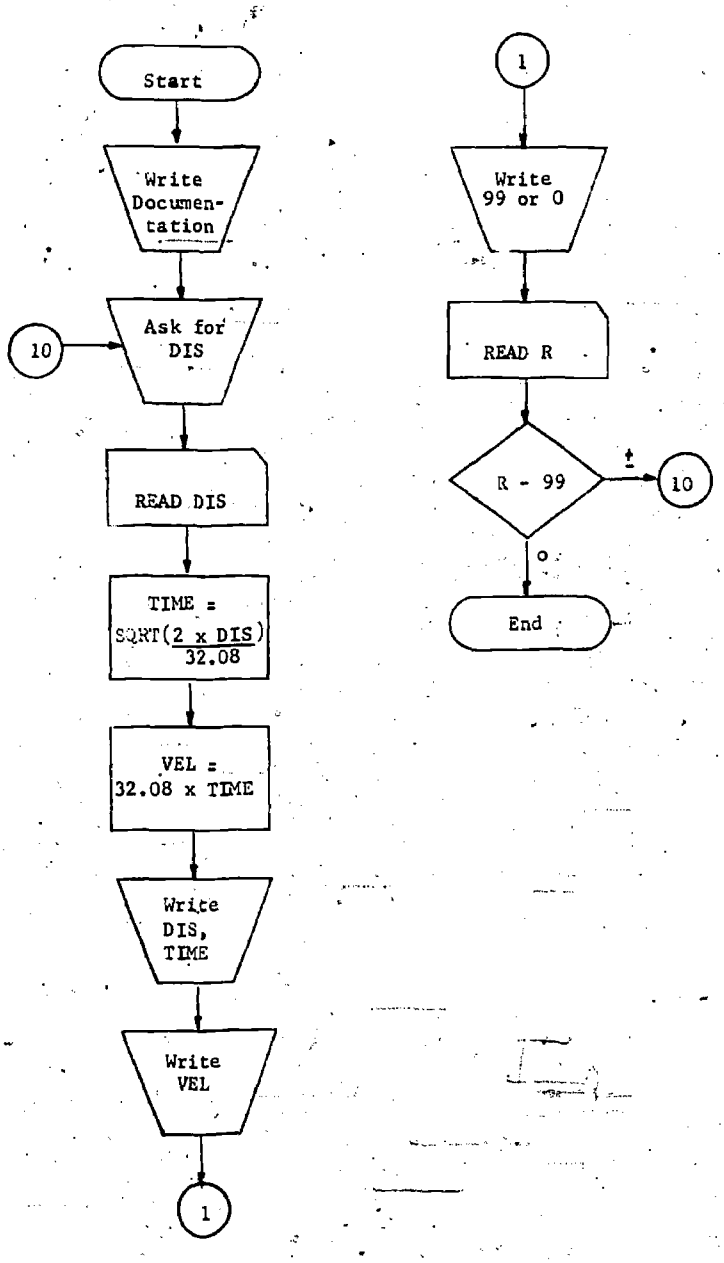
Physics Section V

FREE FALL

This program will consider an object falling under the influence of gravity from any height and compute the time it will take to strike the ground. It will also compute the final velocity of the object. All other forces, including air resistance are ignored.

VARIABLES

DIS - the height from which the object falls
TIME - the time it will take to fall
VEL - the velocity of the object as it strikes the ground.



IBM FORTRAN Coding Form

FREE FALL

FORTRAN STATEMENT

```
      WRITE(1,25)
 2    FORMAT(2X,'THIS PROGRAM WILL FIND THE TIME AND FINAL VELOCITY FOR
     1ANY FREE FALLING OBJECT.',//)
10    WRITE(1,20)
20    FORMAT(2X,'ENTER THE HEIGHT FROM WHICH THE OBJECT FALLS AS A REAL
     1NUMBER.',//)
      READ(6,1)DIS
 1    FORMAT(F10.3)
      TIME=SQRT(2.*DIS/32.08)
      VEL=32.08*TIME
      WRITE(1,22)DIS,TIME
22    FORMAT(2X,'THE TIME TO FALL',F10.2,2X,'FEET IS',F9.2,2X,'SECONDS.'
     1,//)
      WRITE(1,87)VEL
87    FORMAT(2X,'THE FINAL VELOCITY IS',F12.4,2X,'FEET PER SECOND.',///)
      WRITE(1,101)
101   FORMAT(2X,'IF YOU WISH TO MAKE ANOTHER COMPUTATION, ENTER THE NUMB
     1ER 99. IF YOU DO NOT WISH TO CONTINUE ENTER 0.',//)
      READ(6,250)R
250   FORMAT(F6.2)
      IF(R-99.)201,10,201
201   CALL EXIT
      END
```

