EXTENDED BASIC

FOR USE WITH THE

WANG 3300 COMPUTER

Compiled by

Charles P. Burdick

September 1971

This material is intended for experimental and developmental purposes only.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION TO BASIC

CHAPTER 2 LOOPS AND LIBRARY FUNCTIONS

CHAPTER 6 TWO-DIMENSIONAL ARRAYS

CHAPTER 7 USER-DEFINED FUNCTIONS AND SUBROUTINES

CHAPTER 8 MATRIX OPERATIONS

CHAPTER 1

INTRODUCTION TO BASIC

1.1 THE BASIC LANGUAGE

BASIC is an acronym for Beginner's All-purpose Symbolic Instruction Code. The language was first developed by Professors John G. Kemeny and Thomas E. Kurtz at Dartmouth College under the terms of a National Science Foundation grant, and it was first used on the General Electric 265 time-sharing computer system.

It was originally intended for people who have had no training nor experience in computer work. However, it has proven to be so powerful yet so simple to learn that it has become very popular among professionals who use computers in their work, including scientists, mathematicians, engineers, teachers, students, and people in the business field.

The original BASIC has been expanded and this expanded version is referred to as EXTENDED BASIC. This text contains the expanded version. Every reference to BASIC in this text should be interpreted as referring to the expanded version.

The WANG 3300 has features not available on some computers. Since the language presented in this text is intended for use on the WANG 3300, slight modifications may be necessary when writing BASIC programs for use on other computers.

1.2 A PROGRAM IN BASIC

A BASIC program usually consists of data being read into the computer, calculations being performed on the data, and calculated answers being printed out. The program itself is simply a series of typed lines, each beginning with a line number followed by an instruction or data. The computer will execute the instruction exactly as written.

An instruction usually consists of a short English word (LET, PRINT, READ, DATA), some combination of one or more constants (23.45, 1.456E-09), variables (H8, F, K), mathematical expressions (A + B - C/D), and/or mathematical functions (LOG, COS, SQR).

The instructions are executed one at a time in order of increasing line number, regardless of the order in which they are typed.

When programs are written for the WANG 3300, several instructions may be written on a single line. This feature is not available on some other computers.

BASIC programs are entered into the computer by means of a teletype terminal. The model 3315 teletype terminal provides the characters listed on the following page.

a) The capital letters A through Z

b) The digits 0 through 9

c) = Equal to

d) < Less than

e) > Greater than

f) . Decimal point

g) (Left parenthesis

h)) Right parenthesis

i) + Plus

j) - Minus

k) * Asterisk

l) / Slash

m) \ Backslash

n) ↑ Up arrow

o) ← Back arrow

p) ! Exclamation mark

q) , Comma

r) ; Semicolon

s) : Colon

t) $ Dolar sign

u) # pound sign

v) " Double quote

w) ' Single quote

x) @ At

y) & Ampersand

z) Blank

1.3 CONSTANTS

A numeric constant may consist of as many as eight digits. It may be either positive or negative, and may or may not contain a decimal point.

The magnitude of the constant must be less than 10^{63} and greater than or equal to 10^{-65} .

A number may be expressed as the product of a constant and some power of 10. The number $3.1416 \text{ X } 10^{9}$ may be expressed as any one of the following:

3.1416E9　　　　3.1416E09　　　　3.1416E+09

The number $3.1416 \text{ X } 10^{-9}$ may be expressed as 3.1416E-9 or as 3.1416E-09.

This type of notation is called exponential notation or E-format.

Commas are not permitted in constants. The number 1,234,567 is written in BASIC as 1234567.

Constants that are irrational numbers may be approximated by the use of a decimal number or expressed with fractional exponents. For example, the number π may be written as 3.1415927 or rounded to any desired number of significant figures less than nine.

Fractions are written as the quotient of two integers.

Complex numbers cannot be written in BASIC.

Exercises 1.3

1. Write each of the following as a regular decimal number.

a) 3.142E+02

b) 8.197E3

c) 1.99E-1

d) 3.718E-04

e) 4.44E+2

f) 1.542E+4

g) 1.0E+06

h) 1.0E-02

i) 9.987E-3

j) 1.5E6

2. Write each of the following in BASIC exponential form using the format #.####E±##, where # represents a digit.

a) .007841

b) 1,601,000

c) .1492

d) 156,100,000,000

e) .09994

f) 1.234×10^{36}

g) 8.765×10^{-17}

h) 1.492×10^{2}

i) $.003232 \times 10^{7}$

j) 832.9×10^{-3}

1.4 VARIABLES

A variable used to represent a number specified or calculated elsewhere in the program may be designated either by a letter or by a letter followed by a digit. Examples of such variables are: F H K7 M4

A variable representing such a number may take on different values during the execution of the program.

Subscripted variables and string variables are presented in later chapters.

1.5 ARITHMETIC OPERATIONS

The symbols for the arithmetic operations are as follows:

Operator	Operation	Example
+	Addition	X + 6.789
-	Subtraction	A - B3
*	Multiplication	X * Y
/	Division	C / D
↑	Exponentiation	X↑2

Notice that in BASIC letters are always written as capital letters. Spaces are optional. A + B may be written as A+B. Rational expressions must be written on one line, with the numerator separated from the denominator by the division symbol /.

Two consecutive operator sumbols are not permitted.

-3 * -9 incorrect

-3 * (-9) correct

A positive number may be raised to a power that is positive, negative, integer, or decimal.

A negative number may be raised to a positive or negative integer power. It cannot be raised to a decimal power which would produce an imaginary result.

Zero cannot be raised to the zero power.

Division by zero is undefined and not permitted.

A number such as 10^6 may be expressed as either 10↑6 or 1.0E6. Although either is correct, exponential notation (E-format) results in greater computer efficiency.

The asterisk must be used whenever multiplication is intended. Although in algebra xy means to multiply the value of x and the value of y, in BASIC, XY has no meaning. If multiplication is intended, it must be expressed as X*Y.

Numbers containing radicals must be written with fractional or decimal exponents. For example:

$\sqrt{17}$	17↑.5
$7\sqrt{141}$	7 * 141↑.5
$\sqrt[3]{29}$	29↑(1/3)
$\sqrt[5]{13^3}$	(13↑3)↑(1/5) or 13↑(3/5) or 13↑.6

Exercises 1.5

1. Write each of the following in proper BASIC notation:

a) $(6.43)^3$

b) $3X$

c) $a + b \div c$

d) $17.98 \div 3.19$

e) $(-17.81)(-31.62)$

f) y^5

g) πD

h) $xy + h$

i) $d + \frac{e}{f}$

j) $W^{1.5}$

k) $\sqrt[3]{x^2}$

l) $\frac{\sqrt{a^2 + b^2}}{a}$

2. Write each of the following in regular arithmetic or algebraic notation.

a) 87.42 * 91.88

b) A/B

c) 3.1416 * W

d) 87.4 ÷ 63.9 − K

e) 77.65↑3

f) 3.456/M

g) 43.9*X+Y↑.5

h) V↑(2/3)

i) A+B-C/D

j) -1891 * (-42.63)

1.6 PARENTHESES

Use of parentheses in BASIC is the same as in algebra.

Operations within parentheses are performed first.

For each left parenthesis, there must be a right parenthesis, and for e ach right parenthesis there must be a left parenthesis.

Parentheses may be used even when not necessarily required.

A pair of parentheses may be w ritten within another pair of parentheses. These are then called <u>nested parentheses</u>. There is no limit as to how many levels parentheses may be nested.

There are no braces or brackets in BASIC. Nested parentheses are used instead. For example:

<u>algebra</u>	<u>BASIC</u>
$[a(b + c)]^2$	(A * (B + C))↑2

An operation symbol is always required between a pair of parenthesized expressions. For example:

(A + B)(C + D)	incorrect
(A + B)*(C + D)	correct

When translating fractional expressions into BASIC, frequent use is made of parentheses. A general rule which may be of help is as follows:

1. Enclose each numerator in parentheses.
2. Enclose each denominator in parentheses.
3. Enclose each fraction in parentheses.

For example:

algebra	BASIC
$\dfrac{\frac{a + b}{cd}}{\frac{ad}{c - d}}$	((A + B) / (C * D))/((A * D)/(C - D))

This may result in a BASIC expression having more parentheses than is absolutely necessary, but at least the expression will be correct.

Exercises 1.6

Write each in BASIC notation, using parentheses whenever necessary.

1. $\frac{a + b}{c}$ (A+B)/C

2. $\frac{x}{y - w}$ X/(Y-W)

3. $a(k + m)$ A*(K+M)

4. $\frac{a + b}{c - d}$ (A+B)/(C-D)

5. $\frac{ab}{cd}$ A*B/(C*D)

6. x^{w-3} X↑(W-3)

7. $[a(x + y)]^3$ (A*(X+Y))↑3

8. $(x + y)(x - y)$ (X+Y)*(X-Y)

9. $d\left(\frac{a + b}{c}\right)$ D*((A+B)/C)

10. $\frac{\frac{e + s}{g}}{\frac{ef}{h}}$ ((E+S)/G)/(E*F/H)

Opposite each algebraic expression below is a corresponding BASIC expression which contains at least one error. Describe the error and write the correct BASIC expression.

	algebraic	BASIC
11.	$37(j - n)$	37*(J - N) * sign
12.	K^{2+n}	K↑(2 + N)
13.	$\frac{e - f}{g}$	(E - F)/G
14.	$7a + c$	7*(A + C)
15.	$\left(\frac{p + q}{r}\right)^3$	((P + Q)/R)↑3
16.	$a[x + b(y + c)]$	A*(X + B*(Y + C))
17.	$c[(a + b)^3]$	C*((A + B)↑3)
18.	$\frac{a + b}{c + d}$	(A + B)/(C + D)
19.	$\sqrt{a^2 + b}$	(A↑2 + B)↑.5

1.7 EXPRESSIONS

Any valid combination of constants, variables, operations and functions is called an expression.

Examples of valid expressions: 5.678

3.57 + F - X/Y↑2

1234 - I3/(X + Y)

A variable preceded by a minus sign is a valid expression. -1 * R may be written -R. However, -R is not a valid variable name as variable names must begin with a letter.

Exponents may consist of valid expressions.

Examples: X↑(W - 3)

Y↑(2 * X + 4)

K↑(J - 3)

Care must be taken to ensure that a negative number is not being raised to a decimal power, giving an imaginary result.

1.8 ORDER OF OPERATIONS

Arithmetic operations are performed in BASIC in the same order they are performed in algebra:

1. Operations within parentheses are performed first.
2. Exponentiation is then performed.
3. Multiplications and divisions are then performed in the order they occur from left to right.
4. Additions and subtractions are then performed in the order they occur from left to right.

In an expression such as -B↑3, the exponentiation will be performed first, and then the inverse of the result taken. This is therefore equivalent to -(B↑3). If (-B)↑3 is intended, it must be written as such.

Exercises 1.8

Perform the indicated oprerations and compute the final answer.

1. 12 - 10 * 36
2. 49 - 6↑2
3. 20 + 24 / 2
4. 2 * 3 + 4 * 7
5. 10 + 3 * 2 - 16 / 4
6. 2 * 3↑2 / 6
7. 2 * 4 / 8 * 25 / 5
8. 6↑2 + 3 * 10 - 50
9. 3↑(5- 2)
10. (10 - 6) * (4 + 3)
11. 10 - 6 * 4 + 3
12. 2 + 2↑(3 + 1)
13. (2 + 2)↑3 + 1
14. 4 * 20 - 18 / (2 + 1)↑ 2
15. 4 * (20 - 18) / 2 + 1↑ 2
16. (2 + 8)↑ 2 + 20 / 25↑.5

For exercises 17 - 28, write one of the words 'exponentiation' 'multiplication,' 'division,' 'addition,' or 'subtraction' to indicate which operation would be performed **first** in the given expression.

17. A + B - C * D
18. E + F / G
19. (E + F) / G
20. D / E * F + H - H
21. A / B↑X + W
22. A / B + C * D
23. (P + Q - R) * S
24. (S - T + W)↑X
25. S * T + U / (V - W)↑3
26. ((A - B) * C)↑D + E
27. -X↑4 + Y
28. Y - X↑4

For exercises 29 - 40, write the name of the operation which would be performed **last** in the given expression in exercises 17 - 28.

1.9 LINE NUMBERS

Every program statement must be assigned a line number from one to four digits in length. Line numbers identify the lines and indicate the order in which the program lines are to be executed by the system. The lines do not have to be entered in sequence. They are processed in order by line number.

Line numbers should be assigned so that there are unassigned numbers between consecutive statements. Additional lines may then be inserted without changing other statement line numbers. It is common practice to use multiples of 10 for line numbers such as 10, 20, 30, 40, 50, etc., so that other lines may be inserted if necessary.

1.10 THE REM STATEMENT

```
10 REM   PROGRAM FOR SOLVING QUADRATIC EQUATIONS
20 REM   ROOTS WILL NOT BE CALCULATED IF COMPLEX
30 REM   IF ROOTS COMPLEX, MESSAGE WILL BE PRINTED
```

The REM statement consists of a line number followed by REM followed by any remark the programmer wishes to make to identify, explain, or clarify his program. REM statements may be placed anywhere in the program, and as many may be used in the program as desired.

1.11 THE LET STATEMENT

```
10   LET D = B↑2 - 4*A*C
20   LET R1 = (-B + D↑.5)/(2*A)
30   LET R2 = (-B - D↑.5)/(2*A)
```

The LET statement consists of a line number, the word LET, a variable name, an equal sign, and any valid BASIC expression, in that order.

The syetem will evaluate the expression to the right of the equals sign and assign this value to the variable named on the left of the equals sign.

On the WANG computer, use of the word LET is optional. Each of the above statements could have been written as:

```
10 D = B↑2 - 4*A*C
20 R1 = (-B + D↑.5)/(2*A)
30 R2 = (-B - D↑.5)/(2*A)
```

A LET statement may change the value of a variable in a program. For example:

```
40   LET X = 3.1416
50   LET X = X + 1
```

After the execution of these two statements, the system will have changed the value of X to 4.1416. The value assigned to X will remain 4.1416 until changed by further statements.

Although X = X + 1 is a very meaningful statement in BASIC, it is a meaningless algebraic equation. There is no number that X can represent that will make the statement true. This points out the difference between a LET statement and an algebraic equation.

To translate an algebraic equation into a BASIC LET statement:

FIRST: Write the equation so that the left side consists of one variable whose value is to be calculated, and so that the right side consists of the constants and the variables whose values are given or can be calculated.

SECOND: Write the appropriate LET statement, being careful to indicate each operation, and to use parentheses where needed.

For example, the proper LET statement for the algebraic equation $\frac{x}{1456.789} = \frac{14.8734}{.087654}$ would be

```
10 LET X = 14.8734 * 1456.789 / .087654
```

Exercises 1.11

1. Compute the final value of W that would result from the execution of all the given statements in the following program:

```
10 LET E = 10.0
20 LET F = 5.1
30 LET G = E + 3 * F
40 LET E = E + 25 + G
50 LET W = 100 * E
```

2. Which of the following are invalid LET statements? Indicate the error in each such case. Line numbers have been omitted.

```
a)  LET A - B = 6.14169 + E
b)  LET F3 = F - 3   OK
c)  LET -X = X * (-1)
d)  LET A/B = C * G
e)  LET X2 = A + C/D   OK
f)  LET X2 = X * 2   OK
g)  LET R = -R * (-1)   OK
h)  LET R = -R   OK
i)  LET -R = R
j)  LET A*B = B*A
k)  LET AB = A*B
l)  LET 3X = X↑2 - 9.789
m)  LET X3 = X↑2 - 9.987   OK
o)  LET A = -R1 + R2   OK
p)  LET C↑2 = (A↑2 + B↑2)
q)  LET A(B+C) =A*B+A*C
r)  LET A5 = .5 *B3 *H7   OK
s)  LET W + K = K + W   OK
```

3. Write a valid LET statement that could be used to solve each equation for the variable indicated. Be sure to indicate each operation and provide parentheses where necessary. Express π as 3.1416.

	Solve for	
1.	C	$C = 2a^2b^2$
2.	A	$A = \frac{1}{2}bh$
3.	k	$\frac{k}{3} = m + n$
4.	S	$S = 4\pi r^2$
5.	D	$RT = D$
6.	p	$p = 2(h + w)$
7.	C	$C = \frac{5F - 160}{5}$
8.	F	$F = \frac{9C + 160}{5}$
9.	X	$3X = (17.99)(21.44)^2$
10.	I	$RTP = I$
11.	A	$A = bh$
12.	E	$E = (X_2 - X_1)^2$
13.	S	$S = 6a^2$
14.	V	$V = \frac{4\pi r^3}{3}$
15.	P	$5P = S - V$
16.	A	$\frac{A}{19.178} = \frac{21.142}{18.899}$
17.	A	$A = \frac{1}{2}h(b + c)$
18.	W	$W = (x + y)^3$

19. K $K = \left(\frac{J + I}{L + M}\right)^2$

20. P $(Y_2 - Y_1)^2 = P$

21. V $V = \frac{c + d}{e + \frac{f}{r + s}}$

22. C $C = \sqrt{a^2 + b^2}$

23. R $R = \frac{-b + b^2 - 4ac}{2a}$

24. S $S = \left(\frac{x}{y}\right)^{3-a}$

25. T $T = \frac{1}{b^2}\left(\frac{s}{10}\right)^b$

26. K $\frac{308.9}{K} = \frac{187.349}{W}$

27. R $R = ax^2 + bx + c$

28. B $\frac{A}{B} = \frac{C + 87}{D - 99}$

4. For each algebraic equation below there is shown a corresponding LET statement that contains at least one error. Describe the error and rewrite the LET statement correctly.

a) $w = \frac{a + b}{c + d}$ LET W = A + B/C + D

b) $i = prt$ LET I = PRT

c) $3x = 4x^2 - 2$ LET 3X = 4*X↑2 - 2

d) $a = 3b + 2$ LET A = 3B + 2

e) $V = \frac{ab}{c + 9}$ LET V = A*B/C + 9

f) $F = (a + b)(c + d)$ LET F = A + B * C + D

g) $G = \left(\frac{x}{y}\right)^{w-1}$ LET G = (X/Y)↑W - 1

h) $H = \frac{x}{y}^{s-2}$ LET H + X/Y↑(S - 2)

i) $h = \frac{3x + 4y}{x + y}$ LET H = 3*X + 4*Y / (X + Y)

j) $X = \sqrt[3]{H^2}$ LET X = (H↑2)↑1/3

k) $A = \sqrt{s(s-a)(s-b)(s-c)}$

LET A = S*(S-A)*(S-B)*(S-C)↑.5

1.12 MULTIPLE STATEMENTS

When using the WANG computer, you may write more than one statement on a single line. The statements are simply separated by colons.

This allows you to write a program using fewer lines. However, it is standard practice to write only one statement per line beacuse corrections can then be made more easily and reference to an exact statement can be made more precisely.

The three statements

```
50 LET X = 3.1416 * R↑2
60  LET Y = X + 67.65
70  LET Z = X↑2 - 4*Y↑3
```

may be written all on one line as follows:

```
50 LET X = 3.1416 *R↑2: LET Y = X + 67.65: LET Z = X↑2 -4*Y↑3
```

Remember, the w ord LET may be left out and spaces may also be left out. The above statement could be written as :

```
50X=3.1416*R↑2:Y=X+67.65:Z=X↑2-4*Y↑3
```

1.13 THE PRINT STATEMENT

```
100 PRINT A, B, C
110  PRINT D, E, F, G, H, I, J
120 PRINT D; E; F; G; H; I;  J
130 PRINT
140 PRINT A+B, A*B, A↑B,  A*B+A↑B
```

The PRINT statement in line 100 above consists of a line number, the word PRINT followed by the names of variables whose values are to be printed out. The names of the variables are separated by commas. This instruction will cause the values of A, B, and C all to be printed out on the same line.

The teletypewriter is divided into four print zones. The first three zones are 18 characters each and the fourth zone is 16 characters in length. This means there are 70 columns. However, they are numbered 0 through 69, with a new print zone beginning at 0, 18, 36 and 54.

If a PRINT statement contains more than four variable names separated by commas, four values will be printed on one line with the rest being printed on successive lines. For example, the PRINT instruction in line 110 above will cause the values of D, E, F and G to be printed out on one line and the values of H, I, and J to be printed on the next line.

A *semicolon* following a variable name in a PRINT statement signals the system to print the next value immediately following the previous one with just a space between them. This is called *packed-form*. The number of values that will be printed on a line will depend on the length of the numbers. The PRINT statement in line 120 above will cause the values of D, E, F, G, H, I, and J to be printed in packed-form.

A PRINT instruction with nothing following the word PRINT will cause the system to advance the paper one line without printing. It is actually an instruction to print a blank line. The PRINT instruction in line 130 above is such an instruction.

A PRINT statement containing BASIC expressions will cause the system to calculate and to print the values of these expressions. The PRINT statement in line number 140 above will cause the system to print the value of A + B, the value of A*B, the value of A ↑ B, the value of A*B+A↑B.

The use of this type of PRINT statement results in a shorter program, but it has the disadvantage that the values calculated are not stored for use later on in the program.

1.14 PRINTING MESSAGES

```
60  LET S = 3.1416↑(2/3)
70  LET C = S↑2
80  PRINT "VALUE OF S IS", S, "VALUE OF C IS", C
```

The PRINT statement above will cause the system to print THE VALUE OF S IS in the first print zone, the value of S in the second print zone, THE VALUE OF C IS in the third print zone and the value of C in the fourth print zone.

The system will print exactly what is between the quotation marks, including blank spaces.

The maximum length of a print zone is 18 characters. However, a single message need not be limited to 18 characters. When printing a longer message, the zone boundaries will be ignored. For example,

```
90 LET X = (A↑2 + B↑2)↑.5
100 PRINT "SQUARE ROOT OF A SQUARED PLUS B SQUARED IS", X
```

will cause the message to be printed in the first three print zones and the value of X to be printed in the fourth print zone.

The comma in a PRINT statement causes the system to print the next item in the next print zone. The semicolon causes the system to skip a space and then print the next item immediately. The system will not begin printing the next item on that same line if there is not room for it, but it will begin on the next line.

A PRINT statement message may contain any legal BASIC characters except quotation marks. One PRINT statement may contain both commas and semicolons.

Do not confuse a PRINT statement containing a message with a REM statement. The REM statement is printed in the listing of the program. The PRINT statement message is printed with the output of the program.

1.15 THE END STATEMENT

```
9999 END
```

This is an optional statement, indicating the end of the program. It does not have to be the last executable statement in a program, although it usually is. If the system encounters this statement when executing a program, it types END PROGRAM and returns control of the system to the user.

1.16 A LEVEL ONE PROGRAM

A typical level one program in BASIC consists of the following:

1. A REM statement containing the programmer's name, the level of the program and the date.
2. A REM statement indicating the title of the program.
3. LET statements assigning values to various variables.
4. One or more LET statements in which the value of a variable is calculated from the values of the other variables.
5. A PRINT statement to print the calculated values with a message to indicate the names of the variables whose values are being printed.
6. An END statement.

A sample level one program is as follows:

```
10 REM   HAROLD HARRISON   LEVEL ONE  9/10/72
20 REM   HAROLD'S SPECIAL FORMULA
30 LET A = 3.1416/184.317
40 LET B = 8.9942 * 3.94↑4
50 LET C = B + 5.67 * A
60 LET H = A↑2 + B↑2 - (A + B)/(A - B) + C
70 PRINT "A ="; A; "B ="; B; "C ="; C; "H = "; H
80 END
```

Exercises 1.16

Write a level one program for solving one of the following problems.

1. Calculate the interest on a $4500 loan if the interest rate is $9\frac{1}{2}\%$ per year and the money is borrowed for

   a) 7 months

   b) 300 days

   c) 2 years 5 months

   Use the relationship $I = PRT$

   where I is the interest

   P is the principal

   R is the interest rate per year

   T is the time in years

2. Use the Pythagorean relationship $c^2 = a^2 + b^2$ to find

   a) c when a = 123.6971 and b = 99.2233.

   b) a when c = 542.11 and b = 300.00.

   c) b when c = 1.37E11 and a = 4.59E5.

3. Find the total value of 497 nickels, 2350 dimes, 1140 quarters, and 321 half-dollars. $705.35

4. $x = \frac{3a + 4b^2 - 5c}{b}$ Calculate x when a = .0141

   b = .1996

   c = .3243

5. According to the quadratic formula, if $ax^2 + bx + c = 0$, then $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Calculate the two values of x when

a) $3x^2 + 8x + 5 = 0$

b) $5x^2 + 26x + 5 = 0$

c) $8x^2 - 2x = 1$

d) $x^2 - 3x = 2$

e) $x^2 + x - 2 = 0$

6. Calculate the distance D between the two points $P(x_1,y_1)$ and $Q(x_2,y_2)$ if $D = \sqrt{(X_2 - x_1)^2 + (y_2 - y_1)^2}$

when the coordinates of P and Q are

a) P(-17,-19) Q(32,56)

b) P(0,11.31) Q(2.84,22.45)

c) P(-123,456) Q(-789, -101)

7. $V = \frac{4}{3}\pi r^3$. Find the volume V of a sphere with radius r when r is equal to

a) 37.897

b) π

c) 1.87E13

d) 8.67E-09

e) 1,867,467

8. $A = \dfrac{1 + b}{1 + \dfrac{c}{d}}$ Calculate A when $b = 27.8613$

$c = 19.4111$

$d = .17421$

9. $F = \dfrac{x + y}{x - y} - x^2 + z^2$ Calculate F when $X = 831$

$y = 432$

$z = 1171$

10. $A = \sqrt{s(s - a)(s - b)(s - c)}$ where $s = \frac{1}{2}(a + b + c)$

Calculate A when

a) a = 5.55E17 b = 2.87E15 c = 1.87E21

b) a = 14.1414 b = 11.1897 c = 196643

c) a = 21.421 b = 21.421 c = 34.782

11. $x = \dfrac{a - b}{a + c}(a - c)$ Calculate x when $a = 1782$

$b = 1689$

$c = 4455$

12. Find the average of the numbers in the column.

a)	b)	c)
-173.461	1.87E13	.0987
89.893	2.45E27	1.11E-6
1.42E3	8.39E-3	189
	45.6	1.56E4
		90

1.17 THE START AND RUN COMMANDS

```
START
10 .....  ⎫
20 .....  ⎬ (program statements not shown)
30 .....  ⎭
RUN
```

Commands are not a part of the program itself, but they are requests to the system to perform certain functions. Commands do *not* have line numbers. They are executed *immediately* when entered.

The START command initializes the entire user program area of the system, removing all previously stored program text and variables. As soon as this command is executed, the system is ready to receive the new program.

After the new program is entered, the system will execute the program as soon as the command RUN is given.

Notice in the example above that the START and RUN commands do not have line numbers.

1.18 PROCEDURE FOR RUNNING A PROGRAM

1. Set the LINE/OFF/LOCAL switch on the 3315 teletype to LINE.
2. Pick up the telephone receiver and dial the computer number.
3. When connection to the computer is made, a beep will be heard and the red light on the acoustic coupler will go on.
4. Place the telephone receiver on the acoustic coupler.
5. Press the ESCAPE key on the teletype. The System will reply

   ```
   3300 BASIC READY
   :
   ```

   The colon indicates that control of thr system belongs to the user. A green light will then show on the acoustic coupler.
6. Type START and enter your program, pressing the RETURN key at the end of each line and after each command.
7. Type RUN. Your program will be executed and the answers printed out.
8. If you have more programs to run, type START and enter your next program. If you have no more programs to run, set LINE/OFF/LOCAL switch to OFF and replace the telephone receiver.

1.19 MAKING CORRECTIONS

If you should make a mistake while typing a line and you wish to begin the line again, press RETURN key and retype the entire line again using the same line number.

To correct the last character you just typed, press the back arrow key "⟸" This, in effect, backspaces one character each time you press the key. To erase the last three characters you just typed, press the back arrow key three times and retype these three characters.

Typing the backslash character "\" while you are typing a line will cause the entire line to be disregarded. The system will index to the next line, type a colon, and will then await further input.

After each line is entered and the RETURN key pressed, the system performs a syntax error check. If an error is found, the system will type on the next line an up arrow immediately under the location of the error, and it will print an error message number. The user then may refer to the table of error messages in the back of this text to identify the error by code number.

For example:

```
:10 LET X = (A + B)/(C - D
                          ↑ERR 05
```

Code 05 means "MISSING RIGHT PARENTHESIS".
To correct the error, retype the line correctly using the same line number.

1.20 CHANGING A PROGRAM

A program may be changed at any time, either before it has run or even after it has run.

To replace any line in a program, type the same line number again and then type whatever statement that is to replace that line in the program.

To insert a new line into a program, type a line number that is greater than the first and less than the second of the line numbers for the statements between which you wish to insert the new statement. To insert a new statement between line 30 and line 40, type a line number such as 35 and then type the statement that is to be inserted. This can be done at any time the program is being typed. The lines are executed in sequence of their line numbers, not in sequence of the order in which they are entered.

When all corrections and additions have been made, type RUN and press the RETURN key. The changed program will then be executed.

1.21 THE LIST COMMAND

```
LIST

LIST 30

LIST 40, 70
```

The LIST command will cause the system to type out the entire program, as changed or corrected, in line number sequence. This gives the programmer a chance to examine his changed or corrected program before giving the command to RUN.

If a line number follows the word LIST, just that one program line is produced.

If two line numbers separated by a comma follow the word LIST, all the program text from the first through the second line numbers, inclusive, will be listed.

The LIST command, like all other system commands, do not have line numbers.

1.22 INTERRUPTING SYSTEM PROCESSING

To interrupt the system processing during the execution of a program, the user presses the ESCAPE (ESC) key. The system will then terminate the execution of the program and will type

```
3300 BASIC READY

:
```

to return control of the system to the user.

Exercises 1.22

Go to the computer terminal, enter your level one program, and have the system execute the program.

1.23 IMMEDIATE MODE

```
PRINT (3.8945 * (1.6543 - 7 * .008764)↑3)/7.45E-3 = 2113.6194
```

If a program line does not contain a line number, the system will execute the program line immediately when the RETURN key is pressed. This permits the system to act as a conventional calculator. For example, when the immediate mode statement above is entered and the RETURN key pressed, the calculations will be performed immediately and the answer printed.

Immediate mode lines are not stored.

Use of multiple statements on one line is common in immediate mode. For example:

```
X = 2.345: Y = 1.28E6: PRINT X↑2 + Y↑2 - 5.678 * X/Y =
                                                   1.6384E+12
```

Immediate mode provides, in effect, for the entry and execution of a one-line program. Any BASIC verb except INPUT and one which references a line number such as GO TO may be used in immediate mode. These two verbs are introduced in later chapters.

Exercises 1.23

Select a problem from Exercises 1.16 and execute in immediate mode.

Chapter 2

LOOPS AND LIBRARY FUNCTIONS

WRITING A LEVEL TWO PROGRAM

2.1 THE READ AND DATA STATEMENTS

```
20 READ A, B, C
30 LET X = B↑2 - 4*A*C
40 PRINT A, B, C, X
50 DATA 1, 5, 2
60 END
```

The READ statement is used to assign to variables the values indicated in the DATA statement.

Neither the READ statement nor the DATA statement can be used without the other.

When the program above is read into computer storage, the data contained in the DATA statement is assigned to what may be thought of as a data bank. When the program is run, the first number in the DATA statement will be assigned to the first variable listed in the READ statement, the second number will be assigned to the second variable, and so on. In the above program, the value of X will be calculated and the values of A, B, C, and X printed out.

Since placement of the data into the data bank is performed when the program is *read into* computer storage instead of when the program is executed, data statements may be located anywhere in the program. However, they must provide values in the correct order for the READ statements. If several data statements are entered, they are used in order of their statement numbers.

It is common practice to put all the data statements together and place them at the end of the program just before the END statement.

Exercises 2.1

1. What would be the value of X calculated by the following program?

```
20 READ A, B, C
30 LET X = B↑2 - 4 * A * C
40 PRINT A, B, C, X
50 DATA 1, 5, 2
60 END
```

2. What would be the value of W calculated in the following program?

```
20 READ X1, X2, Y1, Y2, Z
30 LET W = Y1 * Y2 / Z - X1 + X2
40 PRINT W
50 DATA 6, 14
60 DATA 12.5, 4
70 DATA 5
80 END
```

3. A teacher calculated the value of F in the following program to be .25. A student calculated the value to be 4. Who was correct?

```
20 READ X1, X2, X3, X4
30 LET F = (X1 + X2) / (X3 + X4)
50 DATA 200, 160
40 DATA 45, 45
60 END
```

4. Are the following two programs equivalent? That is, would they both solve the same problem and give the same answer?

```
20  READ A, B, C
30  LET K = B↑2 - 4*A*C
40 DATA 197.477, 2187.999
50 DATA 44.221
35 PRINT K
60 END
```

```
20 READ B, A, C
30 LET K = B↑2 - 4*A*C
40 PRINT K
50 DATA 2187.999, 197.477, 44.221
60 END
```

5. Are the following programs equivalent?

```
20 READ L, M, N
30 LET P = L↑5 + M*N
40 PRINT P
50 DATA 187
60 DATA 357
70 DATA 123
80 END
```

```
20 READ L,M,N: P=L↑5+M*N: PRINT P: DATA 187,357,123
```

2.2 THE GO TO STATEMENT

To have a program run several times using several sets of data, all that is necessary is to furnish the additional data and insert a GO TO statement in the program.

```
20 READ A, B, C
30 LET X = B - 4 *A * C
40 PRINT "A ="; A; "B  ="; B; "C ="; C; "X ="; X
50 DATA 1, 15, 2, 2.9, 38.7, 3.3, 1.75, 111.41, 5.55
55 GO TO 20
60 END
```

When this program is executed, the first three values in the DATA statement will be assigned to A, B, and C respectively. The value of X will be calculated and the values of A, B, C, and X will be printed.

The statement GO TO 20 will then be executed. This will cause control to be transferred to the statement on line 20, which is the READ statement. The next three numbers in the data statement will then be read and assigned to A, B, and C respectively. The new value of X will be calculated and the values of A, B, C, and X printed.

The GO TO 20 statement will again be executed and procedure continued until there is no more data.

When the system encounters a READ statement and there is no more data to read, a message will be printed out and control will be returned to the user. This, therefore, will terminate execution of a program.

The data statements may contain any type of numbers, and the different types may be mixed. However, the numbers must be separated by commas.

This idea of executing a portion of a program repeatedly is called looping, and it is one of the most important concepts in programming. Other methods of establishing a loop are introduced in later chapters.

An executable statement is one that specifies the action to be performed. A nonexecutable statement is one that provides information. Therefore, a DATA statement is an example of an nonexecutable statement. Another example would be a REM statement. Examples of executable statements would be a LET statement and a GO TO statement.

The execution of a loop may always be interrupted and execution terminated at any time by pressing the ESCAPE (ESC) key.

Exercises 2.2

1. Write a program containing a loop that could be used to calculate all five answers to problem #5 in Exercises 1.16, on page 29.

2. Write a program containing a loop that could be used to calculate answers to all the parts of the following problems in Exercises 1.16.

   a) Problem #1

   b) Problem #2

   c) Problem #6

   d) Problem #7

   e) Problem # 10

2.3 THE INPUT STATEMENT

A program may be written in such a way that the data is ~~input~~ entered during the execution of the program. This is common in programs for playing games such as blackjack or dice with the computer. It is also useful when a programmer prepares and saves a program on paper tape. When the user then runs the program, he supplies the necessary data at the proper place in the program.

For example, consider a program for calculating the roots of a quadratic equation. When the user runs this program, he must supply the values of the coefficients, A, B, and C. The program will contain an INPUT statement such as

```
50 INPUT A, B, C
```

before the first program line which requires these values. When the system encounters this statement, it types the word INPUT and a colon on the following line, and waits for the user to supply the three numbers.

```
INPUT
:
```

The user will then type in the three numbers, in correct order. The system will then proceed to the next program line.

It is a good practice to identify the requested values by a preceeding PRINT statement so that the user will know the variables for which he is to supply values and the exact order in which they are to be entered. For example:

```
20 PRINT "ENTER VALUES FOR COEFFICIENTS A, B, and C"
30 INPUT A, B, C
40 LET X1 = ((-B + B↑2 - 4*A*C)↑.5)/(2*A)
50 LET X2 = ((-B - B↑2 - 4*A*C)↑.5)/(2*A)
60 PRINT A; B; C; X1; X2
70 END
```

The system will print out the descriptive message before printing the word INPUT. It will print:

```
ENTER VALUES FOR COEFFICIENTS A, B, and C
INPUT
:
```

If there is a system-detected error in the entered data, the numbers must be re-entered beginning with the erroneous number. The numbers which precede the error are accepted.

A user may terminate an input sequence without supplying all the required input values by simply typing a carriage return. The system will procede to the next program statement.

Exercises 3.3

1. Write a program using an INPUT statement that would calculate the answers to problem #5 in Exercises 1.16 with one set of data to be supplied by the user at the time of program execution.

2. Write a program using an INPUT statement that would calculate the answer to problem #5 in Exercises 1.16 with several sets of data to be supplied by the user at the time of program execution.

3. Write a program using an INPUT statement for calculating the answers to the following problems in Exercises 1.16 with several sets of data to be supplied by the user at the time of program execution.

3.4 SAVING A PROGRAM

SAVE

The SAVE command will cause all program lines to be punched on the paper tape at the user's teletype terminal. Five inches of leader and five inches of trailer code will be punched before and after the program text. The user must manually turn on the teletype paper tape punch during this operation and turn it off when completed.

Like all other system commands, the SAVE command will have no line number.

Methods for saving a program on casette or disk are introduced in later chapters.