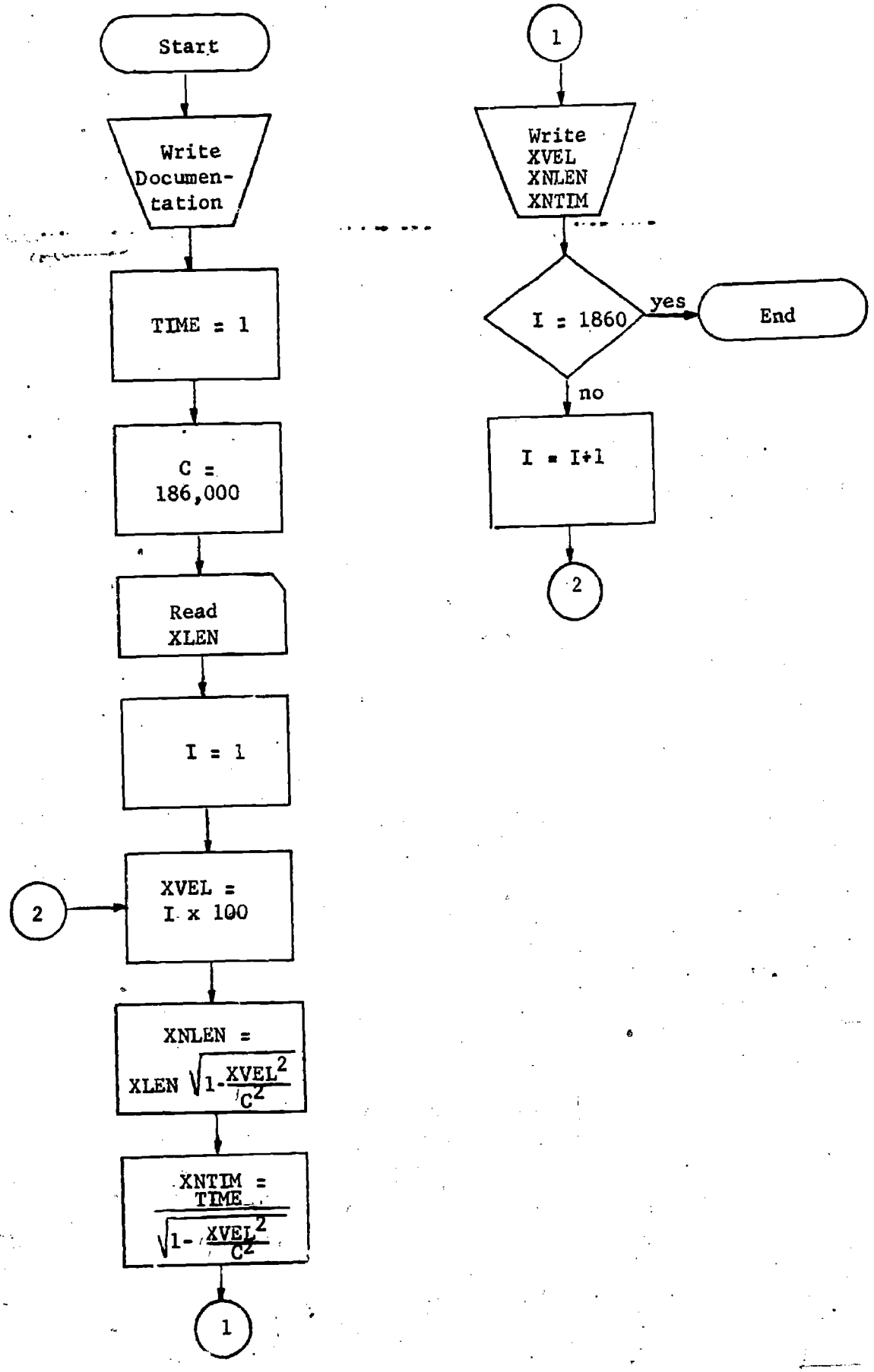
EINSTEIN

This program will give the theoretical increase in length an object undergoes as it approaches the speed of light. It also accounts for the theoretical increase in time.

Great facility can be obtained through the alteration of the limits of the DO-loop governing the velocity of the object.

VARIABLES

XLEN - the length of the object at rest
XVEL - the velocity of the object
XNTIM - the relative length of time
XNLEN - the relative length of the object


Start
Write Documen-tation
TIME = 1
C = 186,000
Read XLEN
I = 1
2
XVEL = I x 100
XNLEN = XLEN $\sqrt{1-\frac{XVEL^2}{C^2}}$
XNTIM = $\frac{TIME}{\sqrt{1-\frac{XVEL^2}{C^2}}}$
1
1
Write XVEL XNLEN XNTIM
I = 1860
yes
End
no
I = I+1
2

IBM FORTRAN Coding Form

EINSTEIN

FORTRAN STATEMENT

```
C       THEORY OF RELATIVITY
        WRITE(3,1)
    1   FORMAT(2X,'THIS PROGRAM WILL COMPUTE THE RELATIVE LENGTH OF AN OBJ
       1ECT AS IT APPROACHES THE SPEED',//,' OF LIGHT; IT ALSO COMPUTES THE
       2 RELATIVE LENGTH OF TIME FOR THE RESPECTIVE VELOCITY.',///)
        TIME=1.
        C=186000.
        READ(2,2)XLEN
    2   FORMAT(F7.2)
        WRITE(3,3)XLEN
    3   FORMAT(2X,'THE LENGTH OF THE OBJECT AT REST IS',4X,F9.3,8X,'THE TI
       1ME AT REST IS 1.00000 SECOND.',///)
        DO 20 I=1,1860
        XVEL=I*100
        XNLEN=XLEN*SQRT(1.-(XVEL**2/C**2))
        XNTIM=TIME*1./SQRT(1.-(XVEL**2/C**2))
        WRITE(3,4)XVEL,XNLEN,XNTIM
    4   FORMAT(5X,'VELOCITY=',F10.0,20X,'LENGTH=',F12.7,20X,'TIME=',F12.7,
       1/1X,'SECONDS')
   20   CONTINUE
        CALL EXIT
        END
```