Exercises 3.4

Enter a level one program into the computer system. Use the SAVE command to have it saved on paper tape.

3.5 LOADING A SAVED PROGRAM

LOAD

When the LOAD command is used, the program that is currently on the paper tape reader at the user's terminal will be loaded into the computer and added on to any program that is in the system's memory. The system will automatically start the tape motion and read the tape. This command is used to enter new programs and to enter additions to programs already entered.

Having a START command precede the LOAD command will erase any previous program text before the new program is entered.

Data that has been saved on paper tape can be entered into a program by use of the LOAD command.

Methods for loading a program from a cassette or disk are introduced in later chapters.

Exercises 3.5

1. Take the paper tape you prepared in Exercise 3.4 and use the LOAD command to load the program into the system. Then have the program executed.
2. Select one of the problems in Exercises 1.16. Prepare a program to solve the problem with the data to be entered by use of paper tape and a LOAD command. Your instructor will furnish you with the paper tape containing the data.

2.6 THE PRINT USING AND IMAGE STATEMENTS

```
110 PRINT USING 130, L, W, A
130 % LENGTH = ###.##  WIDTH = ##.##  AREA = ####.##
```

The PRINT USING statement lists the variables whose values are to be printed out. It also contains a line number the word USING which refers to an image statement.

The image statement, identified by the % sign, indicates what letters or words are to be printed out and indicates the format in which the values are to be printed.

The PRINT USING statement above indicates that the values of L, W, and A are to be printed and they are to be printed in the format indicated by the image statement on line 130. The image statement on line 130 indicates the letters or words that are to be printed and indicates by the use of #'s where the digits are to be printed. It also indicates the location of the decimal point if there is to be one.

If the values of L, W, and A in the example above are 11.12, 10.00, and 111.20 respectively, the printout would appear as:

```
LENGTH =  11.12  WIDTH = 10.00  AREA =  111.20
```

Notice there are no quotation marks around the alphanumeric characters in the image statement that are to be printed out.

If a number that contains a decimal point is to be printed out but the format in the image statement does not contain a decimal point, the number will be printed out truncated. That is, the decimal part is simply dropped. It is not rounded. Leading blanks are always inserted and the value is printed right justified.

If the format in the image statement contains a decimal point, the value is printed with the number of decimal places as indicated by the format, with the decimal part of the number being truncated or filled with zeros to produce the desired number of decimal places.

If the length of the value is greater than the format specification, #'s will be printed out by the system to indicate such an error. For example, if the value 1234.56 were to be printed out according to the format ##.##, the error would be indicated by a printout consisting of #####.

The four characters !!!! are used to indicate the format for exponent form. The number 123.45E-19 to be printed out according to the format ###.###!!!! would appear as 123.450E-19.

If the format specification contains neither a positive nor a negative sign, a negative number is printed out with its negative sign in front of the number and the length of the format specification is automaticially increased by one.

If the format specification contains a plus sign, the true sign of the number will be printed in front of the number. For example, the format specification +###.## would cause the number 156.78 to be printed as +156.78 and the negative number -56.78 to be printed as -56.78.

If the format specification contains a negative sign, a blank for positive numbers and a minus sign for for negative numbers will be printed in front of the numbers.

If there are more values to be printed out than there are formats specified in the image statement, the image statement is reused from the beginning for the remaining values. For example,

```
10 PRINT USING 20, A, B, C, D
20 % ##.##
```

Only one number will be printed per line if the variable names are separated by commas. The comma will cause the carriage return to be executed.

The printout would be:

```
23.34
34.45
56.67
78.78
```

If the variable names are separated by <u>semicolons</u>, the carraige return is suppressed and all the values will be printed on one line. The output would be:

```
23.45    34.45    56.67    78.78
```

Format specifications, including sign, decimal point, and !!!! cannot exceed 24 characters.

Exercises 2.6

Select one of the problems from Exercises 1.16. Write a program for solving all parts of the problem, using a loop. Use a PRINT USING and an image statement to cause the values of the main variables and the values of the calculated answers to be printed out, together with a word or descriptive phrase to indentify each of the values being printed.

2.7 LIBRARY FUNCTIONS

A computer can perform simple arithmetic such as addition, subtraction, multiplication and division. It can also perform simple logical operations that require that it distinguish between 0 and 1. If a computer is to perform complicated mathematical functions such as finding the square root of a number or calculating the sine of an angle, these mathematical functions must first be defined in terms of simple addition, subtraction, multipication and division.

If a programmer had to write a subprogram for each mathematical function every time he was to use it in his program, programming would be much more difficult than it actually is. Instead, subprograms have already been written for the most common mathematical functions, and these subprograms are stored either as a part of the BASIC compiler, or are readily available on a disk storage facility.

To make use of these stored programs, called *library functions*, all the programmer has to do is to write the name of the function followed by parentheses enclosing the expression for which the function is to be computed. This expression may be a constant, a variable, or a BASIC expression. It may also be *another function*.

The expression enclosed in the parentheses is called the argument of the function, and it may be any valid expression for which a numerical value can be calculated, except as noted in the explanations of the following eleven BASIC library functions.

The absolute value function.	ABS( )
The square root function.	SQR( )
The sine function.	SIN( )
The cosine function.	COS( )
The tangent function.	TAN( )
The arc tangent function.	ATN( )
The algebraic sign function.	SGN( )
The natural exponent function.	EXP( )
The natural logarithmic function.	LOG( )
The largest integer function.	INT( )
The random number generator.	RND( )

2.8 THE ABSOLUTE VALUE FUNCTION

```
100 LET Y = ABS(-16.789)
110 LET Z = ABS(X)
120 LET W = ABS(B↑2 - 4*A*C)
```

The absolute value of a positive number is the number itself. The absolute value of a negative number is the corresponding positive number. The absolute value of zero is zero. In the example above, the value of Y is 16.789.

Exercises 2.8

Calculate the value or values of X in each of the following programs.

1.
```
10 LET X = ABS(-23.4 + 10.0)
20 PRINT X
30 END
```

2.
```
10 LET A = 25.0: LET B = 5.2
20 LET X = ABS(A - B↑2)
40 PRINT X
50 END
```

3.
```
10 READ A, B
20 LET X = (ABS(A + B))↑.5
30 PRINT X
40 GO TO 10
50 DATA 20, -4, -20, -5, 30, 6,60, -11
60 END
```

4.
```
110 READ K,W: X = ABS(K) + ABS(W): PRINT K,W,X
120 DATA -1492, -611: END
```

5.
```
10 READ C,D: X = ABS(ABS(C) - ABS(D)):PRINT USING 40, C, D, X
40 % C IS +#.###  D IS ###.#  X IS #.###
50 GO TO 10
60 DATA -14, 22, 36, 36, 125, -100
70 END
```

2.9 THE SQUARE ROOT FUNCTION

```
130 LET C = SQR(79.15)
140 LET D = SQR(X)
150 LET E = SQR(A↑2 + B↑2)
160 LET F = SQR(ABS(B↑2 - 4*A*C))
```

The argument of the square root function must <u>not</u> be negative, as the resulting number would be imaginary. If the value of the argument of the square root function is negative, the system will print an error message.

In line 160 above, the argument is the absolute value function. In this case, it is impossible for the argument of the square root function to be negative.

The following three statements are equivalent:

```
140 LET D = SQR(X)
140 LET D = X↑.5
140 LET D = X↑(1/2)
```

However, the calculations are done with the greatest computer efficiency with the first statement and with the least computer efficiency with the last statement.

There are no library functions for roots other than square roots. The principal cube root, fourth root, fifth root, etc. are calculated by use of fractional or decimal exponents. For example:

```
170 LET G = X↑(1/3)
180 LET H = X↑.25
190 LET H = X↑(1/4)
200 LET I = X↑.2
210 LET I = X↑(1/5)
```

Exercises 2.9

Calculate the value or values of Y in each of the following programs.

1.
```
10 Y = SQR(34 + 66): PRINT Y
```

2.
```
10 A = 3: B = 4: Y = SQR(A↑2 + B↑2): PRINT Y
```

3.
```
10 READ A, B
20 LET Y = SQR(ABS(A↑2 - B↑2))
30 PRINT A, B, Y
40 DATA 10, 6, 8, 10, 7, -7
50 END
```

4.
```
10 Y = SQR(SQR(144) + SQR(16))
20 PRINT Y
```

5.
```
10 Y = (SQR(9))↑3: PRINT Y
```

Write and execute a BASIC program in IMMEDIATE MODE to solve each of the following for W.

6. $W = \sqrt{|(-3.197)^3|}$

7. $\left|\sqrt[3]{-197.32}\right| + \sqrt{|-197.32|}$

8. $\sqrt{\dfrac{(87.94)(37.557)}{\sqrt{-3.9}}}$

9. $(-176.176)^{\frac{5}{3}} + |(-83.2)^3|$

10. $\sqrt{8391} \div \sqrt[3]{1236} - \sqrt[4]{1499}$

11. Write a program for solving for all values of K.

    a) $K^2 = 1592.43$ b) $K^2 - 897.11 = 0$ c) $K^2 - 37.44 = 13.97$

2.10 THE SINE, COSINE, AND TANGENT FUNCTIONS

```
210 LET S = SIN(1.5708)
220 LET C = COS(37/57.2958)
230 LET T = TAN(X + .5236)
```

The argument of the SIN, COS, and TAN library functions must be expressed in radians. The magnitude of the argument cannot exceed 1E8.

A measure of one radian is equivalent to a measure of 180/π in degrees or about 57.2958 degrees. If a measure in degrees is multiplied by π/180 or divided by 57.2958, the result is an equivalent measure in radians.

An angle measure of one minute (1') is 1/60 of a degree. An angle measure of one second (1'') is 1/60 of a minute or 1/3600 of a degree. To convert an angle measure given in degrees, minutes, and seconds to radians, first express the measure in degrees, and then convert to radians. For example, to convert a measure of 37° 14' 55'' to radians:

```
250 LET D = 37 + 14/60 + 55/3600
260 LET R = D/57.2958
```

or

```
250 LET R = (37 + 14/60 + 55/3600)/57.2958
```

To use π to more than four decimal places, round off the following value of π to the number of decimal places desired:

$$\pi \approx 3.14159265358979321$$

There is no library function for the secant, cosecant, and cotangent functions. However, these three functions can be easily programmed in BASIC by use of the following relationships:

$$\text{cosecant } x = \frac{1}{\text{sine } x}$$

$$\text{secant } x = \frac{1}{\text{cosine } x}$$

$$\text{cotangent } x = \frac{1}{\text{tangent } x}$$

Exercises 2.10

Write a program for calculating the value of the indicated variable in each of the following programs.

1. $Q = \sin 39^\circ + \cos 63^\circ - \tan 44^\circ$

2. $W = \dfrac{\sin 123^\circ}{\cos 66^\circ}$

3. $E = \cos 89^\circ\ 13'$

4. $R = \text{Tan } 241^\circ\ 43'\ 10''$

5. $T = \sqrt{\sin^2 15^\circ + \cos^2 35^\circ}$

6. $T = |\sin 253^\circ| + |\tan 300^\circ|$

7. $Y = \sqrt{|\sin 99^\circ + \cos 99^\circ|}$

8. $U = \sec 25^\circ - \csc 14^\circ$

9. $I = \sin 47^\circ + \cot 47^\circ$

10. $P = \sqrt{(23.6)^2 + (11.2)^2 - 2(23.6)(11.2)\cos 58^\circ}$

11. The Law of Cosines states: $\dfrac{a}{\sin A} = \dfrac{b}{\sin B} = \dfrac{c}{\sin C}$

    Write a BASIC program to find

    a) b when a = 37.6, A = 77° 8', B = 41° 17'

    b) a when A = 29° 37', B = 40° 58', c = 152.4

    c) c when a = 35.7, B = 41° 12', C = 61° 40'

12. If $a^2 = b^2 + c^2 - 2bc\cos A$ find c when a = 13.43, b = 17.77 and C = 62° 12'.

2.11 THE ARCTANGENT FUNCTION

```
270 LET A = ATAN(1.0000)
280 LET B = ATAN(X)
```

If the value of the tangent of an angle is known or can be calculated, the angle can be found by use of the ATAN function.

If tan X = y, then x = arctan y.

Therefore, arc tan 1 means the angle whose tangent is 1. Arc tan 1 = 45°, 225°, and the other angles whose tangent is 1.

The statement in line 280 above will cause the system to compute the principal value of arctangent x.

The ATAN function results in an answer that is an angleexpressed in radians. Multiplying the radian measure by 180/π or 57.2958 will result in an equivalent measure in degrees.

For example, the following statement may be used to find the angle in degrees whose tangent is 1.0000:

```
150 LET X = (ATAN(1.0000))*57.2958
```

There are no BASIC library functions for the arcsine and arccosine functions. Methods for calculating the angle when its sine or cosine are known are introduced in a later chapter.

Exercises 2.11

Write a BASIC statement for solving each of the following equations for the variable given. Answer is to be in radians.

1. $a = \arctan 1$
2. $b = \arctan (-1.200)$
3. $c = \arctan \frac{5}{6}$
4. $d = \arctan \frac{\sqrt{3}}{3}$

Write a BASIC statement for solving each of the following equations so that the answer will be in degrees.

5. $e = \arctan (-1)$
6. $f = \text{arc tan } \frac{1}{5}$
7. $g = \arctan (\sin 90°)$
8. $\tan x = 3.7152$

Using the ATAN function, calculate the value of X in degrees if:

9. $\sin x = .3090$ and $\cos x = .9511$
10. $\sin x = .9877$ and $\cos x = .1564$

TRIGONOMETRIC IDENTITIES

1. cosecant $x = \dfrac{1}{\text{sine } x}$

2. secant $x = \dfrac{1}{\text{cosine } x}$

3. cotangent $x = \dfrac{1}{\text{tangent } x}$

4. If $\angle A + \angle B = 90°$,

   $\sin A = \cos B$

   $\tan A = \cot B$

   $\sec A = \csc B$

5. $\sin (180° - x) = \sin x$

   $\cos (180° - x) = -\cos x$

   $\tan (180° - x) = -\cot x$

6. $\sin (90° + x) = \cos x$

   $\cos (90° + x) = -\sin x$

   $\tan (90° + x) = -\cot x$

7. Pythagorean identities.

   $\sin^2 x + \cos^2 x = 1$

   $1 + \cot^2 x = \csc^2 x$

   $\tan^2 x + 1 = \sec^2 x$

8. Quotient identities.

   $\tan x = \dfrac{\sin x}{\cos x}$

   $\cot x = \dfrac{\cos x}{\sin x}$

9. Law of Sines.

   $\dfrac{\text{side } a}{\sin A} = \dfrac{\text{side } b}{\sin B} = \dfrac{\text{side } c}{\sin C}$

10. Law of Cosines.

   $a^2 = b^2 + c^2 - 2bc \cos A$

   $b^2 = a^2 + c^2 - 2ac \cos B$

   $c^2 = a^2 + b^2 - 2ac \cos C$

   $\cos A = \dfrac{b^2 + c^2 - a^2}{2bc}$

   $\cos B = \dfrac{a^2 + c^2 - b^2}{2ac}$

   $\cos C = \dfrac{b^2 + a^2 - c^2}{2ab}$

11. Polar coordinates.

   To convert $P(x,y)$ to $P(r,\theta)$

   $r = \sqrt{x^2 + y^2}$

   $\tan \theta = \dfrac{y}{x}$ or $\theta = \arctan \dfrac{y}{x}$

   To convert $P(r,\theta)$ to $P(x,y)$

   $x = r \cos \theta$

   $y = r \sin \theta$

12. Distance between point

   $A(x_1,y_1)$ and $B(x_2,y_2)$

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

2.12 THE ALGEBRAIC SIGN FUNCTION

```
290 LET K = SGN(-1.8775)
300 LET L = SGN(X)
310 LET P = SGN(COS(Y))
```

If the argument of the algebraic sign function is equal to zero, the value of the sign function is zero. If the argument is equal to a positive number, the value of the function is +1. If the argument is equal to a negative number, the value of the function is -1.

EXERCISES 2.12

Calculate the value of the variable in each of the following:

1. `10 LET W = 3.14 + SGN(-1.4)`

2. `20 LET K = SGN(√137.423)`

3. `30 LET V = SGN(A↑2 - A*A)`

4. `40 LET M = SGN(SQR(121) - 14)`

5. `50 LET X = -SGN(SQR(ABS(-49)))`

2.13 THE NATURAL EXPONENT FUNCTION

```
320 LET Q = EXP(1.25)

330 LET R = EXP(X)

340 LET S = EXP(SQR(W))
```

The EXP function is used to calculate the value of the number e raised to the power designated by the argument of the function.

The number e is an irrational number whose value is approximately 2.71828. The value of e^x is the infinite sum $1 + x + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots$

Statement 320 above will calculate $e^{1.25}$ and assign this value to Q.

Exercises 2.13

Write and execute in IMMEDIATE MODE a program to calculate

1. $e^{1.5}$ 2. $e^{-0.02} + e^{2.7}$ 3. $e^{2.2} - e^{-0.7}$

Write a program to calculate all the values of y in each of the following, when x takes on the values indicated.

4. $y = 3e^{-x}$ $x = 1.6, \quad 3.2, \quad 11.6$

5. $y = \frac{1}{3}e^{3x} \div \frac{1}{2}e^{2x}$ $x = 7.3, \quad 5.5, \quad -2$

6. $y = -3e^{-2x}$ $x = 0, \quad -.1, \quad .14$

7. $y = \dfrac{e^{x} - e^{-x}}{2}$ $x = -1, \ -.95, \ 1, \ 2, \ 3$

This is called the hyperbolic sine of x (sinh x).

8. $y = \dfrac{e^{x} - e^{-x}}{2}$ $x = -1.5$, .66. 3.84

This is called the hyperbolic cosine of x (cosh x).

9. $y = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ $x = 4.4, \ 1.1, \ -3.3$

This is called the hyperbolic tangent of x (tanh x).

10. $y = e^{-x^{2}}$ $x = -.55, \ -.97, \ 0.01, \ 1.2$

2.14 THE NATURAL LOGARITHMIC FUNCTION

```
50 LET T = LOG(325.188)
60 LET U = LOG(X)
70 LET V = LOG(EXP(3.5))
```

The natural logarithmic function calculates the natural logarithm (to the base e) of the absolute value of the argument.

If $e^{0.5} = 1.64873$, then $\log_e 1.64873 = 0.5$.

Therefore, the statement

```
40 Y = LOG(EXP(0.5))
```

will assign the value of 0.5 to Y.

Exercises 2.14

Write a program to calculate all the value of Y in each of the following equations, using the given values of x.

1. $y = |\log_e x|$ $\qquad x = .44,\ 1.1$

2. $y = \frac{1}{2}\log_e x^2$ $\qquad x = -.11,\ 1.7,\ 2$

3. $Y = \log_e|x| + \log_e\sqrt{x} - x\log_e x$ $\qquad x = 43.44,\ 456.77$

2.15 THE GREATEST INTEGER FUNCTION

```
80 LET A = INT(11.456)
90 LET B = INT(H)
95 LET C = INT(H + .5)
```

The greatest integer function calculates the greatest integer not greater than the value of the argument.

The statement on line 80 will assign the value 11 to A. If H in line 90 has a value of -4.56, the integer -5 will be assigned to B, since -5 is the greatest integer not greater than -4.56. Remember, -4.56 is greater than -5 since a number is greater than any number to its left on the real number line.

One important use of this function is for rounding numbers to any desired place. For example, line 95 above will cause the value of H to be rounded to the nearest integer:

Value of H	Value of H + 0.5	Value of INT(H + 0.5)
13.7294	14.2294	14
8.5000	9.0000	9
0.0099	0.5099	0
-3.4999	-2.9999	-3
-1.5000	-1.0000	-1

Notice that 8.5000 rounds to 9.0 while -1.5 rounds to -1.

Exercises 2.15

Calculate the value of the variable in each of the following programs.

1.

```
10 READ H
20 X = 10 * INT(H + .5)
30 PRINT X
40 GO TO 10
50 DATA 3.499, -3.499, 7.50, -7.50
60 END
```

2.

```
10 Y = INT(-13.987) * INT(9.56); PRINT Y
```

3.

```
20 W = SQR(INT(100.43)) + SQR(INT(ABS(-9.5)))
```

2.16 RANDOM NUMBER GENERATOR

```
40 LET H = RND(X)
```

The random number generator will generate a random number between 0 and 1 each time the statement is encountered in a program. The numbers produced are decimals .

BASIC requires that a variable name be shown within the parentheses following RND. Any name can be used, and X is commonly used, as shown in line 40 above.

To create random numbers that are <u>integers</u> greater than zero, make use of both the RND and the INT functions:

```
50 LET V = INT(10 * RND(X))
```

Multiplying RND(X) by a power of 10 and using the INT function as shown, random integers can be created with the number of digits desired.

When a program containing the RND function is run again, the computer will generate the same set of random numbers. To ensure that a <u>different</u> set of random numbers will be produced each time the program is run, the statement RANDOM is placed at the beginning of the program. For example,

```
10 RANDOM
20 PRINT RND(X)
```

2.17 WRITING A LEVEL TWO PROGRAM

Specifications for a typical level two program are:

1. A REM statement giving your name and the level of your program.
2. A REM statement giving the title of your program.
3. One or more loops using READ, DATA, and GO TO statements so that the program will be executed repeatedly until all the data has been used.
4. A printout utilizing one or more PRINT USING statements with messages identifying the numbers being printed out.

For example:

```
10 REM  SPIRO NIXON  LEVEL TWO  12/7/72
20 REM  RECIPROCAL TRIGONOMETRIC FUNCTIONS.
30 REM  DATA IS IN DEGREES
40 READ D
50 LET R = D / 57.2958
60 REM C1, S, AND C2 ARE THE COSECANT, SECANT, AND COTANGENT
70 LET C1 = 1/SIN(R)
80 LET S = 1/COS(R)
90 LET C2 = 1/TAN(R)
100 PRINT USING 120, C1, S, C2
120 % CSC IS ###.##    SEC IS ###.##    COT IS ###.##
130 GO TO 40
140 DATA 37, 134, 189, 230, 298, 310
150 END
```

Exercises 2.17

Business Problems

1. Compound Interest

If p dollars are invested at a rate of r% per period and interest is compounded at the end of each of n periods, the amount to which the p dollars will increase by the end of n periods can be calculated by use of the formula $A = p(1 + r)^n$

If the interest rate is given as an annual rate, it can be converted to the rate per period by

$$\text{rate per period} = (\text{annual rate})\frac{(\text{length of period in months})}{12}$$

$$= (\text{annual rate})\frac{(\text{length of period in days})}{365}$$

Calculate the amount to which the principal will increase given the principal, the annual interest rate, the length of a period, and the number of periods as follows:

Principal	Annual interest rate	Compounded	Time
\$1,340	6%	quarterly	2 years
\$12,400	9.5%	semi-annual	7 years
\$25,530	4.25%	monthly	45 months
\$10,000	7.5%	annually	16 years
\$10,000	7.5%	semi-annual	16 years
\$10,000	7.5%	quarterly	16 years
\$10,000	7.5%	monthly	16 years
\$10,000	7.5%	daily	16 years

Calculate the interest earned in each of these examples.

2. Salesman's Commission

Calculate a salesman's commission if his commission rate is 4% of sales for the first $8,000 of sales and 5% of his sales above $8,000 for the period. His sales for the period were: $940, $810, $1152, $1060, $1090, $905, $755, $1400, $832, $999

3. Gross Pay

Calculate gross pay, given h, the number of regular hours worked;
r, the rate of pay per hour;
v, the number of hours of overtime worked;

if gross pay = r(h + 1.5v)

In the output, print the employee number, and the values of h, r, v, and gross pay.

Employee number	h	r	v
16972	40	$4.25	4
16883	40	$6.19	8
16999	38	$7.11	0
17008	40	$6.80	3
17011	32	$5.20	0

4. Updating Charge Accounts

Calculate charge account balances at the end of the month when given B, the balance at the beginnihg of the month; C, the charges made during the month; P, the payments made during the month; and R, the returns made during the month.

In the printout, print the customer number, each of the items B, C, P, R, and the balance at the end of the month.

B	C	P	R
$0.00	$17.88	$10.00	$0.00
$43.17	$35.00	$15.00	$0.00
$89.90	$0.00	$50.00	$0.00
$745.00	$15.75	$125.00	$5.75
$100.50	$0.00	$10.00	$0.00

5. Income Tax

Calculate taxable income (T) given gross income (G) and number of exemptions (E) if

$$T = G - (\$625)(E) - (10\%)(G)$$

Then calculate federal income tax if the tax on taxable income between $10,000 and $12,000 is $2190 plus 32% of the amount over $12,000.

Calculate the New York State income tax if the tax on taxable incomes between $11,000 and $13,000 is $520 plus 8% of the amount over $11,000.

G = $12,800	E = 1	C = $15,000	E = 3
G = $12.600	E = 0	G = $16,123	E = 4

Algebra Problems

6. Evaluating Polynomials

Evaluate the polynomial

$y = 6.179x^5 - 0.193x^4 + 1.492x^3 + 1.776x^2 + 7.250x + 1.116$

for the values of x given below.

The computations can be done with greatest computer efficiency if the operations are rearranged as follows:

$x = (((((6.179x - 0.193)x + 1.492)x + 1.776)x + 7.250)x + 1.11$

This is called the nested method and is equivalent to evaluating a polynomial by synthetic division.

$x = 1.7, \quad 3.42, \quad 6.9, \quad 11, \quad 21.5$

7. Equation of a line when two points are given.

The graph of the equation y = (slope)(x) + (y-intercept) is a straight line, where the slope and y-intercept are given. If the coordinates (x_1, y_1) and (x_2, y_2) of two points on the line are given, the slope and y-intercept can be calculated from the relationships:

$$\text{slope} = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{and y-intercept} = y_2 - (\text{slope})(x_2)$$

Write the equation for the line containing the two given points.

(8,11) and (-3,2)   (-9,-2) and (5,11)

(32,64) and (-18,-9)   (1.86, 9.37) and (8.91, 11.23)

(9.99, 8.76) and (-14.53, -16.76)

8. Point of intersection of two lines.

Utilizing the concepts described in problem #7, calculate the coordinates of the point of intersection of two lines, given the coordinates of two points on each line.

(8.1, 9.2), (-3.2, 2.1) and (7.3, 5.5), (9.4, 3.3)

(-16.6, 11.2), (13.4, 4.45) and (19.18, 17.77), (15.5, 12.2)

9. Quadratic formula.

If $ax^2 + bx + c = 0$, where a, b, and c are real numbers (that is, they are not imaginary) and $a \neq 0$, then

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Calculate the roots of each of the following quadratic equations.

$$x^2 - 4.7x + 4.48 = 0$$

$$2.1x^2 + 7.34x + 2.78 = 0$$

$$6.65 = 3.25x^2 + 5.89x$$

$$x^2 + 2.85x - 4.73 = 0$$

$$236x^2 - 1149\,x = 752$$

$$3586.15x^2 - 3432.77x = 2158.14$$

10. <u>Solution of sets of linear equations in two variables.</u>

$$\text{If } a_1x + b_1y = c_1 \text{ and } a_2x + b_2y = c_2$$

then

$$\text{then } x = \frac{c_1b_2 - c_2b_1}{a_1b_2 - a_2b_1} \qquad \text{and } y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1}$$

Calculate the roots x and y for each of the following systems of equations.

$$\begin{aligned} 6x + 10y &= 7 \\ 15x - 4y &= 3 \end{aligned} \qquad \begin{aligned} 3x + 7y &= 4 \\ 4x - 3y &= -7 \end{aligned} \qquad \begin{aligned} \tfrac{1}{2}x + \tfrac{1}{3}y &= 8 \\ \tfrac{3}{2}x - \tfrac{4}{3}y &= -4 \end{aligned}$$

$$\begin{aligned} x - y &= 200 \\ .03x - .04y &= 2 \end{aligned} \qquad \begin{aligned} .02x &= .03\,y + 1 \\ x + y &= 800 \end{aligned}$$

11. Solution of sets of linear equations in three variables.

$$\begin{aligned} \text{If } a_1x + b_1y + c_1z &= d_1 \\ a_2x + b_2y + c_2z &= d_2 \\ a_3x + b_3y + c_3z &= d_3 \end{aligned}$$

then $x = \dfrac{D_x}{D}$ $\quad y = \dfrac{D_y}{D}$ $\quad z = \dfrac{D_z}{D}$

where

$$D = a_1(b_2c_3 - b_3c_2) - a_2(b_1c_3 - b_3c_1) + a_3(b_1c_2 - b_2c_1)$$

$$D_x = d_1(b_2c_3 - b_3c_2) - d_2(b_1c_3 - b_3c_1) + d_3(b_1c_2 - b_2c_1)$$

$$D_y = a_1(d_1c_2 - d_2c_1) - a_2(d_1c_3 - d_3c_1) + a_3(d_1c_2 - d_2c_1)$$

$$D_z = a_1(b_1d_2 - b_2d_1) - a_2(b_1d_3 - b_3d_1) + a_3(b_1d_2 - b_2d_1)$$

Calculate the values of x, y, and z in each of the following systems of equations.

$$\begin{aligned} 2x + 3y + 2z &= 11 \\ x - 3y + 3z &= 5 \\ 5x + y - 4z &= 3 \end{aligned} \qquad \begin{aligned} 3x - 2y - 3z &= -1 \\ 6x + y - 2z &= 7 \\ 9x + 3y + 4z &= 9 \end{aligned}$$

$$\begin{aligned} 4x + y &= 5 \\ 3x + 2z &= 0 \\ 2y - 3z &= 3 \end{aligned}$$

11. Trigonometric functions.

Calculate the value of the sine, cosine, tangent, cosecant, secant, and cotangent of each of the following angles. Arrange the output to appear in six columns, each with an appropriate column heading.

Data set A: 37°, 165°, 183°, 271°, 472°, 329°

Data set BA 55° 14', 32° 27', 189° 50'

272° 15' 15'', 181° 22' 14''

12. Law of Cosines.

$$a^2 = b^2 \div c^2 - 2\ bc\ \cos A$$

$$b^2 = a^2 + c^2 - 2\ ac\ \cos B$$

$$c^2 = a^2 \div b^2 - 2\ ab\ \cos C$$

Calculate the length of the third side in each of the following triangles.

	a	b	∠C
Data set A:	15.171	19.422	74°
	23.991	33.182	331182
	1.777	2.665	33°
Data set B:	123.45	122.66	17° 16'
	456.78	329.44	45° 45'
	876.54	543.32	33° 33'

13. Law of Sines.

$$\frac{\text{side } a}{\sin A} = \frac{\text{side } b}{\sin B} = \frac{\text{side } c}{\sin C}$$

Calculate the length of side a in each of the following triangles.

Data set A:

∠A = 31°	∠B = 29°	c = 21.873
∠A = 30°	∠B = 30°	c = 17.157
∠A = 57°	∠B = 48°	c = 74.491
∠A = 49°	∠B = 61°	c = 63.833

Data set B:

∠A = 15° 16"	∠B = 30° 24"	c = 192.91
∠A = 29° 13"	∠C = 23° 14"	b = 115.17
∠A = 57° 25"	∠C = 47° 44"	b = 1774.11
∠A = 48° 48"	∠B = 61° 51"	c = 18883.83

14. Area of a triangle.

Calculate the area of each of the following triangles using the formula

$$A = \sqrt{s(s - a)(s - b)(s - c)}$$

where a, b, and c are the lengths of the sides of the triangle and s is one-half the perimeter.

a	b	c
11	21	29
123	101	267
1492	1350	2500
11.15	15.91	22.22
.013	.019	.019
1.67E14	2.45E15	1.88E14
1.58E-5	1.06E-6	1.60E-5

Statistics Problems

15. Standard Deviation.

Standard deviation (SD) is an average of the degree to which a set of scores deviates from their mean. It is a measure of variability. To calculate SD:

subtract the mean from each score to obtain the deviation (D);

square each of these deviations to obtain D^2 ;

add these squares to obtain ΣD^2 ;

divide this sum by the number of scores (N);

take the square root of the result.

$$SD = \sqrt{\frac{D^2}{N}}$$

A second method for calculating SD is:

square each score (X^2); count the number of scores (N);

sum the scores (ΣX); sum the squares of the scores (ΣX^2)

find mean of scores ($\frac{\Sigma X}{N}$); find mean of squares ($\frac{\Sigma X^2}{N}$);

subtract $\frac{\Sigma X^2}{N} - \left(\frac{\Sigma X}{N}\right)^2$; take square root of result.

$$SD = \sqrt{\frac{\Sigma X^2}{N} - \left(\frac{\Sigma X}{N}\right)^2}$$

Calculate the standard deviation of each of the following sets of scores.

set #1	set #2	set #3
89	48.1	65.11
72	65.5	66.23
79	72.7	67.87
91	85.9	70.42
80	91.6	91.99
79	100.0	92.21
75	80.2	93.35
91	97.6	94.86

16. Correlation between two sets of scores.

A coefficient of correlation expresses the degree of relationship between two sets of s cores, by numbers ranging from +1.00 for a perfect positive correlation to -1.00 for a perfect negative correlation. A correlation of 0.00 indicates no relationship between the two sets of scores.

One method of calculating correlation coefficient if the product-moment method which is as follows:

Call one set of scores X scores and the other set of scores the Y scores.

square each X score, giving X^2;

square each Y score, giving Y^2;

calculate the product XY for each pair of scores;

find ΣX, the sum of all the X scores;

find ΣY, the sum of all the Y scores;

find N, the number of scores in one set of scores;

find ΣX^2, the sum of the squares of each X score;

find ΣY^2, the sum of the squares of each Y score;

find ΣXY, the sum of the products XY;

substitute in the following equation:

$$r = \frac{\frac{\Sigma XY}{N} - \left(\frac{\Sigma X}{N}\right)\left(\frac{\Sigma Y}{N}\right)}{\sqrt{\frac{\Sigma X^2}{N} - \left(\frac{\Sigma X}{N}\right)^2}\ \sqrt{\frac{\Sigma Y^2}{N} - \left(\frac{\Sigma Y}{N}\right)^2}}$$

The formula for the correlation coefficient may also be written:

$$r = \frac{\frac{\Sigma XY}{N} - (M_x)(M_y)}{(SD_x)(SD_y)}$$

in which M_x = mean of the X scores

M_y = mean of the Y scores

SD_x = standard deviation of the X scores

SD_y = standard deviation of the Y scores

Calculate the product-moment correlation coefficient between the two given sets of scores in each of the following:

X	YY	X	Y	X	Y
89	92	528.1	214.4	165	412
72	71	577.2	275.6	172	410
88	65	531.3	230.7	183	391
91	90	599.2	291.5	192	388
100	92	398.2	180.0	205	382
88	88	450.0	200.7	245	375
65	75	468.5	214.6	262	356
78	85	591.4	289.5	280	351
42	65	523.5	210.3	283	366
100	85	555.5	222.2	290	357

Chapter 3

CONDITIONAL TRANSFER AND FLOWCHARTING

Writing a Level Three Program

3.1 THE IF-THEN STATEMENT

```
400 IF (A + B) = C THEN 460
410 IF (A↑2 - B↑2)<> C THEN 490
420 IF A < = B THEN 500
```

The IF-THEN statement may be considered as a conditional GO TO statement. This statement causes the system to skip the normal sequence of program lines and go to the line number following the word THEN, providing certain conditions are met.

The IF-THEN statement consists of the word IF, followed by any BASIC expression, followed by a relational operator, followed by any BASIC expression, followed by the word THEN, followed by a line number.

The relational operators that may be used and their meaning are as follows:

<	is less then	<=	is less than or equal to
>	is greater than	>=	is greater than or equal to
=	is equal to	<>	is not equal to

If the specified condition is met, the system goes to the line number designated by THEN. If the specified condition is not met, the program will continue on to the next line in the program. In the following program?

```
440 IF D<0 THEN 500
450 IF D>0 THEN 530
470 R1 = -B/(2*A)
```

if the value of D is negative at the time line 440 is executed, the system will transfer to line number 500. If this condition is not met the system will simply continue on to the next line in the program, which is line 450.

If D is positive at the time line 450 is executed, the system will transfer to line 530. If this condition is not met, the system will continue on to the next line in the program which is line 470. This will cause the value for R1 to be calculated.

The following program shows how these statements could be used in a program for calculating the real roots of a quadratic equation.

```
400 REM     A, B, AND C ARE THE COEFFICIENTS; D THE DISCRIMINANT
420 READ A, B, C
430 LET D = B↑2 - 4*A*C
440 IF D<0 THEN 500
450 IF D>0 THEN 530
460 REM  CALCULATING ROOTS WHEN DISCRIMINANT IS ZERO
470 LET R1 = -B/(2*A)
480 LET R2 = R1
490 GO TO 560
500 REM  MESSAGE TO BE PRINTED IF DISCRIMINANT IS NEGATIVE
510 PRINT  "ROOTS ARE NOT REAL"
520 GO TO 420
530 REM  CALCULATING ROOTS WHEN DISCRIMINANT IS POSITIVE
540 R1 = (-B + SQR(D))/(2*A)
550 R2 = (-B - SQR(D))/(2*A)
560 PRINT A; B; C; R1; R2
570 GO TO 420
580 DATA 3, 8,5,   1.8, 5.7, 9.9,   1,2,1
590 END
```

When the value of the discriminant is negative at the time line 440 is executed, the system transfers to line 500 which causes the message "ROOTS ARE NOT REAL" to be printed out. Control is then transferred to the READ statement on line 420.

If the value of the discriminant is not negative at the time line 440 is executed, the system will continue on to the next line.