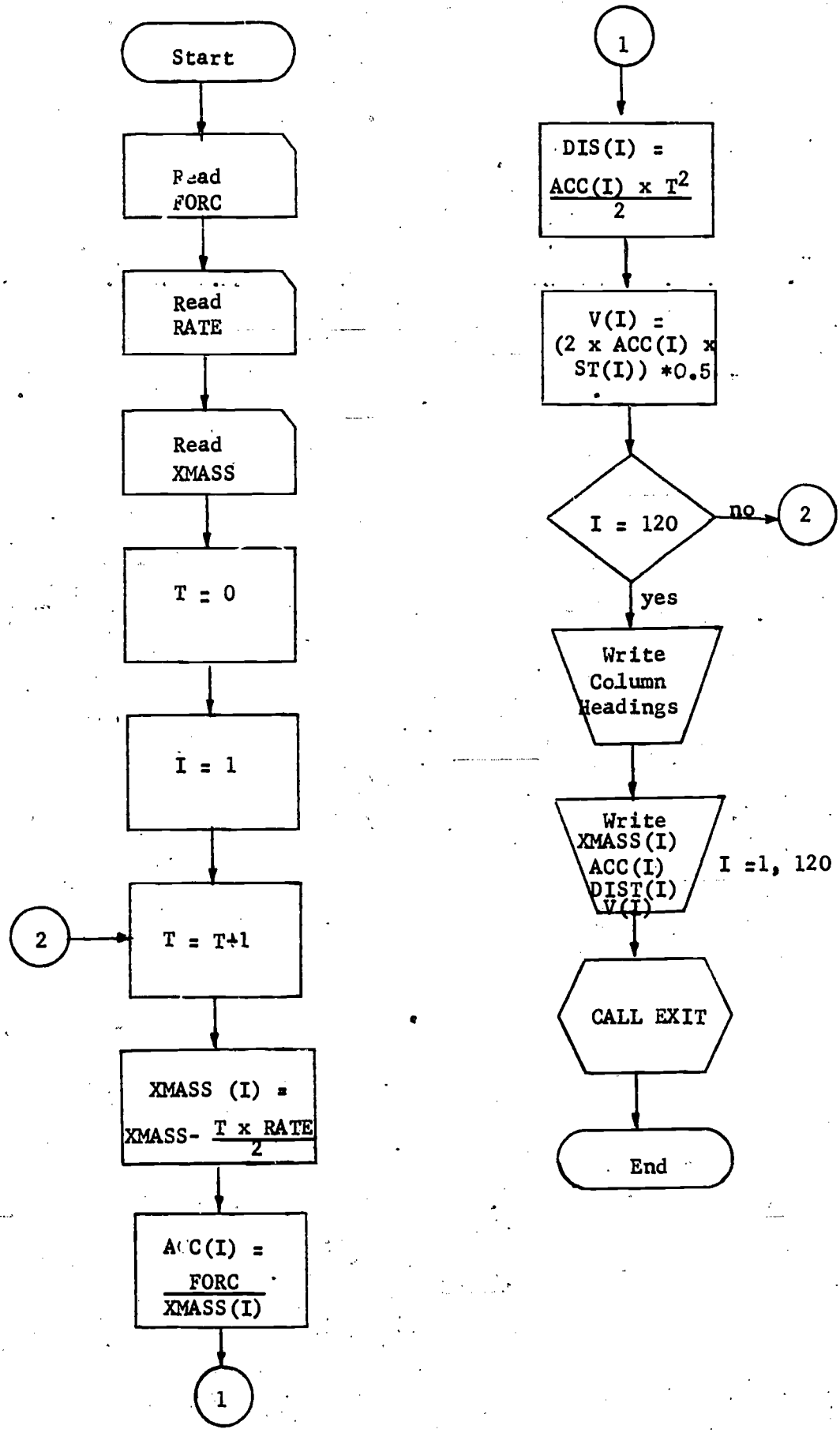
ROCKET

In a rocket, fuel is continuously being burned at a tremendous rate. Consequently the mass of the rocket is constantly decreasing while the force of the rocket engine remains constant. This situation results in a continual increase in acceleration as predicted by the formula F = ma.

This program will compute the mass, acceleration, distance traveled, and velocity of the rocket at one second intervals for two minutes.

VARIABLES

FORC - force produced by the rocket engine in lbs. or nt.
RATE - rate of fuel consumption in mass/sec.
XMASS - mass of the rocket and fuel.
ACC - acceleration.
DIS - distance traveled.
V - velocity of the rocket.


Start
Read FORC
Read RATE
Read XMASS
T = 0
I = 1
2
T = T+1
XMASS (I) = XMASS- T x RATE / 2
ACC(I) = FORC / XMASS(I)
1
1
DIS(I) = ACC(I) x T2 / 2
V(I) = (2 x ACC(I) x ST(I)) *0.5
I = 120
no
2
yes
Write Column Headings
Write XMASS(I) ACC(I) DIST(I) V(I)
I =1, 120
CALL EXIT
End

IBM

ROCKET

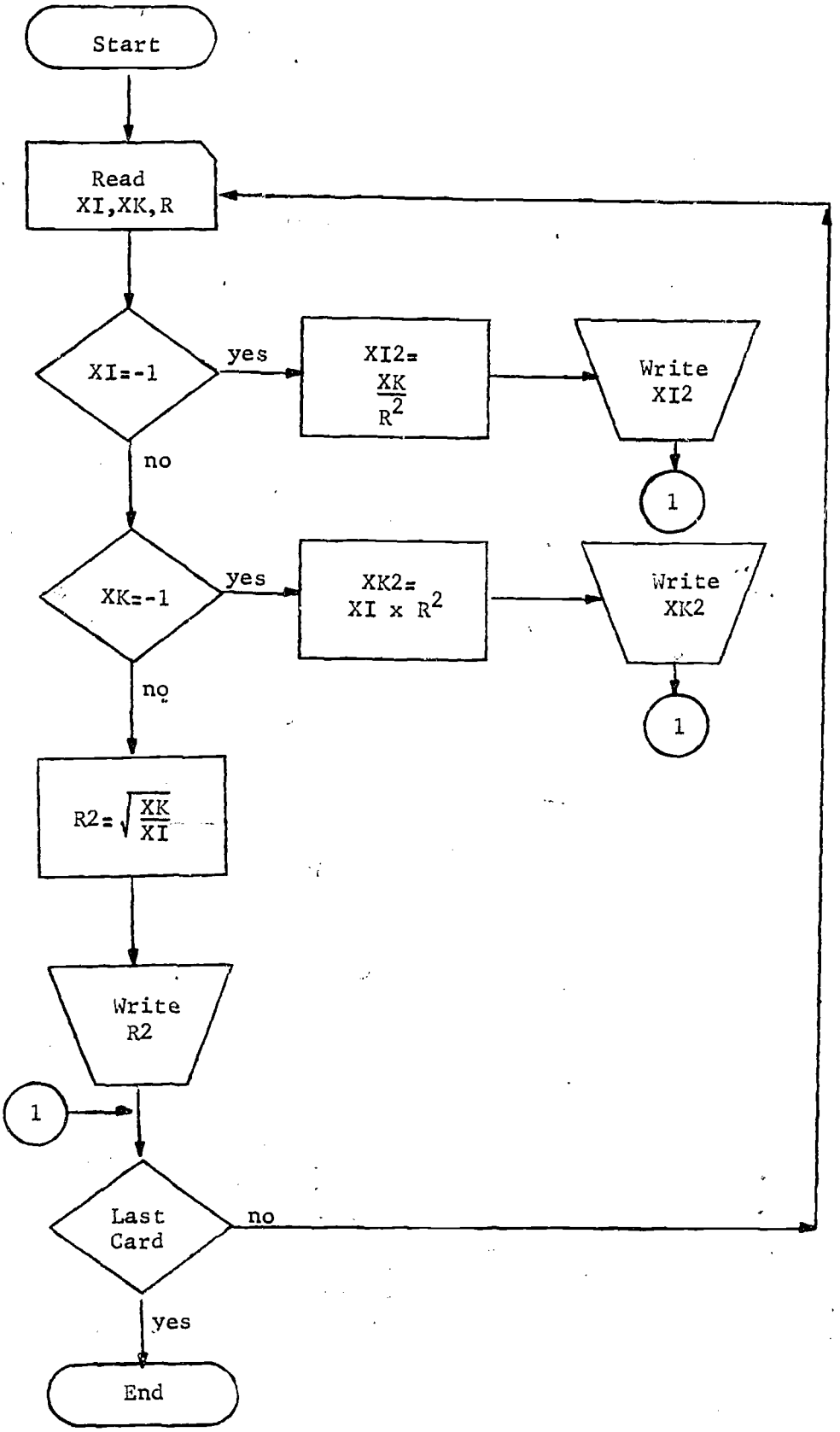
```
C       THIS PROGRAM WILL COMPUTE THE ACCELERATION, DISTANCE TRAVELED, AND
C       THE VELOCITY OF SOME OBJECT, SUCH AS A ROCKET, WHERE THE FORCE
C       REMAINS CONSTANT BUT THE MASS DECREASES.
        DIMENSION ACC(120),XMASS(120),DIS(120),V(120)
        READ(2,2)FORC
      2 FORMAT(F12.1)
        READ(6,2)RATE
        READ(6,2)XMASS
        T=0.
        DO 25 I=1,120
        T=T+1.
        XMASS(I)=XMASS-T*RATE/2.
        ACC(I)=FORC/XMASS(I)
        DIS(I)=((ACC(I)*T**2)/2.)
        V(I)=(2.*(ACC(I)*DIS(I)))**.5
     25 CONTINUE
        WRITE(3,22)
     22 FORMAT(2X,'MASS',8X,'ACCELERATION',8X,'DISTANCE TRAVELED',8X,'VELO
    1CITY',///)
        WRITE(3,21)(XMASS(I),ACC(I),DIS(I),V(I),I=1,120)
     21 FORMAT(2X,F10.3,10X,F10.3,10X,F10.3,10X,F10.3,10X,F10.3)
        CALL EXIT
        END
```

LIGHT INTENSITY

This program solves the equation $I = K/R^2$. The unknown term must be entered as the value -1.