When the value of the discriminant is positive at the time line 450 is executed, the system will transfer to line 530. The value of the two roots are calculated. The values of the coefficients and of the two roots are printed out, and control is transferred to the READ statement in line 420.

When the value of the discriminant is zero at the time line 450 is executed, the system continues on to the next statement. The values of the two roots are calculated, and control is transferred to line 560. This causes the values of the coefficients and the value of the roots to be printed out. Control is then transferred back to the READ statement in line 420.

The loops in the program cause control to be always transferred back to the READ statement in line 420. When the system executes this statement and there is no more data, execution of the program is terminated.

It is possible to have control transferred to some statement in the program *after* the last data item has been read and used. One way of doing this is to place a *dummy* data item as the last item in a data statement. For example,

```
600 S = 0
610 READ K
615 IF K = 99999 THEN 640
620 S = S + K↑2
630 GO TO 610
640 PRINT " SUM OF SQUARES OF VALES OF K IS ", S
650 DATA 9.9,  12.567,  89.86,  123.45,  44.44, 1.96, 99999
660 END
```

In this program, the value of S is initialized at zero. As each value of K is read, its value is squared and this is then added to S. Control is then transferred to the READ statement in line 610 which causes the next value of K to be read, squared, and added to S. The last data item, 99999, is a *dummy* item. When it is read, the IF-THEN statement in line 615 causes control to be transferred to line 640 which causes the message SUM OF VALUES OF K IS and the value of S to be printed.

When the dummy item is read, control is transferred immediately to line 640 *before* the data item is used in any calculations. The data item is simply serving as a *flag* in the program to enable to have control transferred after the last data item has been read but before execution of the program is terminated.

This method can also be used to transfer control to the END statement after all the data has been read so that the program will terminate without an OUT OF DATA message being printed out.

The two expressions in the IF-THEN statement may be constants, variables, or combinations of constants, variables, and operations.

The line number following the word THEN may be any line number that exists in the program. However, it makes no sense if the line number following the word THEN is that of the *next* line in the program. For example, in the sequnce of statements

```
700 IF K > 0 THEN 710
710 W = 4*K + K↑2
```

the IF-THEN statement in line 700 serves no useful purpose. Control will be transferred to 710 regardless of what value K is. If K *is* greater than zero, control will be transferred to 710 by the IF-THEN statement. If K is *not* greater than zero, control will be transferred to 710 because it is next in the sequence of the program.

Exercises 3.1

1.a)What would be the printout of the following program:

```
 10 READ S
 20 IF S = 99999 THEN 999
 30 IF S ≤ 10000 THEN 90
 40 IF S ≤ 15000 THEN 70
 50 LET R = .05
 60 GO TO 100
 70 LET R = .045
 80 GO TO 100
 90 LET R = .04
100 LET C = R * S
110 PRINT USING 120, S, C
120 % TOTAL SALES  $#####.##   COMMISSION  $###.##
130 GO TO 10
140 DATA 11000,  16400,  8800, 15000
999 END
```

b) Rewrite the above program so that each salesman's number and his total sales are read in and each salesman's number, sales and commission are printed out, using the following data:

Number	Sales
1427	$10,090
1425	$11,789
1420	$15,555
1417	$ 8,960
1416	$10,000

2. What would be the printout of the following program?

~~XREAD~~

```
  5 PRINT "A", "B", "C"
 10 READ A, B
 11 IF A = 99999 THEN 999
100 IF (A - B)≤0 THEN 140
110 IF (A - B)≥0 THEN 160
120 LET C = 2 * A + 3 * B
130 GO TO 210
140 LET C = A + 3 * B
150 GO TO 210
160 LET C = 0
210 PRINT A, B, C
220 GO TO 10
900 DATA 13, 10, 25, 4, 16, 16, 99999, 99999
999 END
```

3. Write a program for calculating the value of Z, given the value of X and of Y, if

Z = X + Y when Y is greater than X

Z = X - Y when Y is less than X

Z = 3.85 when X and Y are equal.

X	Y
36.765	19.88
34.234	45.99
23.456	23.456

4. Test for divisbility.

To determine whether the number X is divisible by the number Y, subtract the quotient X/Y from the greatest integer not greater than X/Y. If the result of the subtraction INT(X/Y) - (X/Y) is zero, the quotient must be a whole number, and X must therefore be divisible by Y.

If the difference is negative, the quotient must have digits in its decimal part, and therefore X is not divisible by Y.

Write and execute a program to determine which of the following numbers is divisible by 13.

371,293 4,826,809 62,748,517 5,826,809
36,936,931

5. Quadratic formula.

Using the formula given in exercise #9 on page 77, calculate the roots of the following equations. If the discriminant is negative, do not attempt to calculate the roots, but have the message ROOTS ARE NOT REAL printed out.

$$8.91x^2 - 11.49x + 4.51 = 0$$

$$6.77x^2 + 14.92x + 24.09 = 0$$

$$81.23x^2 - 27.61x + 14.80 = 0$$

$$25.48x^2 - 14.77x + 27.27 = 0$$

$$7.76x^2 - 88.66x - 70.70 = 0$$

3.2 INCREMENTING A VARIABLE IN A LOOP

The value of a variable may be incremented by a fixed amount each time a loop in a program is executed. For example:

```
800 LET S = 0
810 LET X = 924
820 LET S = S + X
830 LET X = X + 1
840 IF X = 1137 THEN 860
850 GO TO 820
860 PRINT S
870 END
```

The purpose of this program is to calculate the sum of all whole numbers from 924 through 1136 and to print that sum.

Lines 820, 830, 840, and 850 constitute a loop. Each time the loop is executed, the value of X is incremented by 1 and this new number added to the existing sum. When the value of X becomes 1137, then exit is made from the loop to statement 860, which causes the sum to be printed.

If the variable has an initial value of zero and is incremented by one each time the loop is executed, the variable is called a counter. In effect, it counts the number of times the loop is executed. This is very useful in calculating averages. For example,

```
900 LET C = 0
910 LET S = 0
920 READ X
930 IF X = 999999 THEN 980
935 REM  S IS THE SUM OF ALL X
940 LET S = S + X
950 C = C + 1
960 GO TO 920
965  REM  A IS THE AVERAGE OF ALL X
970 A = S/C
980 PRINT A
985 DATA  154358,  897420,  263547,  203948,  658493
986 DATA  884499, 9287654,  092342,  996633, 203948
990 END
```

In this program the variable C is used as a counter. In effect, it is a count of the number of times the loop was executed. Since this equal to the number of data items read, it can be used in calculating the average of the numbers.

When a variable is incremented by a fixed amount each time a loop is executed, the increment may be any number. It can be a positive number, negative number, whole number or decimal. An increment of zero would, of course, serve no useful purpose.

The purpose of the program below is to print out the value of the sine, cosine, and tangent of every angle from 0° to 180° inclusive that is divisible by 3. The variable A is incremented by 3 each time the loop is executed. The program itself generates its own data.

```
900 LET A = 0
910 LET R = A/57.2958
920 PRINT A, SIN(R), COS(R), TAN(R)
930 A = A + 3
940 IF A > 180 THEN 970
960 GO TO 910
970 END
```

This is not the only way a variable may be incremented each time a loop is executed. Another, more efficient method is introduced in the next chapter.

Exercises 3.2

1. What would be the printout of each of the following programs?

a)

```
10 PRINT "VALUE OF X", "X SQUARED", "X CUBED"
15 LET X = 1
20 PRINT X, X↑2, X↑3
30 LET X = X + 1: IF X < 7 THEN 20
999 END
```

b)

```
3  LET N = 0: LET S = 0
10 READ X: IF X = 99999 THEN 40
20 LET S = S + X: LET N = N + 1: GO TO 10
40 LET A = S/N
50 PRINT A
60 DATA 20, 30, 40, 80, 160, 99999
70 END
```

2. For each of the following, write a program for printing the table described:

a) A table of all temperatures in both Farenheit and Centrigrade readings from -40°F to 212°F in increments of one degree if $C = \frac{5}{9}(F - 32)$ where C is the reading in degrees Centrigrade and F is the reading in degrees Farenheit.

b) A table of values of the sine, cosine, tangent, cosecant, secant, and cotangent of all angles from 0° through 360° in increments of five degrees.

c) A table of the square root, cube root, and fourth root of all numbers from 0.1 to 0.5 in increments of 0.01.

d) A table of sales tax on all sales from one cent to fifty dollars rounded to the nearest cent if the sales tax is 4%.

e) A table of all values of Y when X increases from 1.78 to 1.89 in increments of 0.01, given that $Y = 13.31x^3 - 1.49x^2 + 7.83x - 11.21$

f) a table of all values of $\sin^2 x - \cos^2 x$ as x increases from 0 to 6.30 radians in increments of 0.0175 radians.

3.3 THE STOP STATEMENT

The STOP statement will cause the system to terminate execution. There may be several STOP statements in a program. When the system encounters a STOP statement, it types

```
3300 BASIC READY
:
```

and returns control to the user. It serves the same purpose as a GO TO 9999 where 9999 is the line number for the END statement.

Program execution will be terminated when the system encounters a STOP statement, an END statement, an error, or when the ESC key is pressed.

3.4 THE LOGICAL FUNCTIONS BOOL, AND, OR

```
100 LET X = BOOL (A = 7)
120 LET Y = BOOL (B↑2 + C↑2 <D)
130 LET W = AND (A, B, C)
140 LET V = AND (A + B)
150 LET K = OR(A, B, C)
```

The value of the BOOL function will be 1 if the relation within the parentheses is true and zero if it is not true.

If the value of A in line 100 above is 1, then the value of 1 will be assigned to the variable X. If the value of A is not 7, the number zero will be assigned to X. The computer is handling the truth value of true as the number 1, and the truth value of false as zero.

The value of the AND function will be 1 if each of the expressions within the parentheses is non-zero. Otherwise, the value of the AND function will be zero.

The value of the OR function will be 1 if any expression within the parentheses has a non-zero value. If each expression within the parentheses has a value of zero, the value of the OR function will be zero.

With both the AND function and the OR function, the system is handling the truth value of the expression within the parentheses so that zero has a truth value of false and non-zero numbers have a truth value of true.

3.5 STRING VARIABLES

Variables may represent things other than numbers. Variables may represent any BASIC alphanumeric characters including digits, letters, and BASIC special characters. A set of alphanumeric characters enclosed within double quotation marks is called a literal string. You are already familiar with literal strings used in PRINT statements.

A variable used to represent a literal string is called a string variable. A string variable name consists of any letter followed by $. Examples of string variable names are: A$ N$ K$ Z$.

A literal string may be any length that can be expressed on one program line. However, a string variable can represent a literal string of only up to 18 characters.

String variables permit the user to input, process, output, and print literal strings such as names, addresses, part numbers, and report titles.

The following program is intended for reading part numbers, the number of each part on hand, calculating the total number of part WBC4 on hand, and printing out this total.

```
10 LET T = 0
30 READ B$, X
40 IF B$ = "ZZZZZ" THEN 80
50 IF B$ = <> "WBC4" THEN ~~60~~ 30
~~55 GO TO 30~~
60 T = T + X
70 GO TO 30
80 PRINT "PART NUMBER", B$, "TOTAL ON HAND", T
90 DATA "WBC4", 2352, "WBC4", 199, "WKL9", 1887
91 DATA "WNC4", 8766, "ABC7", 345, "WNC4", 456
92 DATA "WBC4", 1496, "WKL9", 187, "WBC4", 1456
93 DATA "ZZZZZ", 0
9999 END
```

The output from this program would be:

```
PART NUMBER        WBC4                   TOTAL ON HAND       5504
```

Arithmetic operations cannot be performed on on string variables. However, each alphanumeric character is actually handled in the computer as a numeric code. Therefore, alphanumeric characters may be compared for equality, and for greater than and less than relationships.

Below is an assortment of statements containing literal strings and string variables which show the various ways they may be used in BASIC statements. Line numbers have been omitted.

```
LET N$ = "HJKLT"
IF B < N$ THEN 90
IF F >= "G5H6" THEN 9999
PRINT "NAME IS", A$
LET A$ = M$
IF "ABCD" < J$ THEN 60
INPUT A$, B$, V
READ Y$, U$, K
LET A$ = "CHARLES HARRISON"
LETB$ ="1685 NOTT TERRACE"
LET C$ = "SCHENECTADY, N.Y."
PRINT A$, B$, C$
```

The PRINT USING and image statements are used for alphanumeric string variables and literal strings. Each character in the image statement format is replaced by a character in the literal string.

```
10 LET A$ = "HENRY JEFFERSON"
20 PRINT USING 30, A$
30 % SALESMAN: ###############
```

The text is printed out left-justified, with blanks being inserted on the right. If the text string is longer than the format specification, the text string is truncated on the right. The printout of the above program would be:

```
SALESMAN:  HENRY JEFFERSON
```

Exercises 3.5

1. What would be the printout of the following program?

```
10 READ A$, B$, C$, D$
20 PRINT C$, B$, A$, D$
30 DATA "ST ARRANGE IT CORR", "OD SENSE IF YOU JU"
31 DATA "THIS MAKES VERY GO", "ECTLY"
32 END
```

2. What would be the printout of the following program?

```
10 PRINT "EMPLOYEE", "GROSS PAY"
20 PRINT
30 PRINT
31 REM  H IS THE NUMBER OF REGULAR HOURS WORKED
32 REM  R IS THE HOURLY PAY RATE
33 REM V IS THE NUMBER OF OVERTIME HOURS WORKED
40 READ N$, R, H, V
50 LET P = R*(H + 1.5*V)
60 PRINT N$, P
70 GO TO 40
80 DATA "GEORGE MARS", 4.40,  40,  4
81 DATA "HENRY JONES", 6.00,  40,  8
82 DATA "HAROLD MILLER",  7.10,  38,  0
83 DATA "ERIC LAROCK", 6.60,  40,  2
84 DATA "PETER MASON", 5.20,  32,  0
90 END
```

3. Describe the printout of the following program.

```
10 READ A$, B$, C$, D$, E$, F$, G$
20 PRINT USING 300, A$
30 PRINT USING 310, B$, C$, D$
40 LET X = 0
50 LET R = X/57.2958
60 PRINT USING 320, SIN(R), COS(R), TAN(R)
70 LET X = X + 1
80 IF X>360 THEN 100
90 GO TO 50
100 PRINT USING 310, E$, F$, G$
200 LET X = 0
210 R = X/57.2958
220 PRINT USING 320, 1/SIN(R), 1/COS(R), 1/TAN(R)
230 LET X = X + 1
240 IF X>360 THEN 350
250 GO TO  210
260 DATA "TRIG RATIO TABLES", "SINE", "COSINE", "TANGENT"
270 DATA "COSECANT", "SECANT", "COTANGENT"
300 %              ################
310 %#########     ########      ########
320% ####.####     ####.####     ####.####
350 END
```

3.6 THE STRING FUNCTION

The STR function permits the user to extract, examine, compare, or r eplace a specified portion of an alphanumeric string.

STR(D$, 5, 7) means to take the seven characters beginning with the fifth of the string D$.

STR(E$,9) means to take the remainder of the string E$, beginning with the ninth character.
The STR function may be used on either side of an equal sign or relation. It can be used in any BASIC statement where alphanumeric variables are permissible.

Below are some examples of some statements containing the STR functions.

```
10 LET B$ = "ABCDEFGHIJKLMN"

20 LET C$ = STR(B$,6,6)            C$ is set to the six
                                   characters in B$ beginning
                                   with the sixth, "FGHIJK"'

30 LET D$ = STR(B$,8)              D$ is set to the remaindor
                                   of B$ starting with the
                                   eighth character, "HIJKLMN"

40 STR(E$,4,6) = C$                characters 4 through 9 in
                                   E$ are set to C$, "FGHIJK"

50 STR(E$,1,3) = STR(D$,5,3)       characters 1 through 3 in
                                   E$ are set to 'LMN' which
                                   are the three characters
                                   in D$ starting with the fifth.

60 IF STR(E$,3,2) = 'NF' THEN 90   If characters 3 and 4 in
                                   in E$ are 'NF', then control
                                   is transferred to line 90.
```

This function is used a great deal in updating master files, including changing addresses, and correcting errors such as errors in a file of report card grades.

Exercises 3.6

1. What would be the output of each of the following programs?

a)

```
10 READ A$, B$, C$
20 IF A$ = "END" THEN 120
30 IF A$ = "C. P. JOHNSON" THEN 60
40 PRINT A$, B$, C$
50 GO TO 10
60 LET B$ = "1717 DUDLEY AVE"
70 PRINT A$, B$, C$
80 GO TO 10
90 DATA "A. W. ANDERSON","1836 NOTT TERRACE","SCHENECTADY"
100 DATA "C. P. JOHNSON","1721 FOREST ROAD", "SCHENECTADY"
110 DATA "W. W. STOWE", "1947 ALBANY ST","SCHENECTADY"
115 DATA "END", "END", "END"
120 END
```

b)

```
10 READ N$, M$
20 IF N$ = "HENRY MARTIN" THEN 50
30 PRINT N$, M$
40 GO TO 10
50 LET M$ = "F  D  D  C"
60 PRINT N$, M$
70 GO TO 10
80 DATA "GEORGE HAMILTON", "A  B  A  C"
90 DATA "HENRY MARTIN", "B  B  C  B"
100 DATA "CHRISTOPHER GEORGE", "F  F  D  C"
```

e)

```
10 READ A$
20 IF A$ = "END" THEN 180
30 LET W$ = " "
40 LET X$ = "."
50 LET STR(R$1,1) = STR(A$,1,1)
60 LET STR(R$,2,1) = X$
70 LET STR(R$,3,1) = W$
80 LET STR(R$,4,1) = STR(A$,2,1)
90 STR(R$,5,1) = X$
100 LET STR(R$,6,1) = W$
110 LET STR(R$,7) = STR(A$,3)
120 PRINT R$
130 GO TO 10
140 DATA "PWMCNALLY"
150 DATA "LAARMSTRONG"
160 DATA "GBCHARBONNEAU"
170 DATA "END"
180 END
```

d)

```
10 READ A$, B$,
20 LET C$ = "IS937ISF WNYUN"
30 LET M$ = STR(A$,1,4)
40 LET STR(N$,1,2) = STR(A$,9)
50 LET STR(N$,3,1) = STR(B$,1,1)
60 LET P¢ = STR(B$,3,5)
70 LET STR(Q$,1,2) = STR(B$,8)
80 LET STR(Q$,3,2) = STR(C$, 1,2)
90 LET R$ = STR(C$,6,2)
100 LET STR(T$,1,1) = STR(C$,8,1)
110 LET STR(T$, 2,2) = STR(C$,13,2)
120 PRINT USING 130, M$, N$, P$, Q$, R$, T$
130 % #### ### ##### #### ###
140 DATA "DONTKOOKYO","UVTHINKTH"
150 END
```

2. Write a program for reading in each of the code words below and changing the middle three letters of the last word to WKV and changing the middle three letters of each of the other words to the last three letters of the word thats follows it.

WABVPQRTV

AQIHOPBNF

CJDRGKLME

UYYGWWZXV

3.7 THE RESTORE STATEMENT

```
100 RESTORE
120 RESTORE 6
```

The RESTORE statement allows repetitive use of data in DATA statements.

When RESTORE is encountered, the system returns to the first data number. This means a set of data may be used several times in a program.

The statement in line 100 above will cause the system to return to the first data value.

The statement in line 120 above will cause the system to return to the sixth data value.

The program on the next page will cause the system to print out the values among those in the given data whose square is less than 197,000 and also to print out the values among those in the given data whose cube is less than 1,011,789.

```
10 PRINT USING 200
20 PRINT USING 210
30 READ X
40 IF X = 99999 THEN 140
50 LET S = X↑2
60 IF S>=197000 THEN 30
70 PRINT USING 220, X, S
80 GO TO 30
140 RESTORE
145 PRINT USING 230
150 PRINT USING 240
155 READ Y
160 IF Y = 99999 THEN 300
165 LET C = Y↑3
170 IF C>=1011789 THEN 155
175 PRINT USING 250, Y, S
180 GO TO 155
190 DATA 92, 98, 165, 170, 432, 450, 467, 499, 99999
200 % DATA ITEMS WHOSE SQUARE IS LESS THAN 197,000
210 % DATA ITEM                    DATA ITEM SQUARED
220 % #########                    #################
230 % DATA ITEMS WHOSE CUBE IS LESS THAN 1,011,789
240 % DATA ITEM                      DATA ITEM CUBED
250 % #########                      ###############
300 END
```

Exercises 3.7

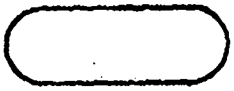
1. What would be the output of the following program?

```
10 PRINT "X", "X SQUARED"

20 READ X

30 IF X = 99999 THEN 70

40 A = X↑2

50  PRINT X, A

60 GO TO 20

70 RESTORE 2

80 PRINT "X", "SUM OF X AND X SQUARED"

90 READ X

100 IF X = 99999 THEN 120
105 LET B = X + X↑ 2
110 PRINT X, B

115 GO TO 90

120 RESTORE 5

130 PRINT "X", "X CUBED"

135 READ X

140 IF X = 99999 THEN 170

145 LET C = X↑3

150 PRINT X, C

160 DATA 1, 2, 5, 6, 7, 9, 10, 99999

170 END
```

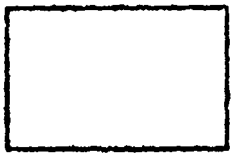
3.8 FLOWCHARTING

A general plan of a program should be formulated before the program is actually written out in detail. One type of such a plan is a flowchart, in which figures such as circles, rectangles, and diamonds are used to represent the logic of the solution to the problem.

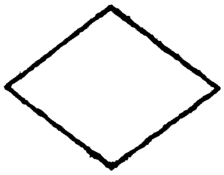
A few of the symbols commonly used are:



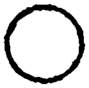
The terminal symbol, used to represent the beginning or the end of a program.



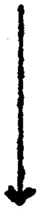
The operation symbol used to represent calculations. It is also used to represent any process not described by any other more specific symbol.



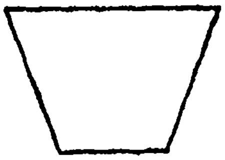
The decision symbol, used to indicate where in the program a decision has to be made, quantities compared, or a question answered before the proper path through the program can be determined.



The connector symbol used when a flowchart must be continued onto another page or when there is a break in the path of the line of flow.



The flow symbol indicates the direction of flow through the program. The direction of flow is usually from top to bottom and left to right whenever possible. Flow lines do not cross.



The input-output symbol represents the place in the program where data is read into the system or where the system produces printed output.

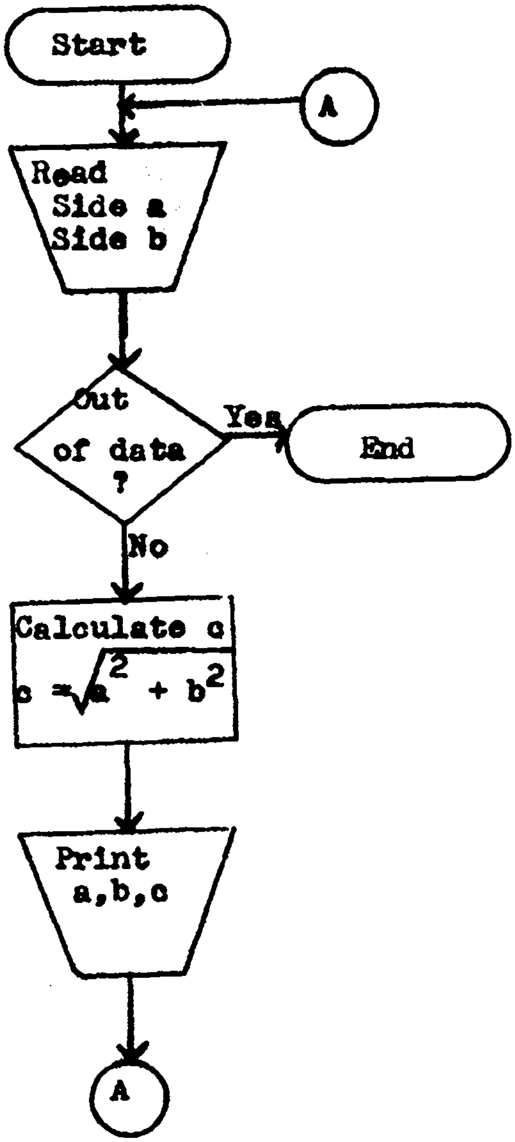
There are several other commonly used symbols, some of which are introduced in later chapters.

Flowcharting of some type is usually essential in planning complicated programs. However, the best way to learn flowcharting techniques is to begin by flowcharting even the simplest programs.

Flowcharting is an aid in planning your program, in debugging your program to find programming errors, and in attempting to explain your program to others.

A flowchart may be prepared by sketching the symbols on paper, or by use of a template which is a clear plastic plate containing cutouts of the flowcharting symbols.

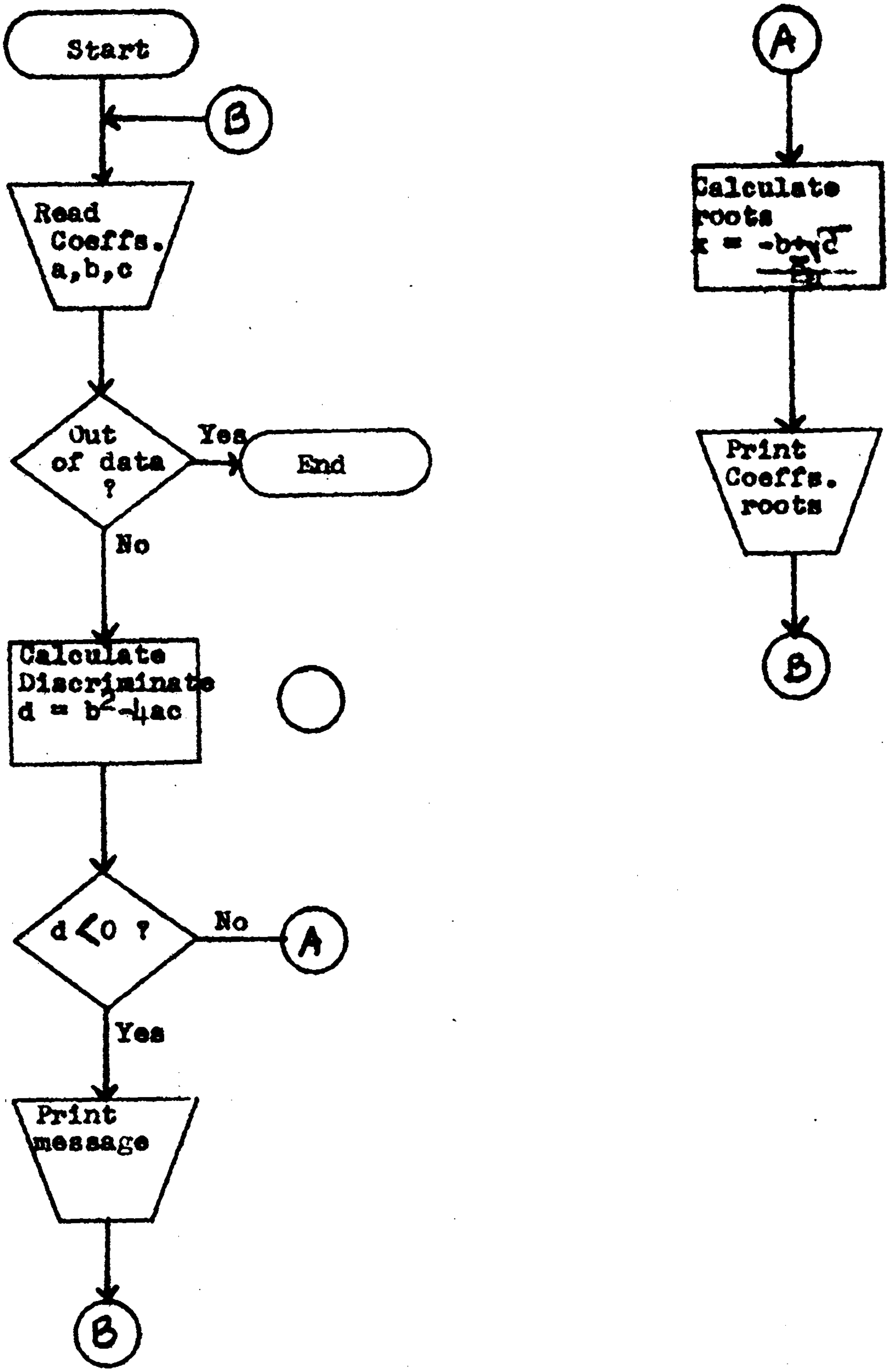
A flowchart for planning a program to calculate the length of the hypotenuse of a right triangle when given the lengths of each of the two legs, using the Pythagorean relationship $c = \sqrt{a^2 + b^2}$, might be written as follows:



The program might then be written as:

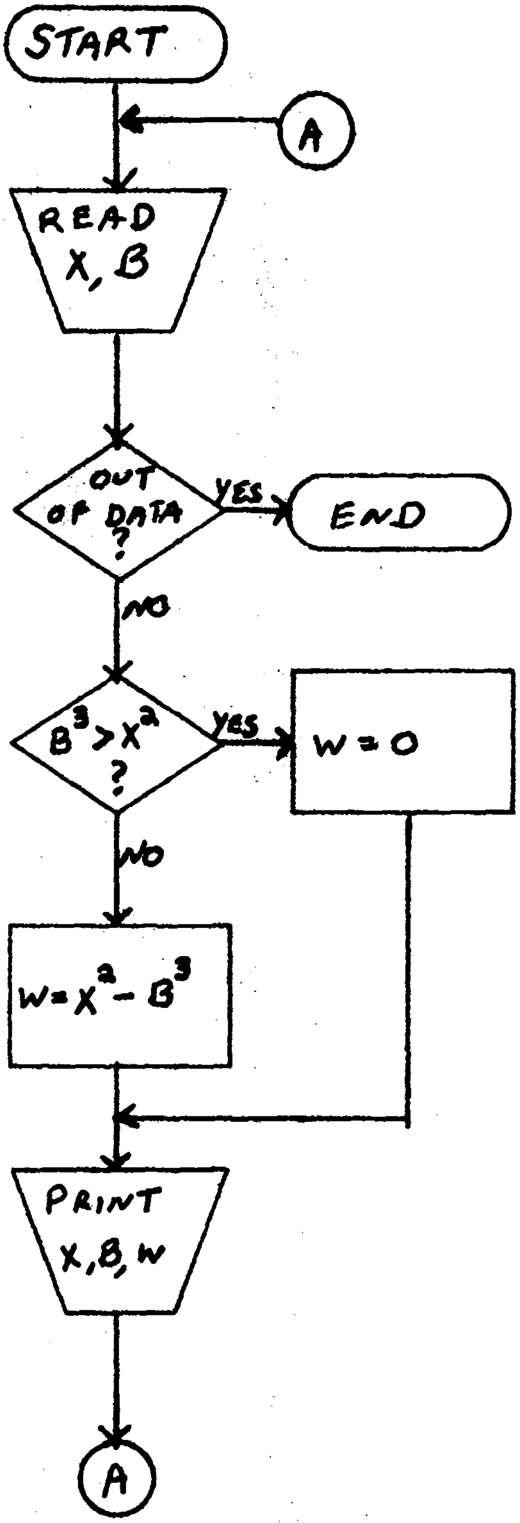
```
10 READ A, B
20 IF A = 99999 THEN 999
30 LET C = SQR(A↑2 + B↑2)
40 PRINT A, B, C
50 GO TO 10
999 END
```

The following is a flowchart for planning a program to calculate the real roots of quadratic equations.

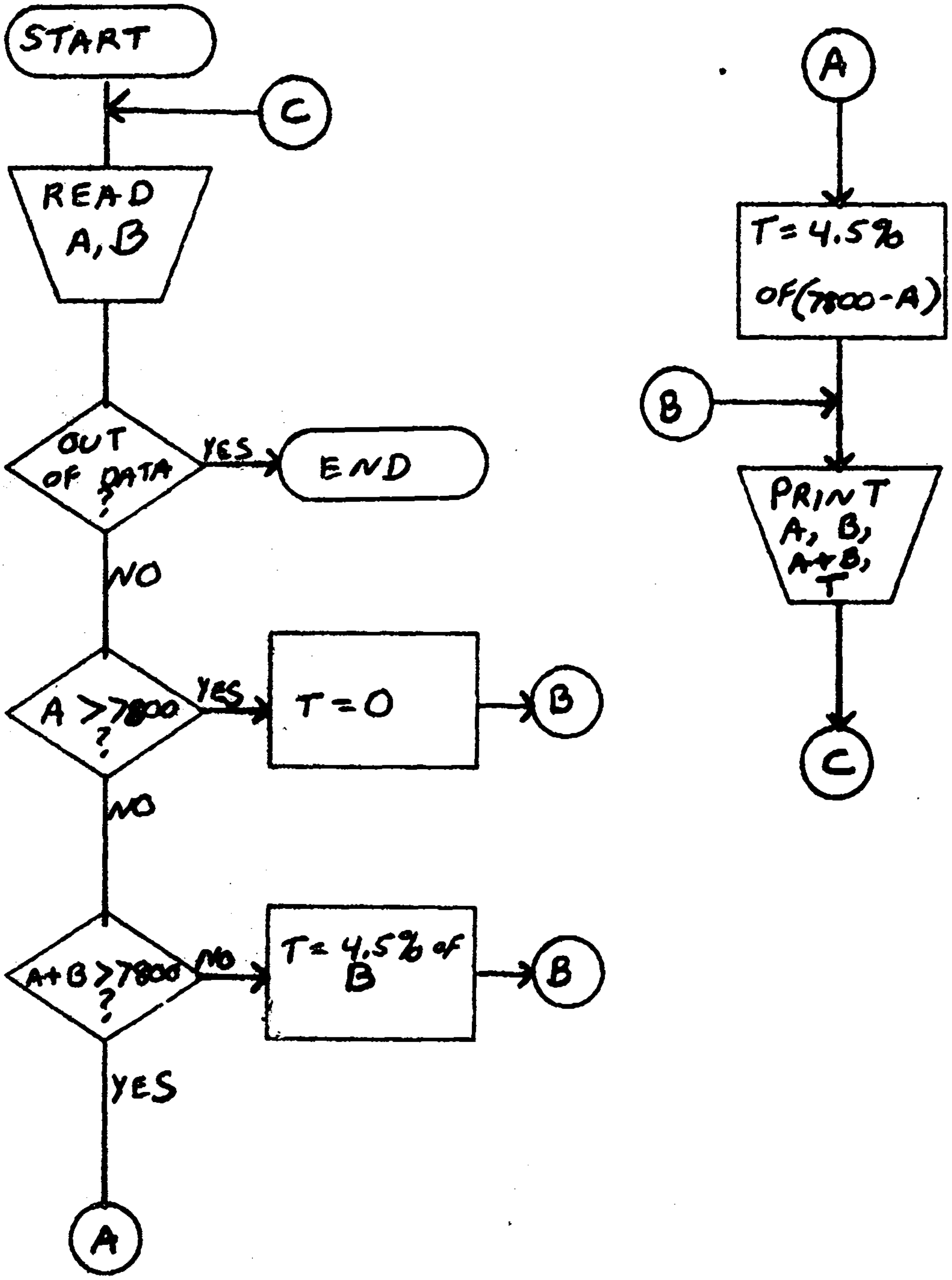


Exercises 3.8

1. Write the program that has been outlined in the following flowchart.



2. Write the program that has been outlined in the following flowchart. Data may be omitted.



3. **Flowchart each of the following programs.**

a)

```
5 LET S1 = 0: LET S2 = 0: LET N = 0
10 READ X
20 IF X = 99999 THEN 70
30 LET S1 = S1 + X
40 LET S2 = S2 + X↑2
50 LET N = N + 1
60 GO TO 10
70 LET A1 = S1/N
80 LET A2 = S2/N
90 PRINT A1, A2
100 END
```

B)

```
50 LET C = -50
60 LET F = 9/5*C + 32
70 LET C = C + 1
80 IF C>100 THEN 110
90 PRINT C, F
100 GO TO 60
110 END
```

3.9 FINDING MAXIMUM AND MINIMUM VALUES

```
10 LET M = -1.0E+60          5 READ X
                             1ϕ IF X = 999999 THEN 70
20 READ X                    15 M=X

25 IF X = 999999 THEN 70

30 IF X>M THEN 50

40 GO TO 20

50 LET M = X

60 GO TO 20

70 PRINT "HIGHEST SCORE IS"; M

80 END
```

The program above will determine which number in a given set of data is the greatest number.

When the first data item is read, it is compared with M which has arbitrarily set at -1.0E+60, a very small number. Since X is greater than M, the program will branch to line 50 which will cause the value of X to be stored in location M.

The next data value will then be read. If it is greater than M, the program will branch to line 50 and this value of X will be stored in location M. If X is npt greater than M, nothing will happen and the program will branch back to line 20 to read the next data value.

The value of M will always be the greatest value of X that has been read. The program will therefore determine the greatest number in any given set of numbers.

The following program will determine the least number in any given set of numbers.

```
10 LET L = 1.0E+60
20 READ X
30 IF X = 99999 THEN 80
40 IF X<L THEN 60
50 GO TO 20
60 LET L = X
70 GO TO 20
80 PRINT "LEAST SCORE IS", L
90 END
```

Exercises 3.9

1. Write a program that will determine the greatest and also the least score in any given set of scores. Omit the data.

2. Write a program for calculating the maximum value of C given sets of values of A and B if $C = -A^3 + 13.6B - 10$. Omit the data.

3. Write a program for calculating the maximum value of K if $K = \sin X + \cos X - \tan X$, as X varies from -1.5708 to +1.5708 radians in increments of .075 radians.