VARIABLES

XI - the intensity of the light
XK - the candle power of the source
R - the distance of the source to the meter


Start
Read XI,XK,R
XI=-1
yes
XI2= XK / R^2
Write XI2
1
no
XK=-1
yes
XK2= XI x R^2
Write XK2
1
no
R2= √(XK/XI)
Write R2
1
Last Card
no
yes
End

IBM FORTRAN Coding Form

LIGHT INTENSITY

FORTRAN STATEMENT

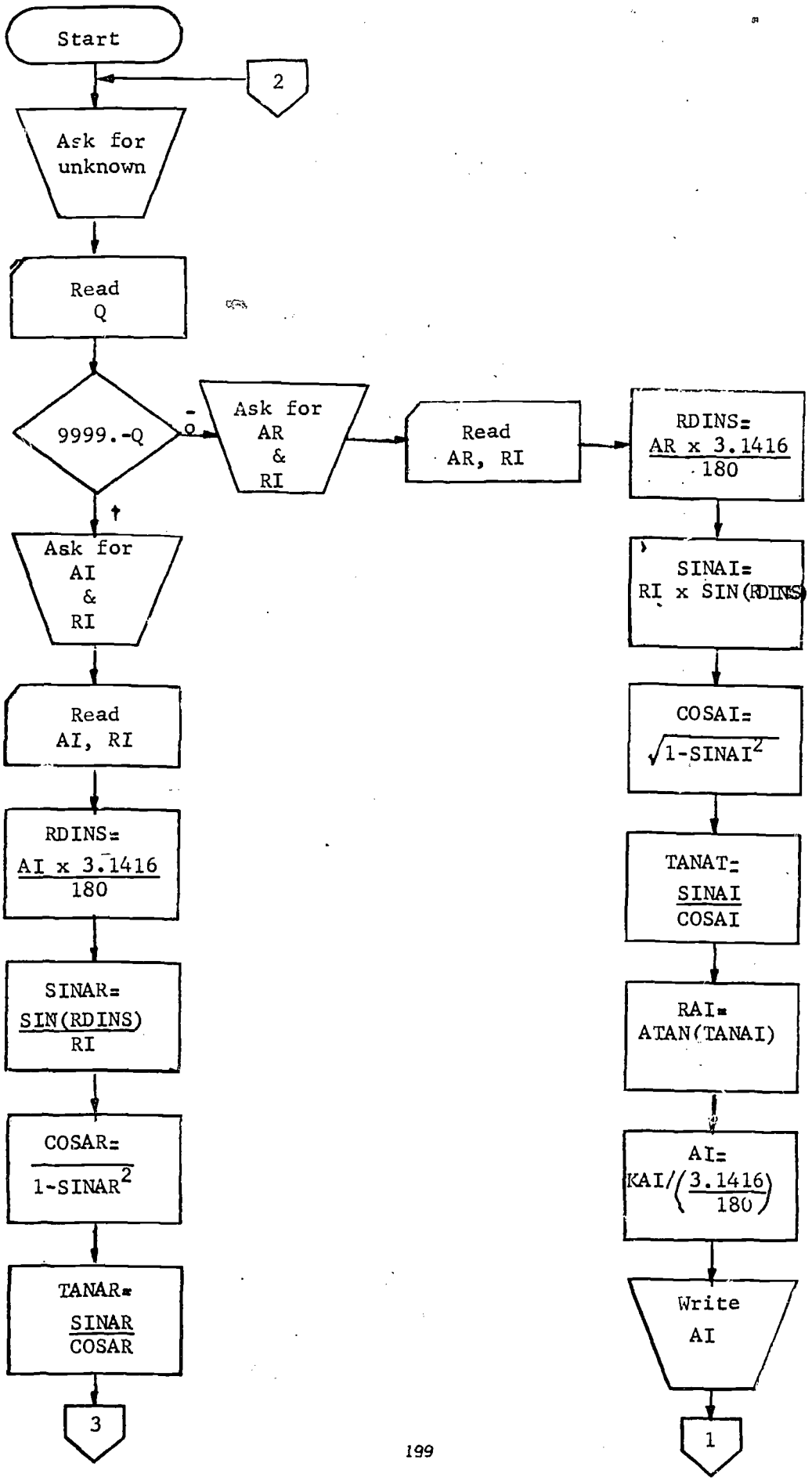
```
      WRITE (3, 1)
    1 FORMAT (2X, 'PROGRAM TO FIND I, K, OR R IN THE EQUATION I=K/R**2 WHERE
     1 I=LIGHT INTENSITY, K=CANDLE POWER OF SOURCE,'/', AND R=DISTANCE OF
     2SOURCE FROM METER.'//)
   55 READ(2, 3)XI, XK, R
    3 FORMAT(3F11.4)
      IF(1.+XI)21, 51, 21
   21 IF(1.+XK)57, 53, 57
   51 XI2=XK/R**2
      WRITE(3, 4)XI2
    4 FORMAT(2X, 'THE INTENSITY IS', F12.5//)
      GO TO 55
   53 XK2=XI*R**2
      WRITE(3, 5)XK2
    5 FORMAT(2X, 'THE POWER OF THE SOURCE IS', F12.6//)
      GO TO 55
   57 R2=SQRT(XK/XI)
      WRITE(3, 6)R2
    6 FORMAT(2X, 'THE DISTANCE OF THE SOURCE IS', F12.6//)
      GO TO 55
   54 CALL EXIT
      END
```