4. Write a program for calculating the minimum value of Y if $Y = .313x^3 - 2.12x^2 - 4.41x + 16.11$ as x varies from -1 to +1 in increments of .1 .

3.10 FACTORIAL N

Factorial n, denoted by n!, is defined as $n! = n(n - 1)(n - 2)(n - 3) \ldots (1)$. That is, $5! = 5\cdot4\cdot3\cdot2\cdot1$ or 120. 0! is defined as being equal to 1.

The following program may be used to calculate n! when n is 2 or greater.

```
10 READ N
20 LET Y = N
30 LET A = N - 1
40 LET Y = Y * A
50 LET A = A - 1
60 IF A = 0 THEN 80
70 GO TO 40
80 PRINT N; "FACTORIAL IS EQUAL TO"; Y
90 END
```

The largest factorial that tho computer can handle is about 50!

Exercises 3.10

1. Modify the above program so that when N = 0 the message ZERO FACTORIAL IS EQUAL TO 1, and when N is negative, the message NEGATIVE FACTORIAL IS UNDEFINED is printed out.
2. Write a program to determine the largest factorial that can be handled by your computer.

3.11 WRITING A LEVEL THREE PROGRAM

A typical level three program includes a flowchart and the following specificarions:

1. A REM statement giving the programmer's name, date and level of the program.
2. A REM statement giving the title of the program.
3. One or more loops using an IF-THEN statement to exit from the loop.

Optional features for the program include:

a) dummy data item;

b) incrementing a variable in a loop;

c) the STOP statement;

d) the logical functions BOOL, AND, OR;

e) string variables;

f) the STR function;

g) the RESTORE statement;

h) PRINT or PRINT USING statements for printing appropriate column headings;

i) PRINT or PRINT USING statements for printing values in neat columns.

Exercises 3.11

Prepare a flowchart and write a level three program for one or more of the following problems.

Algebra Problems

1. Divisibility. Using the techniques described on page 92c, write a program for testing each of the following numbers for divisibility by 7, 13, 23, and 31.

   373,497 5,837,903 63,549,717

   5,927,913 36,936,931

2. PRIME numbers. A prime number is any integer greater than one whose only factors are itself and one. Two is prime. No even number greater than two is prime because it has two as a factor.

   To determine whether N is a prime number, test it for divisibility by all odd numbers greater than one and less than the square root of N.

   a) Write a program to test whether 51,051 is prime.

   b) Write a program to test whether any given set of numbers contains one or more primes. Omit data from your program.

   c) Write a program to determine which integers less than 500 are primes.

3. Fibonacci numbers. The first eleven Fibinacci numbers are 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, and 89. Each number os a Fibonacci series is the sum of the two numbers immediately preceeding it.

   a) Write a program to calculate and print the first 100 Fibonacci numbers.

   b) Write a program to deminstrate that the square of any number in the Fibinacci series and the product of the number just before and just after it in the series always differ by 1.

4. The value of sin x. If x is expressed in radians, then

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

   a) Write a program for calculating sin 37° correct to four decimal places.

   b) Write a program for calculating sin x as x increases from 0 to π radians in increments of .1 radians.

5. **Sets of linear equations in two variables.**

If $Ax + By = C$ and $Dx + Ey = F$, then

$$x = \frac{CE - BF}{AE - BD} \text{ and } y = \frac{AF - CD}{AE - BD} .$$

If $AE - DB = 0$, there is no unique solution to the system of equations.

Write a program for calculating the value of x and y in each of the following sets of linear equations. If there is no unique solution to the system of equations, have the message THE EQUATIONS ARE EQUIVALENT OR INCONSISTENT printed out.

a) $x + 2y = 5$
$2x - 5y = -8$

b) $2x - 3y = 16$
$3x + 2y = 11$

c) $5x - 3y = 21$
$x + 2y = -1$

d) $x + 2y = 3$
$2x + 4y = 6$

e) $1.83x - 4.57y = 153.4$
$5.67x + 9.36y = -139.2$

6. **Factorial n.** a) Using the routine described on page 122, write a program for calculating all values of Y, given the values of X below, if $Y = \frac{6X^2 + 7!}{5!}$

$X = 7, 11, 13, 16, 17, 41.39$

b) Write a program for calculating all values of W, given the values of X below, if $W = \frac{X! - 3!}{(X - 3)!}$

$X = 3, 8, 11, 16$

7. <u>Permutations.</u> A permutation is an arrangement of a group of things in a definite order. The three letters A, B, and C can be arranged in six different orders: ABC, ACB, BCA, BAC, CAB, and CBA. Each is a permutation.

The number of permutations of n things taken n at a time, written ${}_nP_n$, is $n!$ There are $3!$ or $3\cdot2\cdot1$ or 6 permutations of the three letters A, B, and C if taken three at a time.

The number of permutations of n things taken r at a time is ${}_nP_r = (n)(n-1)(n-2)(n-3)(n-4)\ldots(n-r+1)$. The number of permutations of 7 things taken 5 at a time is $7\cdot6\cdot5\cdot4\cdot3$ or 2520.

This may be stated in another way:

${}_nP_r$ = the product of the first r factors of $n!$.

Still another formula for stating the same thing is ${}_nP_r = \frac{n!}{(n-r)!}$. That is ${}_7P_5 = \frac{7!}{2!} = \frac{7\cdot6\cdot5\cdot4\cdot3\cdot2\cdot1}{2\cdot1}$. This is equivalent to the first five terms of $7!$. This last formula is often the easiest one to program.

a) Write a program for calculating the number of different orders in which X number of students can be arranged in a line when X takes on the values of 32, 39, 17, 14, and 23.

b) The number of permutations for n different things arranged in a circle is $(n - 1)!$. Write a program for calculating the number of different orders in which Y number of people can be arranged in a circle when Y takes on the values of 26, 23, 31, 37, and 44.

c) Write a program to calculate each of the following:

$_{25}P_7$ $_{25}P_{18}$ $_{13}P_9$ $_{13}P_4$ $_9P_6$ $_9P_3$

$_8P_7$ $_8P_8$ $_{11}P_{10}$ $_{11}P_{11}$ $_{21}P_{20}$ $_{21}P_{21}$

From the results of this program can you see any relationship between the values of $_nP_r$ and $_nP_{n-r}$? between $_nP_n$ and $_nP_{n-1}$?

8. Combinations. A combination is a definite group of elements without regard to their arrangement among themselves.

If the four letters A, B, C, and D are taken three at a time, four different combintaions can be formed: ABC, ABD, ACD, and BCD. In combinations, the order of the letters does not matter.

The number of combinations of n things taken r at a time is written $_nC_r$. It is equal to the number of permutations of n things taken r at a time, divided by r!.

$${}_{n}C_{r} = \frac{{}_{n}P_{r}}{r!} = \frac{n!}{(n-r)!r!}$$

$${}_{4}C_{3} = \frac{4!}{1!3!} = \frac{4\cdot3\cdot2\cdot1}{1\cdot3\cdot2\cdot1} = 4$$

Write a program to calculate each of the following:

${}_{13}C_{10}$ ${}_{13}C_{3}$ ${}_{14}C_{9}$ ${}_{14}C_{5}$

${}_{13}C_{13}$ ${}_{13}C_{12}$ ${}_{9}C_{9}$ ${}_{9}C_{8}$

From the results of this program do you see any relationship between ${}_{n}C_{r}$ and ${}_{n}C_{n-r}$? between ${}_{n}C_{n}$ and ${}_{n}C_{n-1}$?

9. <u>Binomial expansion.</u> In the rth term of $(a + b)^{n}$

the exponent of b is $(r - 1)$;

the exponent of a is $(n - r + 1)$;

the coefficient is ${}_{n}C_{(r-1)}$

The rth term = ${}_{n}C_{(r-1)}a^{n-r+1}\ b^{r-1}$

Write a program for calculating

the seventh term of $(x + 3)^{13}$

the eleventh term of $(x - 11.31)^{21}$

the fifth term of $(3.17x - 14.3)^{9}$

10. Random numbers. Write a program for generating 25 random numbers between 0 and 1, 30 random numbers between 10 and 100 and 5 random numbers between 100 and 1000.

11. Quadratic formula. a) Write a program for calculating the real roots of each of the following quadratic equations. If the roots are not real, have the message ROOTS NOT REAL printed out.

b) Modify your program so that each complex root will be handled in the computer as two real numbers F and G where $F = \frac{-b}{2a}$ and $G = \frac{\sqrt{|b^2 - 4ac|}}{2a}$
Have the two real numbers printed out as the complex number F + Gi. Do this for each complex root.

$$6.1x^2 - 13.6\,x + 25.7 = 0$$

$$1.1x^2 + 9.3x - 2.7 = 0$$

$$17x^2 + 8x + 5 = 0$$

$$1.4x^2 - 2x - 1 = 0$$

$$16x^2 + 8x + 1 = 0$$

12. Maximum and minimum values. Write a program for calculating all values of Y and for printing out only the maximum and minimum values of Y in each of the following equations as X varies from -60 to +60 in increments of .1.

a) $Y = 2x^3 - 6x^2 + 7$

$Y = 3x^3 - 15x^2 + 16x$

c) $Y = 3x^4 - 4x^3 - 12x^2 + 3$

d) $Y = x^3 - 27x$

e) $Y = x^5 - 5x + 2$

STATISTICS

13. The geometric mean. The geometric mean of a set of N numbers is the nth root of the product of the numbers.

$$G = \sqrt[N]{X_1 X_2 X_3 \ldots X_n}$$

Write a program for calculating the geometric mean of each of the following sets of numbers:

a) 2,8,7,9,10,23

b) 1,3,9,7,11,11

c) 1,3,10,20,30

14. The harmonic mean. The harmonic mean, H, of a set of N numbers is the reciprocal of the arithmetic mean of the reciprocals of the numbers.

Write a program for finding the harmonic mean of each of the following sets of numbers.

a) 3,5,7,8,9,10

b) 345,567,99

15. If a person travels a given distance at an average rate of A miles per hour, and returns at a rate of B miles per hour, the average speed for the entire trip is the harmonic mean of A and B. Write a program for finding the average speed for each of the following trips:

trip	av. speed going	av. speed returning
#1	55	65
#2	48	61
#3	350	275
#4	642	751
#5	511	711

16. The root mean square (r.m.s). The root mean square or quadratic mean of a set of numbers is the square root of the mean of the sum of squares of the number.

$$rms = \sqrt{\frac{s x^2}{n}}$$

Write a program for finding the root mean square of each of the following sets of data:

a) 87,67,65,67,89,88,76,93,,83,91

b) 3,5,7,6,4,8,2,7,8,6,5,4,8,7,6,5,9,9,8

c) 234, 567, 853, 472, 4987, 445, 666

17. <u>The mean deviation.</u> To find the mean deviation or average deviation of a set of N numbers,

find the mean of the numbers, $\bar{X}$;

subtract the mean from each number and sum the absolute values of the results, $\Sigma|X - \bar{X}|$;

divide by N, $\frac{\Sigma|X - \bar{X}|}{N}$

$$M.D. = \left|\overline{X - \bar{X}}\right|$$

Write a program for finding the mean deviation of the following set of data:

87 89 65 86 93 87 88 68 74 30

67 78 73 91 29 78 76 74 93 77

18. <u>Range of a set of scores.</u> The range of a set of scores is the difference between the greatest and the least value in the set of data.

Write a program for finding the range of a set of data. The data is to be supplied at the time of execut ion of the program. Therefore, have the program contain an input statement.

19. Standard deviation. The standard deviation of a set of scores is the root mean square of the deviations from the mean. It is also called the root mean square deviation. It is equal to the square root of the result of subtracting the square of the mean of the scores from the mean of the squares of the scores.

$$S.\ D. = \sqrt{\overline{X^2} - (\overline{X})^2}$$

Write a program for calculating the standard deviation of each of the following sets of scores.

Set #1	Set #2	Set #3
89	48.1	2018
88	55.6	2021
76	77.2	2023
91	81.3	2024
88	90.4	2027
85	93.2	2030
90	96.2	2030
89	97.9	2031
	98.3	2032
	98.8	2034
	99.9	2035
		2037
		2039
		2040
		2041
		2047

20. <u>Correlation coefficient.</u> A product-mement formula for the linear correlation coefficient is:

$$r = \frac{N \Sigma XY - (\Sigma X)(\Sigma Y)}{\sqrt{[N \Sigma X^2 - (\Sigma X)^2][N \Sigma Y^2 - (\Sigma Y)^2}}$$

Write a program to calculate the linear correlation coefficient for the values of X and Y given below.

X	Y
75	78
73	76
77	78
74	75
78	78
72	75
70	77
88	74
68	80
77	77
79	77
81	79

21. Frequency distribution. Write a program to calculate the number of measurements that fall into each of the following classes or categories:

60 - 62
63 - 65
66 - 68
69 - 71
72 - 74

Use the following data.

60.2	66.0	68.3
67.0	64.1	63.2
63.1	61.3	66.4
66.3	63.0	68.0
64.0	66.1	61.9
67.1	67.3	68.0
64.0	66.1	61.9
63.4	64.8	64.9
67.5	66.7	67/2
66.2	67.9	68.4
67.8	68.3	73.6
69.0	66.8	68.6
71.0	72.3	74.0

22. Standard scores. If deviations from the mean are expressed in terms of the standard deviation, they called standard units or standard scores. They are also called z-scores. A z-score is obtained by dividing the deviation from the mean by the standard deviation.

$$z = \frac{X - \bar{X}}{s}$$

Write a program for calculating the standard deviation of each of the following sets of scores and for converting a score of 112 on the Chemistry test and a score of 189 on the Physics test to z-scores. Which of these two scores is relatively better?

Chemistry	Physics
99	200
100	190
115	250
114	186
88	175
76	150
120	202
55	190
125	187
108	189
109	140
112	165
116	175
101	190
100	200

ALGEBRA PROBLEMS

23. **Approximating real roots of higher degree equations.**

Write a program using the SGN( ) function that will determine whether the value of f(x) is positive, negative or zero for each value of x as x increases from -20 to +20 in increments of .01. Whenever there is a change in sign of f(x) for two consecutive values of x, one root of the equation lies between these two values of x. Have the program print out these values of x. From these results, determine the real roots of the equation to the nearest tenth.

Modify your program to find the real roots of the equations to the nearest hundredth.

$f(x) = x^3 - 3x^2 + 4x - 5$ $\qquad x^3 - 3x^2 + 4x - 5 = 0$

$f(x) = 2x^3 + 6x^2 - x + 4$ $\qquad 2x^3 + 6x^2 - x + 4 = 0$

$f(x) = 2x^4 + x^2 - 10$ $\qquad 2x^4 + x^2 - 10 = 0$

$f(x) = 2x^4 + 6x^2 - 5$ $\qquad 2x^4 + 6x^2 - 5 = 0$

$f(x) = x^6 - 5x^5 + 9x^4 - 15x^3 + 20x^2 - 10x + 12$

$x^6 - 5x^5 + 9x^4 - 15x^3 + 20x^2 - 10 x +$

BUSINESS PROBLEMS

24. **Social Security tax.** Write a program for calculating Social Security tax (also called FICA tax), if the tax rate is 4.8% of gross salary on the first $7800 earned in a calendar year. There is no FICA tax on earnings over $7800.

Column #1 is the total salary paid prior to the present pay period since the beginning of the year. Column #2 is the earnings for the present pay period.

Col #1	Col #2
$5205.00	$520.50
$6147.00	$614.70
$8814.00	$881.40
$7500.00	$750.00
$7788.00	$778.80
$4380.00	$560.00
$8450.00	$983.00
$7150.00	$665.50
$7200.00	$718.00
$7000.00	$800.00

25. **Federal income tax.** Write a program for calculating Federal income tax on income between $6000 and $16,000 given gross income and number of dependents if

$$TI = GI - (625)(DEP) - (10\%)(GI)$$

where TI is taxable income;

GI is gross income;

DEP is total number of dependents and exemptions;

The tax table is as follows:

Taxable income	Tax is
$6,000 to $8,000	$1130 plus 25% of amount over $6,000
$8,000 to $10,000	$1630 plus 28% of amount over $8,000
$10,000 to $12,000	$2190 plus 32% of amount over $10,000
$12,000 to $14,000	$2830 plus 36% of amount over $12,000
$14,000 to $16,000	$3550 plus 39% of amount over $14,000

DATA: Gross Income	DEP
$16,000	2
15,570	1
15,250	4
14,780	2
13,500	2
13,125	7
12,800	2
11,819	4
8,000	6
7,250	1
10,777	5

26. State income tax. Write a program for calculating New York State income tax on incomes between $7000 and $17,000 given gross income and total number of dependents and exemptions given in exercise 25, when the tax table is as follows:

Taxable income	Tax is
$7,000 to $9,000	$260 plus 6% of amount over $7,000
$9,000 to $11,000	$380 plus 7% of amount over $9,000
$11,000 to $13,000	$520 plus 8% of amount over $11,000
$13,000 to $15,000	$680 plus 9% of amount over $13,000
$15,000 to $17,000	$860 plus 10% of amount over $15,000

27. Mortgage amortization.

a) Write a program to calculate the number of monthly payments of $230 each necessary to pay off an $11,000 mortgage if the interest rate is 7.5%. The monthly payment of $230 includes interest.

b) Write a program to calculate the amount of mortgage a person could afford if the interest rate is 6.5% and he can afford payments of $200 a month, including both principal and interest.

CHAPTER 4

THE FOR LOOP

WRITING A LEVEL FOUR PROGRAM

4.1 THE FOR AND NEXT STATEMENTS

```
10 Y = 0
20 Y = Y + 2
30 IF Y>100 THEN 999
40 S = Y↑2
50 PRINT Y, S
60 GO TO 20
999 END
```

```
10 FOR Y = 2 to 100 STEP 2
20 S = Y↑2
30  PRINT Y, S
40 NEXT Y
999 END
```

The program on the left above will print every even number from 2 to 100 inclusive and its square. In the program at the right above, the same thing is accomplished in a shorter program by use of the FOR and NEXT statements. The program lines in the range of the FOR statement are executed repeatedly beginning with Y = 2. Y is then incremented by 2 until the value of Y passes 100. The general form of the FOR statement is FOR A = B TO C STEP D.

The variable A can be any scalar variable. The program lines in the range of the FOR statement are executed repeatedly, beginning with A = B. Thereafter, A is incremented by D until the value passes the limit specified by C. The STEP portion of the statement may be positive or negative or may be omitted. If omitted, a step size of +1 is assumed. B, C, and D may be any valid expressions for which values can be calculated. Examples of some FOR statements are as follows:

```
FOR X = 1 to 75
FOR Y = .1 to F STEP .05
FOR W6 = 3*R to -sin(R) STEP -2
FOR Y9 = P to (C + D)/5 STEP 5*J + INT(Q)
```

FOR and NEXT statements form a pair. For every FOR statement at the beginning of the loop there must be a NEXT statement at the end of the loop that names the same variable.