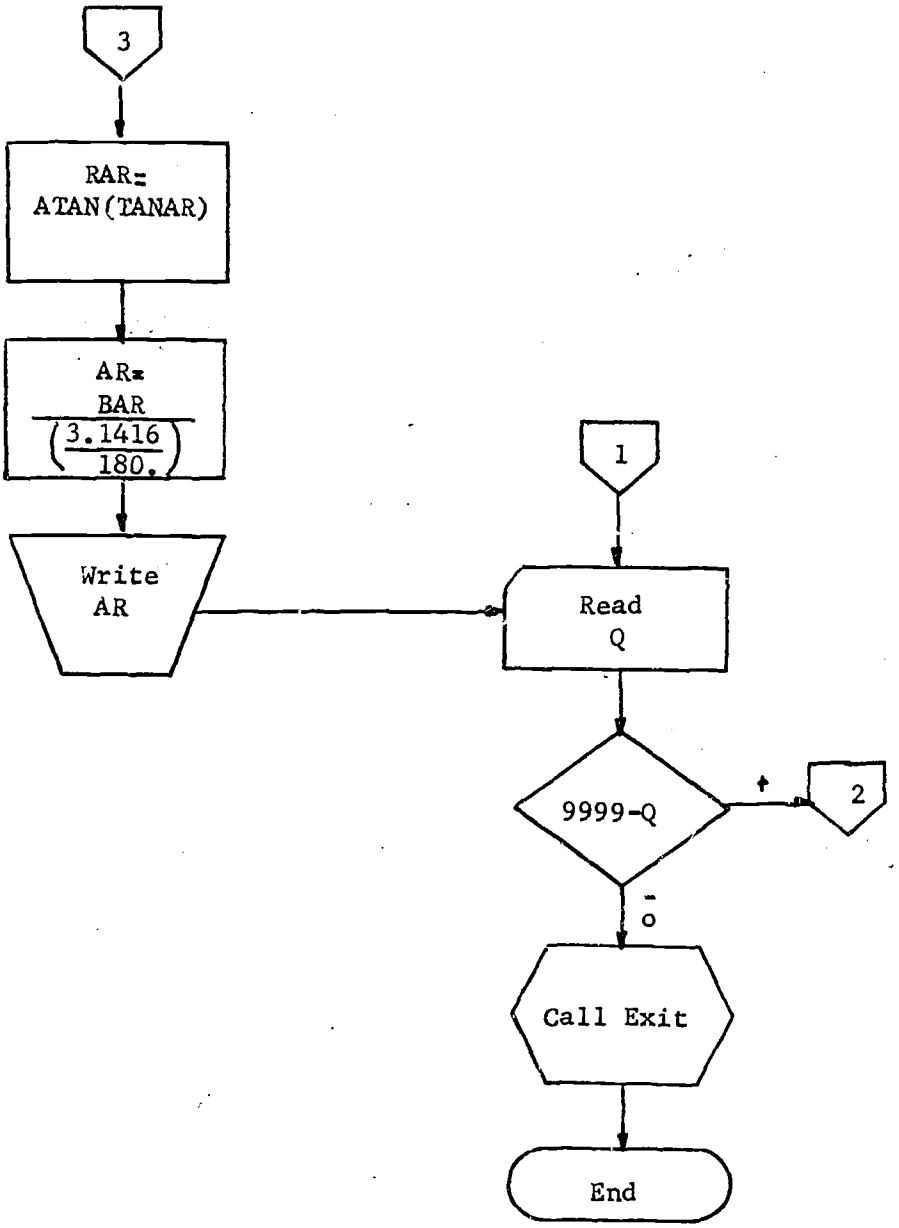
THE INDEX OF REFRACTION

This program will find the angle of incidence if the angle of refraction is known, or vice-versa, for any given index of refraction.

VARIABLES

Q - variable to determine which angle has been entered
AR - angle of refraction
RI - index of refraction
AI - angle of incidence


Start
2
Ask for unknown
Read Q
9999.-Q
0
Ask for AR & RI
Read AR, RI
RDINS= AR x 3.1416 / 180
SINAI= RI x SIN(RDINS)
COSAI= √1-SINAI²
TANAT= SINAI / COSAI
RAI= ATAN(TANAI)
AI= KAI/(3.1416/180)
Write AI
1
Ask for AI & RI
Read AI, RI
RDINS= AI x 3.1416 / 180
SINAR= SIN(RDINS) / RI
COSAR= 1-SINAR²
TANAR= SINAR / COSAR
3


3
RAR=
ATAN(TANAR)
AR=
BAR
(3.1416/180.)
Write
AR
1
Read
Q
9999-Q
+
2
-
0
Call Exit
End

IBM

FORTRAN Coding Form

THE INDEX OF REFRACTION

1 2

```
C THIS PROGRAM FINDS THE ANGLE OF INCIDENCE OR THE ANGLE OF REFRACTION
C OF A LIGHT RAY.
      WRITE (1,1)
    1 FORMAT ('PROGRAM DEALING WITH ANGLES OF INCIDENCE AND REFRACTION',
     1///)
   57 WRITE (1,2)
    2 FORMAT ('IF YOU WISH TO FIND AN ANGLE OF INCIDENCE ENTER 9999. IF Y
     1OU WISH TO FIND AN',/,'ANGLE OF REFRACTION ENTER 8888.'//)
      READ (6,3)Q
    3 FORMAT (F5.0)
      IF (9999.-Q) 51,51,53
   51 WRITE (1,4)
    4 FORMAT ('ENTER THE ANGLE OF REFRACTION AND THE INDEX OF REFRACTION
     1IN THE FORM XXX.XXXX',/,'LEAVING ONE SPACE AFTER EACH ENTRY.'//)
      READ (6,5)AR,RI
    5 FORMAT (2F9.4)
      RDINS = AR*3.1416/180.
      SINAI = RI*SIN(RDINS)
      COSAI = SQRT(1.-SINAI**2)
      TANAI = SINAI/COSAI
      RAI = ATAN(TANAI)
      AI = RAI/(3.1416/180.)
```

IBM

FORTRAN Coding Form

PAGE 2 OF 2

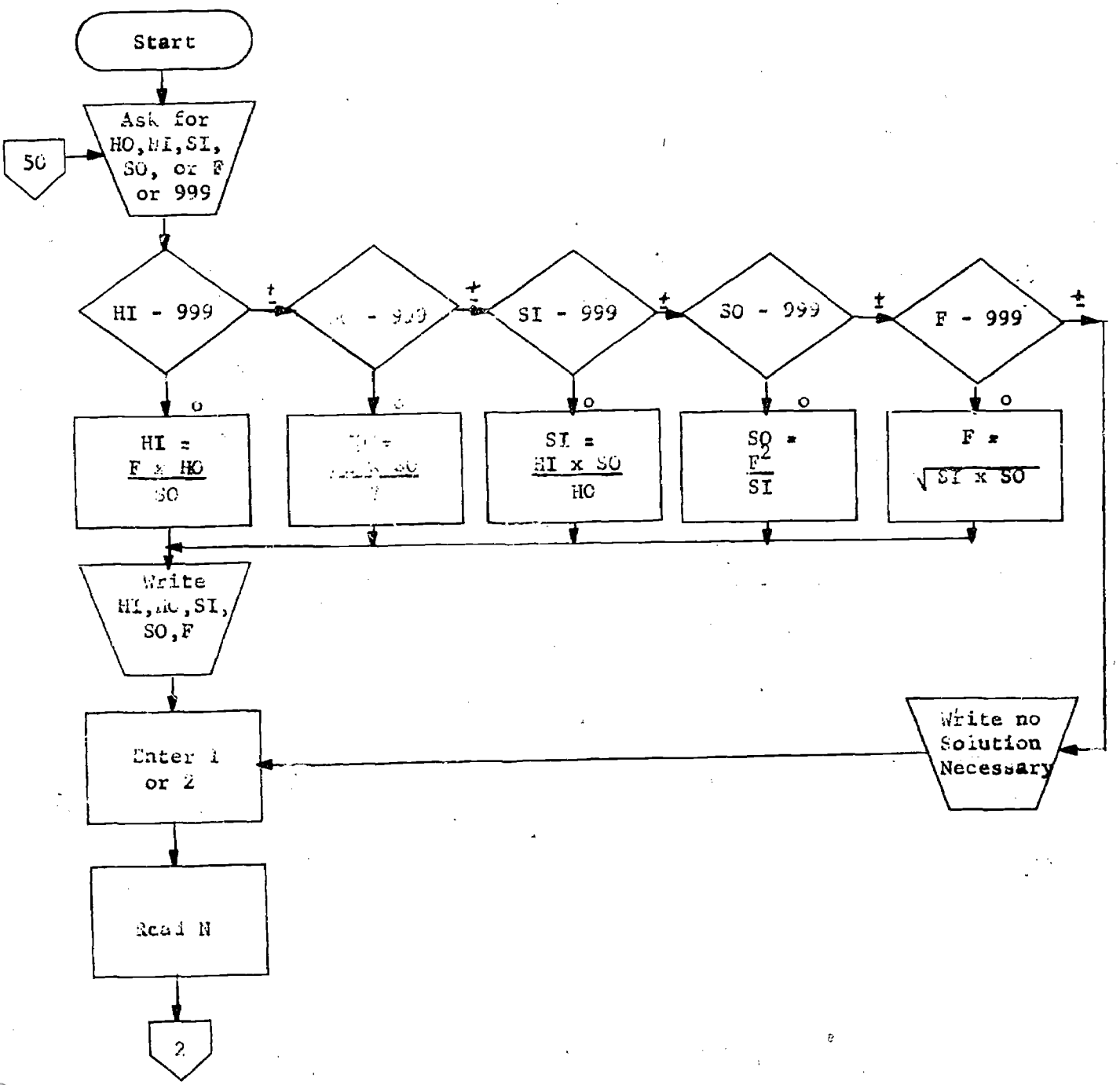
```
      WRITE (1,6)AI
    6 FORMAT ('THE ANGLE IS',F8.2,'DEGREES',//)
   55 WRITE (1,7)
    7 FORMAT ('IF THIS IS YOUR LAST PROBLEM,ENTER 9999. IF NOT,ENTER 8888
     1.')
      READ (6,3)Q
      IF (9999.-Q) 54,54,57
   53 WRITE (1,8)
    8 FORMAT ('ENTER THE ANGLE OF INCIDENCE AND THE INDEX OF REFRACTION
     1IN THE FORM XXX.XXXX',/,'LEAVING ONE SPACE AFTER EACH ENTRY.',/)
      READ (6,5)AI,RI
      RDINS = AI*3.1416/180.
      SINAR = SIN(RDINS)/RI
      COSAR = SQRT(1.-SINAR**2)
      TANAR = SINAR/COSAR
      RAR = ATAN(TANAR)
      AR = RAR/(3.1416/180.)
      WRITE (1,6)AR
      GO TO 55
   54 CALL EXIT
      END
```

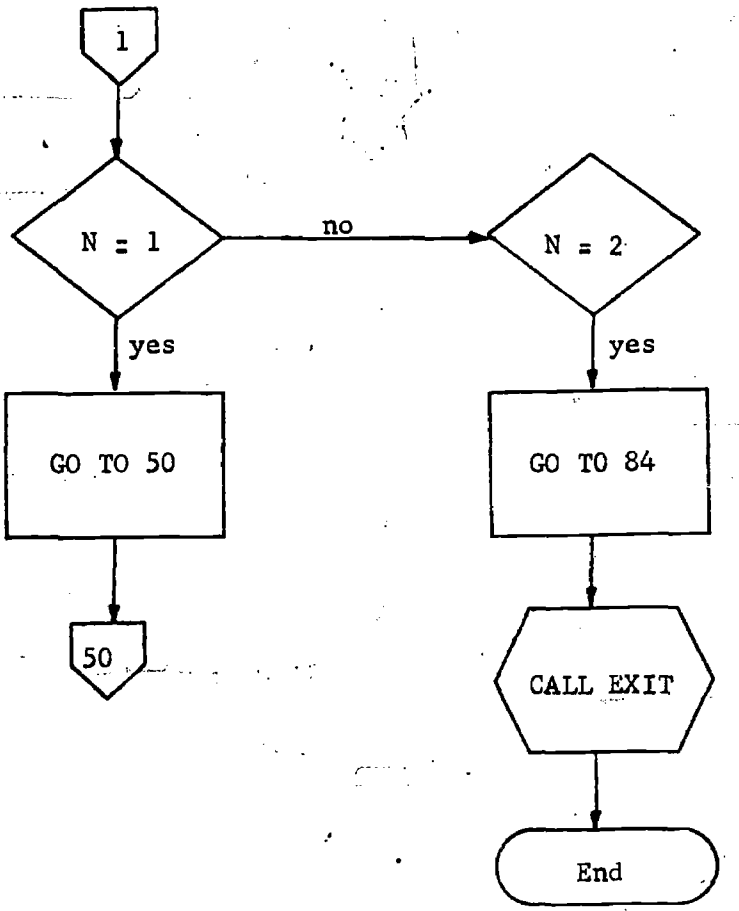
PARABOLIC MIRROR

This program uses a series of IF statements to determine the unknown value. After solving for the appropriate unknown, the program direction is determined through the use of a COMPUTED GO TO statement.

VARIABLES

HI - the height of the image
HO - the height of the object
SI - the image distance
SO - the object distance
F - the focal distance





IBM

FORTRAN Coding Form

PARABOLIC MIRROR

```
C     THIS PROGRAM WILL COMPUTE THE VALUE OF ONE UNKNOWN IN THE
C     EQUATIONS DEALING WITH PARABOLIC MIRRORS
   50 WRITE(1,27)
   27 FORMAT(2X'WRITE IN VALUES FOR THE IMAGE HEIGHT,OBJECT HEIGHT,IMAGE
     1DISTANCE,OBJECT DISTANCE,'/'AND THE FOCAL DISTANCE.TYPE IN 999.FOR
     2THE VALUE YOU WISH TO COMPUTE.'/'IN TYPING VALUES BE SURE TO PLACE
     3THE DECIMAL POINT IN EACH REQUIRED VALUE.')
      READ(6,28)HI
   28 FORMAT(F10.3)
      READ(6,28)HO
      READ(6,28)SI
      READ(6,28)SO
      READ(6,28)F
      IF(HI-999.)200,300,200
  200 IF(HO-999.)201,301,201
  201 IF(SI-999.)202,302,202
  202 IF(SO-999.)203,303,203
  203 IF(F-999.)204,304,204
  204 WRITE(1,32)
   32 FORMAT(2X'YOU HAVE SUPPLIED ALL VALUES THEREFORE NO COMPUTATION IS
     1NECESSARY.')
      GO TO 34
  300 HI=F*HO/SO
      GO TO 34
```

IBM FORTRAN Coding Form

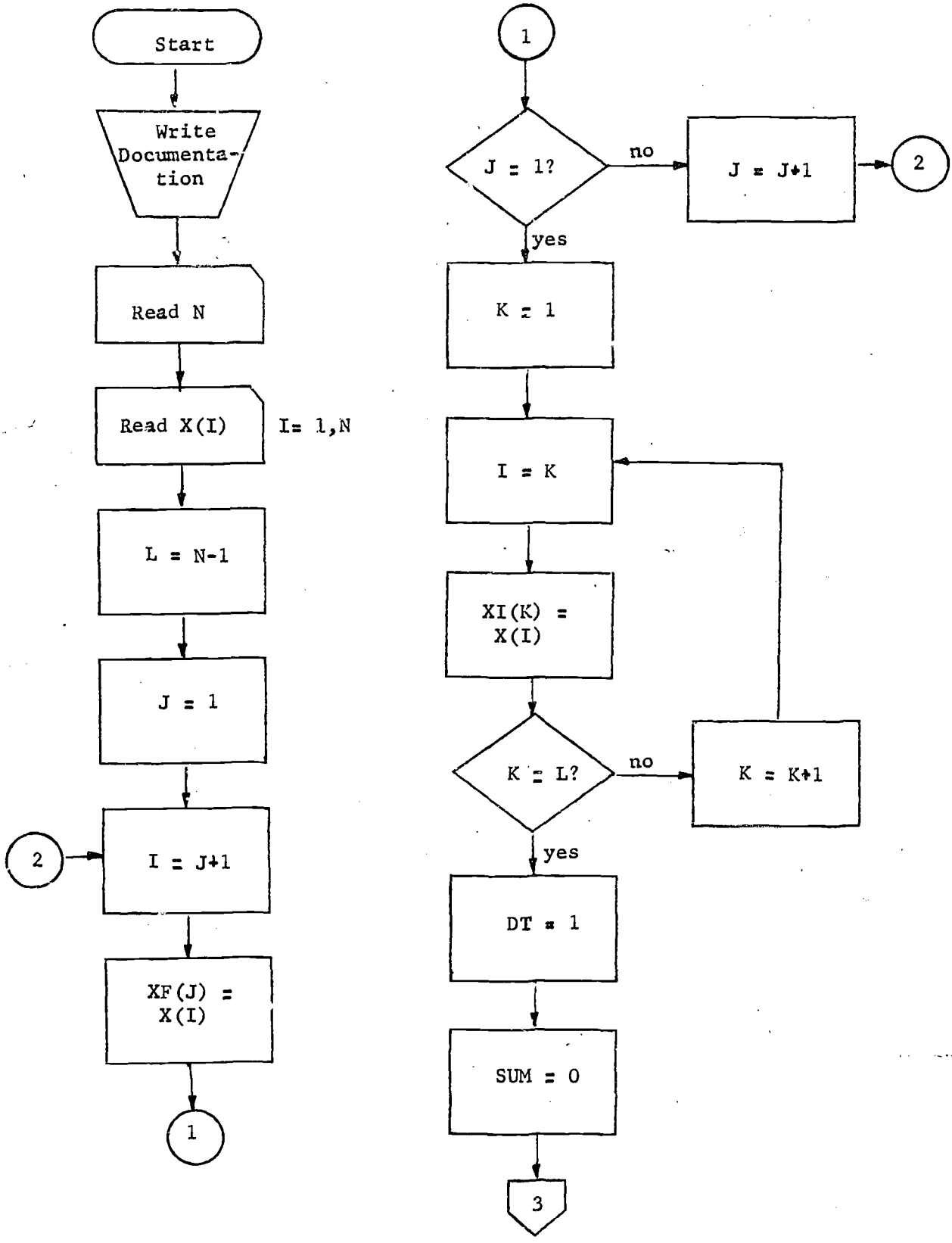
```
      HO=HI*50/F
      GO TO 34
302   SI=HI*50/HO
      GO TO 34
303   SO=F**2/SI
      GO TO 34
304   F=SQRT(SI*SO)
20    WRITE(3,25)HI,HO,SI,SO,F
25    FORMAT(2X'IMAGE HEIGHT',F8.2,2X,OBJECT HEIGHT',F8.2,2X,IMAGE DISTA
     1NCE',F8.2,2X,OBJECT DISTANCE',F8.2,2X,'FOCAL DISTANCE',F8.2)
34    WRITE(1,35)
35    FORMAT'2X'IF YOU HAVE ANOTHER PROBLEM ENTER 1,IF NOT ENTER 2.')
      READ(6,36)N
36    FORMAT(I1)
      GO TO (50,84)N
84    CALL EXIT
      END
```

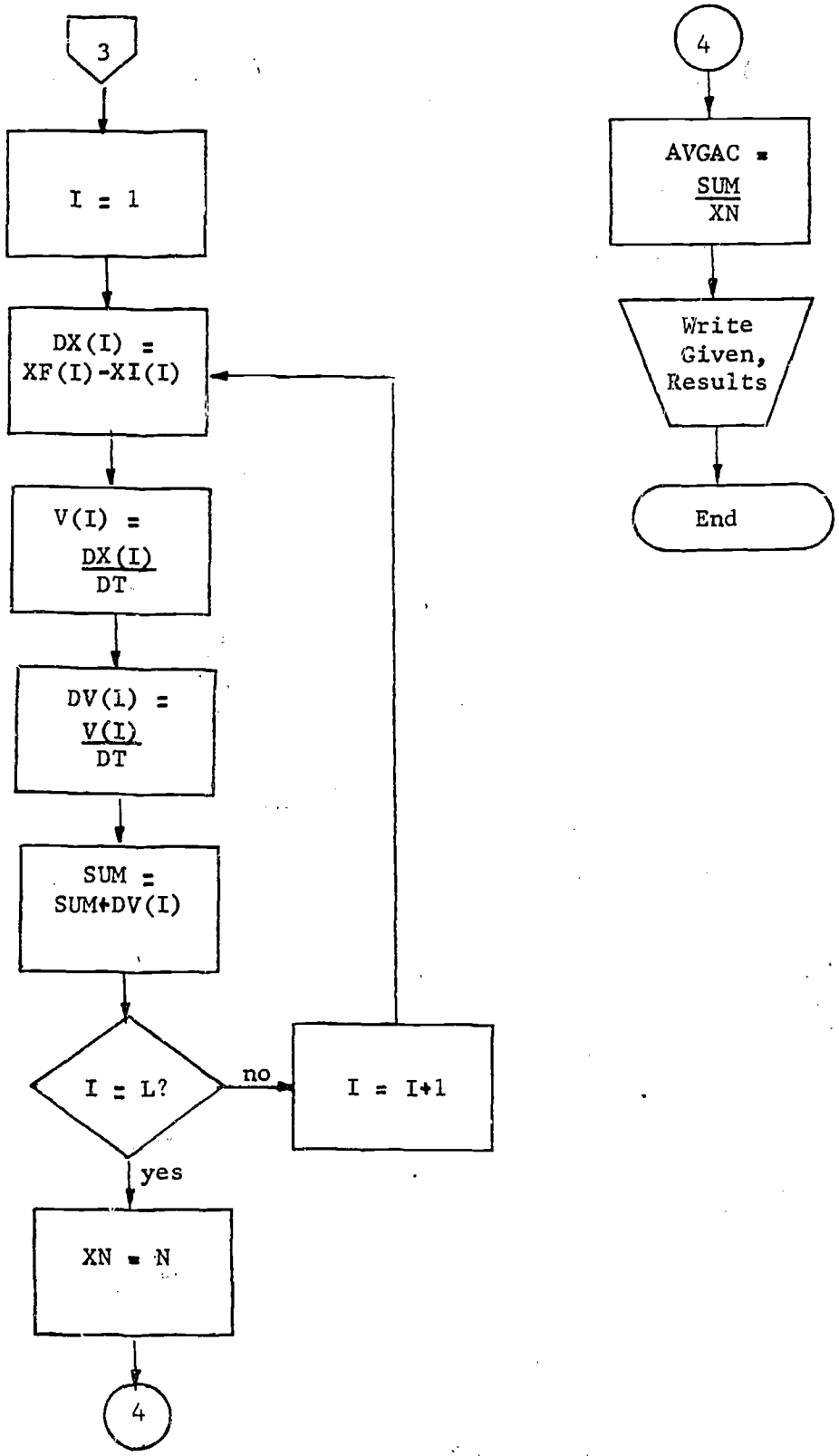
ACCELERATION

This program is designed to make the computations necessary to compute a value for the average acceleration of a body acted upon by a constant force. The most important data required by this program are the displacement positions along the X axis.

VARIABLES

N - the number of displacement positions
X - the values of the displacement positions
L - number of intervals
XI - initial position of object in each interval
XF - final position of object in each interval
DT - time constant for the intervals
V - velocity
DV - change in velocity
XN - number of values
AVGAC - average acceleration


Start
Write Documenta-tion
Read N
Read X(I)
I= 1,N
L = N-1
J = 1
2
I = J+1
XF(J) = X(I)
1
1
J = 1?
no
J = J+1
2
yes
K = 1
I = K
XI(K) = X(I)
K = L?
no
K = K+1
yes
DT = 1
SUM = 0
3


3
I = 1
DX(I) = XF(I)-XI(I)
V(I) = DX(I) / DT
DV(I) = V(I) / DT
SUM = SUM+DV(I)
I = L?
no
I = I+1
yes
XN = N
4
4
AVGAC = SUM / XN
Write Given, Results
End

IBM

FORTRAN Coding Form

ACCELERATION

1 2

FORTRAN STATEMENT

```
      DIMENSION X(100),XI(100),XF(100),DX(100),V(100),DV(100)
      WRITE(3,2)
    2 FORMAT(2X,'THIS PROGRAM WILL ANALYZE EXPERIMENTAL DATA AND COMPUTE
     1 THE AVERAGE ACCELERATION FOR',//,'A BODY UPON WHICH A CONSTANT FO
     2RCE ACTS. THE PROGRAM MUST BE SUPPLIED WITH THE FOLLOWING',//,'DAT
     3A--THE NUMBER OF DISPLACEMENT POSITIONS, THE TIME INTERVAL INVOLVE
     4D, AND THE VALUES',//,' OF THE DISPLACEMENT POSITIONS.',///)
      READ(2,3)N
    3 FORMAT(F10.3)
      READ(2,4)(X(I),I=1,N)
    4 FORMAT(5F10.3)
      L=N-2
      DO 255 J=1,L
      I=J+1
      XF(J)=X(I)
  255 CONTINUE
      DO 25 K=1,L
      I=K
      XI(K)=X(I)
   25 CONTINUE
      DT=.1
      SUM=0.
      DO 10 I=1,L
      DX(I)=XF(I)-XI(I)
```

IBM

FORTRAN Coding Form

PROGRAM

PROGRAMMER

DATE

PUNCHING INSTRUCTIONS

GRAPHIC

PUNCH

PAGE 2 of 2

FORTRAN STATEMENT

```
      V(I)=DX(I)/DT
      DV(I)=V(I)/DT
      SUM=SUM+DV(I)
   10 CONTINUE
      XN=N
      AVGAC=SUM/XN
   18 WRITE(3,5)
    5 FORMAT(2X,'INITIAL',11X,'FINAL',10X,'CHANGE IN',5X,'CHANGE IN',7X,
     1'ACCELERATION')
      WRITE(3,87)
   87 FORMAT(2X,'POSITION',8X,'POSITION ',8X,'POSITION',9X,'VELOCITY',//
     1//)
      WRITE(3,6)(XF(I),XI(I),DX(I),V(I),DV(I),I=1,L)
    6 FORMAT(4X,F9.3,7X,F9.3,8X,F9.3,7X,F9.3,7X,F9.3,///)
      WRITE(3,7)AVGAC
    7 FORMAT(' THE AVERAGE CHANGE IN ACCELERATION IS',F8.4)
      CALL EXIT
      END
```