The FOR statement is used at the beginning of the loop and the NEXT statement must always be used at the end of the loop.

It is often possible to type the entire loop on a single line. For example:

```
10 FOR Y = 2 TO 100 STEP 2:  S = Y↑2:  PRINT Y,S:  NEXT Y
```

The disadvantage of doing this is that if the line contains an error, the entire line must be retyped. If the loop is typed on four separate lines, correcting a single error is much easier. Rememb er, the END statement is optional.

If the line number is left out, the program will be executed in immediate mode:

```
FOR Y=2 to 100 STEP 2:  S=Y  2: PRINT Y,S: NEXT Y
```

Spaces are optional. The above program may be written:

```
FORY=2TO100STEP2:S=Y↑2:PRINTY,S:NEXTY
```

The following is a program for calculating and printing the values of the sine, cosine, and tangent of all angles from 0 to 360 degrees inclusive in steps of one degree:

```
5 REM  D IS ANGLE IN DEGREES.  R IS ANGLE IN RADIANS
10 FOR D = 0 TO 360
20 LET R = D/57.2958
30 LET S = SIN(R)
40 LET C = COS(R)
50 LET T = TAN(R)
60 PRINT D, S, C, T
70 NEXT D
80 END
```

This same program may be shortened to:

```
10 FOR D = 0 to 360
20 R = D/57.2958
30 PRINT D, SIN(R), COS(R), TAN(R)
40 NEXT D
```

This program may also be typed all on one line:

```
10FOR D=0 TO 360:R=D/57.2958:PRINT D, SIN(R), COS(R), TAN(R):NEXT D
```

If illegal values are assigned to the variables in a loop or if the loop designated by STEP is in the wrong direction or zero, the loop is executed only once. Examples of invalid values are:

```
FOR X = 1 TO 50 STEP -5
FOR X = -1 to-100 STEP 5
FOR X = 1 TO 100 STEP 0
```

There is no limit to the number of statements that can be between the FOR statement and the NEXT statement ending that loop.

Branching *into* the range of a FOR loop from outside is not permissible.

Branching out of the range of a FOR loop *is* permissible.

EXERCISES 4.1

1. Some of the following programs contain one or more errors. Which of the programs contain the error or errors? Rewrite each erroneous program correctly.

a)
```
10 FOR X = 0 TO -53 STEP -2   minus sign
20     Y = X*2 + X↑2
30     PRINT X,Y
40 NEXT X
50 END
```

b)
```
10 FOR F = -42 TO 212 STEP .01
20     C = 5/9 * (F - 32)
30     PRINT F;"DEGREES F.", C;"DEGREES C."
40 NEXT C —— NEXT F
50 END
```

c)
```
10 FOR P = 10000 TO 20000 STEP 100
20     I = P * .0735/12)
30     IF I > 100 THEN 100
40 NEXT P
100 PRINT"INTEREST EXCEEDS $100 WHEN P BECOMES";P
110 END
```

d)
```
10 FOR R=-3.1416 TO 3.1416 STEP .01:PRINT R,SIN(R):NEXT R:END
```

e)
```
10S=0:FORN=1TO324:S=S+N:NEXTN:PRINTS:END
```

2. What would actually be accomplished by the program in Exercise 1(e)?

4.2 NESTED LOOPS.

One loop may be contained completely within another loop. Such loops are called nested loops. In the program below, the values of X^2, X^3, X^4, and X^5 are calculated and printed out for the values of X from .1 to 1 in increments of .1.

```
10 FOR X = .1 TO 1 STEP .1
20 FOR P = 2 TO 5
30 LET Y = X↑P
40 PRINT Y,
50 NEXT P
60 NEXT X
70 END
```

It is common practice to indent the statements contained in a loop for clarity. Loops can then be more easily distinguished. This especially true for nested loops. The program above could have been written:

```
10  FOR X = .1 to 1 STEP .1
20     FOR P = 2 TO 5
30      LET Y = X↑P
40      PRINT Y,
50     NEXT P
60  NEXT X
70  END
```

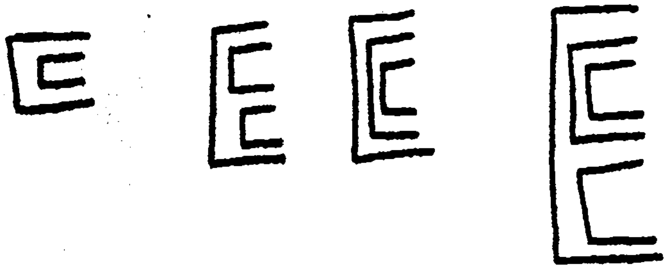
The inner loop will be executed four times for each time the outer loop is executed. The outer loop will be execut ten times so the inner loop will be executed a total of 40 times, and there will be 40 values of Y printed out.

Notice the comma at the end of the PRINT statement in line 40. If this comma were not there, one number would be printed out on each line. By placing the comma there, one number will be printed in each print zone. If line 40 had been written

```
40   PRINT Y;
```

with a semicolon at the end, the numbers would have been packed closer together with more numbers on a line.

Loops may be nested with no limit. However, great care must be taken not to branch into the range of a FOR loop from outside. Below are schematic diagrams of valid FOR loops.



EXERCISES 4.2

Indicate which of the following programs contain errors. Correct the erroneous programs.

1.
```
10 FOR A = 1 TO 10
20   FOR B = -6 to -4 STEP -.5   +
30      C = 3*A + 4*B
40      PRINT A, B, C
50 NEXT A
60   NEXT B
70 END
```

2.
```
   R
FOX=-20TO20:FORY=4TO7:Z=3*X*Y:PRINTX,Y,Z:NEXTY:NEXTX:END
```

3.
```
10 FOR C = 10 TO 20 STEP .5
20   FOR D = 1 TO 5
30      FOR E = -2 TO -3 STEP .1   - sign
40         F = C↑D + C * E
50         PRINT C,D,E,F
60      NEXT E
70   NEXT D
80 NEXT C
90 END
```

```
3.  10 FOR K = 1 TO 100:FOR L = 1 TO 20:FOR M = 1 TO 5
    20 N = K + L + M:PRINT K,L,M,N
    30 NEXTL: NEXTM: NEXT K:END

4.  10 S1=0:S2=0:S3=0
    20 FOR U = 1 TO 5
    30    FOR V = 1 TO 8
    40      W = U*V
    50      S1 = S1 + W
    60    NEXT V
    70    FOR X = 1 TO 11
    80       Y = X↑2 + S1
    90       S2 = S2 + Y
    100   NEXT X
    110   S3 = S3 + S1 + S2
    120 NEXT U
    130 PRINT S3
    140 END

5. 5READ A,B,C,D,E,F
   10 FOR X = A TO B STEP C: FOR Y = D TO E STEP F
   20 Z = X + Y:PRINT"X=";X,"Y=";Y,"Z=";Z
   30 NEXT Y: NEXT X
   40 DATA 132,  164,  2,  256,  744,  12
   50 END
```

4.3 SHARING OF COMMON DATA.

The value of variables in one program can be saved for use in a subsequent program.

To do this, a COM statement is entered as the first numbered line in both programs. A program may contain several COM statements just as long as they are entered as the first numbered executable lines.

The first command in the subsequent program is RESTART. The RESTART command will remove all user program text and noncommon variables from the system, but common variables and their values are not changed.

This is very useful when a program is too long or too complicated for the storage capacity of the system and must be broken down into two or more shorter or simpler programs

On the next page is an example of how a long, complicated program may be broken into three smaller programs connected by COM statements.

```
START
10 REM   THIS IS THE FIRST PROGRAM
20 COM  A, B, C, R1, R2
     }          program statements
RUN
RESTART
10 REM   THIS IS THE SECOND PROGRAM
20 COM  A, B, C, R1, R2
30 COM Q, R, T
     }          program statements
RUN
RESTART
10 REM   THIS IS THE THIRD PROGRAM
20 COM Q, S, T
     }          program statements
RUN
```

The first time RESTART is used above, it will remove all the first program text and variables except the values of A, B, C, R1 and R2. These will be saved for used in the second program. The second time RESTART is used above, all the second program text and variables will be removed except for the values of Q, S, and T which will be saved and can be used in the third program.

4.4 THE RERUN COMMAND.

GOTO starting line no.

If the RERUN command is used in place of the RUN command, the entire program will be run from the beginning but the values of the variables will not be first reset to zero. The values of the variables will be left at their current setting.

The RERUN command may be used to run a portion of a program. RERUN 80 means to rerun the program beginning with program line 80.

RERUN is a command. Therefore, it has no line number.

4.5 WRITING A LEVEL FOUR PROGRAM.

A level four program contains one or more FOR loops. A typical level four program is as follows:

```
10  REM    Harry Ackerman  Level four     11/19/72
20  REM   TABLE OF VALUES OF SINE FROM 9 TO 10 DEGREES
30  REM   INCREMENTS ARE ONE MINUTE
40  FOR D = 9 TO 10 STEP 1/60
50      R = D/57.2958
60      PRINT D, R, SIN(R)
70  NEXT D
80 END
```

EXERCISES 4.5

1. Write a program for calculating the values of the sine of every angle from 12° 13' 14" to 13° 14' 17" in increments of one second,

2. Write a program for calculating the areas of all rectangles as the length varies from 123 to 156 in increments of 1, and as the width varies from 17 ti 19 in increments of .1.

3. Write a program for calculating the areas of all triangles as side a varies from 10 to 15, side b varies from 17 to 30, and as side c varies from 9 to 23, all in increments of 1.

4. Write a program to calculate the products XY as X varies from -36.17 to -34.19 in increments of .04 and as Y varies from -6 to 58 in increments of 2.

5. Write a program to print out the temperature in degrees Centigrade and degrees Fahrenheit as the temperature variesffrom -58° C. to -38° C. in increments of $.1^{\circ}$ C.

6. Salaries are being increased by 7.5%. Write a program for printing out the old and the new salaries from $10,500 to $13,700 in increments of $100.

7. Write a program for evaluating the function $Y = 3x^3 - 2x^2 + 5x - 9$ for each value of x from -3 to +3 in increments of .1.

CHAPTER 5

ONE-DIMENSIONAL ARRAYS

WRITING A LEVEL FIVE PROGRAM

5.1 ARRAY VARIABLE NAMES.

An array is a set of storage locations in computer memory which is reserved for a list of values. The array must be given a name, which may be any one of the 26 letters of the alphabet.

An array variable consists of the variable name followed by an expression in parentheses which indicates an individual storage space. The array variable A95) indicates the fifth storage space in the array whose name is A. A(10) indicates the tenth storage space of array A, and so forth.

The expression in the parentheses may be any valid BASIC expression whose value can be calculated. For example, the following are valid array variables, which may also be called subscripted variables:

B(K), C(K*2), D(X+4), W(T/I+7)

The numeric value of the first subscript must be 1. A subscript of zero is not allowed. A(0) is invalid.

5.2 THE DIMENSION STATEMENT.

```
10 DIM A(5), B(7), C(10)
```

The DIM statement reserves space for array variables which will be seferenced in the program. Space may be reserved for more than one variable with a single DIM statement by separating them with commas as shown above.

The dimension of an array cannot exceed 255.

The space to be reserved must be indicated by an integer. Expressions are not allowed. Arrays may be dimensioned larger than necessary.

DIM statements must appear before any use of the variable in the program. It is common practice to place all DIM statements at the beginning of the program before any executable program statements.

In the example above, 5 storage locations are reserved for the numbers in list A, 7 for the numbers in list B, and 10 for the numbers in list C.

5.3 READING AN ARRAY.

```
10 DIM A(5), B(7), C(10)
20 FOR I = 1 TO 5:   READ A(I):   NEXT I
30 FOR J = 1 TO 7:   READ B(J):   NEXT J
40 FOR K = 1 TO 10:  READ C(K):   NEXT K
50 DATA 23,  45,  67,  89,  53,  72,  90,  45,  99
51 DATA 88,  77,  77,  55,  99,  33,  35,  37,  38
52 DATA 36,  45,  55,  32
```

The above program will cause the first five numbers to be read into array A, the next seven numbers to be read into array B, and the next ten numbers to be read into array C.

The values in each of the array locations would then be as follows:

A(1)	23	B(1)	72	C(1)	55
A(2)	45	B(2)	90	C(2)	99
A(3)	67	B(3)	45	C(3)	33
A(4)	89	B(4)	99	C(4)	35
A(5)	53	B(5)	88	C(5)	37
		B(6)	77	C(6)	38
		B(7)	77	C(7)	36
				C(8)	45
				C(9)	55
				C(10)	32

If the lists that are to be read into the arrays all have exactly the same number of elements, there are two ways of arranging the data and reading the lists into the array positions.

```
10 DIM D(5), E(5), F(5)
20 FOR X = 1 TO 5:  READ D(X):  NEXT X
30 FOR X = 1 TO 5:  READ E(X):  NEXT X
40 FOR X = 1 TO 5:  READF(X):   NEXT X
50 DATA 2,4,6,8,10,   1,3,5,7,9,   20,30,40,50,60
```

When the above program is executed, the first five data items will be read into array D, the next five into array E, and the next five into array F.

In the program below, the data items have been rearranged. The first item is the first number in list D, next is the first item in list E, next is the first item in list F, then the second item from list D, E, and F respectively, and so on. This arrangement of the data permits a shorter program.

```
10 DIM  D(5),  E(5),  F(5)
20 FOR X = 1 TO 5: READ D(X), E(X), F(X): NEXT X
30 DATA 2, 1, 20, 4, 2, 30, 6, 3, 40, 8, 4, 50, 10, 5, 60
```

EXERCISES 5.3

1. Write a program for reading in the following lists into the proper array positions.

List A	List B	List C
1	17.3	25.5
3	18.6	16.2
9	19.9	
	20.3	

2. Write a program for reading in the following lists into proper array positions, using only one FOR loop in the program.

List K	List W	List V
186	666	1874
192	159	1923
213	367	1848
144	214	1971

3. Find the error in the following program:

```
                 7
10 DIM F(5),  G(5),  H(5)
20 FOR X = 1 TO 5:  READ F(X):  NEXT X
30 FOR X = 1 TO 7:  READ G(X):  NEXT X
40 FOR X = 1 TO 4:  READ H(X):  NEXT X
60 DATA 23,24,25,26,27,28,1,2,3,4,5,6,100,200,300,400
```

4. Find the error in the following program:

```
DIMB(9):FORX=1TO9:  READB(X):NEXT X:DATA9,8,7,6,5,4,3,2,1
```

5. A student prepared the following program:

```
10 DIM B(6),  N(6),  P(6)
20 FOR X = 1 TO 6:  READ B(X):  NEXT X
30 FOR Y = 1 TO 6:  READ N(Y):  NEXT Y
40 FOR Z = 1 TO 6:  READ P(Z):  NEXT Z
50 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
```

He then rewrote the program as:

```
10 DIM B(6),  N(6),  P(6)
20 FOR X = 1 TO 6:  READ B(X),N(X),P(X):  NEXT X
30 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
```

Are these two programs equivalent? That is, will the two programs each read in the same numbers into the same positions of the three arrays?

6. Is the program

```
10 FOR X = 6 TO 1 :  READ A(X):  NEXT X
20 DATA 16, 18, 20, 22, 24, 26
```

equivalent to the program

```
10 FOR X = 1 TO 6:  READ A(X):  NEXT X
20 DATA 26, 24, 22, 20, 18, 16
```

5.4 CALCULATING WITH SUBSCRIPTED VARIABLES.

```
10  DIM D(5), E(5), F(5)
20 READ D(I), E(I)
15 FOR I = 1 TO 5
30 LET F(I) = D(I) * E(I)
40 NEXT I
50 DATA 1.23, 23, 1.89, 16, 5.24, 5, 3.25, 40, 4.68, 33
```

As a result of executing the above statements, the system will have the following values in array F:

F(1) will be the product of 1.23 and 23

F(2) will be the product of 1.89 and 16

F(3) will be the product of 5.24 and 5

F(4) will be the product of 3.25 and 40

F(5) will be the product of 4.68 and 33

The value of the index variable in a FOR loop must not be changed by a program line within a loop. However, the index variable may be used as either a subscript or as a value or as both.

```
10 DIM A(10),B(10),F(10)
20 FOR X = 1 TO14: READ A(X), B(X): NEXT X
30 FOR K = 3 TO 13 STEP 2
40 LET L = K + 1
50 LET F(K) = A(K) + B(L) + 2 * K
60 PRINT A(K), B(L), F(K)
70 NEXT K
80 DATA    (at least 14 data items)
90 END
```

The statements in the previous program are valid because the value of K is not changed by any statement <u>within</u> the loop. It is only changed by the nature of the FOR and NEXT statements. Notice that K is used both as a subscript and as a value in line 50.

The following program is invalid because the value of the index variable K is changed by a program statement within the loop, statement on line 30.

```
20 FOR K = 3 TO 13 STEP 2
30 LET K = K + 1                              INVALID
40 LET F(K) = A(K) + B(K) + 2 * K
50 NEXT K
```

The following is a simple program for finding the sum of all the numbers in an array:

```
S = 0:  FOR J = 1 TO 50:  S = S + F(J):  NEXT J
```

EXERCISES 5.4

1. Find the error or errors in the following:

```
5 DIM A(14), B(20), C(14)
10 FOR I = 1 TO 14
20 READ A(I), B(I)
30 NEXT I
40 FOR I = 15 TO 20
50 READ B(I)
60 NEXT I
70 FOR I = 1 TO 14
75 J = I + 6            if I=1, J-7=ϕ - not allowed in WANG BASIC
80 C(I) = A(J-7) * B(J) + B(J-2) - 3*I + I/J
90 NEXT J
100 DATA (34 data items)
```

2. Write a program that will:

   read in 10 data items into array C;

   read in 10 data items into array D;

   calculate the first element in array E by adding the first element in array C to the last element in array D;

   calculate the second element in array E by adding the second element in array C to the ninth element in array D;

   and so forth.

3. Write a program that will:

read in 100 data items into array K;

read in 100 data items into array L;

calculate the single sum of the 100 products

K(X) * L(X) for all X 1 to 100.

4. Write a program that will:

read in 80 data items into array M;

calculate the sum of all the numbers in the even-numbered positions (second, fourth, sixth);

calculate the sum of all the numbers in the odd-numbered positions (first, third, fifth);

calculate the single sum of all the 40 products first number times second, third times fourth, fifth times sixth, and so forth.

5. Write a program that will:

read in five numbers, calculate their average, place the average in A(1);

read in five more numbers, calculate their average, place their average in A(2);

continue until a total of 25 sets of five numbers each have been read in, the average fcr each set of five numbers calculated and placed in array A;

calculate the average of all the numbers in array A.

5.5 PRINTING OUT ARRAYS.

```
110 PRINT "LIST D",  "LIST E",  "LIST F"
120 FOR I = 1 TO 57: PRINT D(I), E(I), F(I):  NEXT I
```

These two statements above will cause three column headings to be printed out and under each column heading in a vertical column the appropriate array will be printed out.

```
220 PRINT "LIST D": FOR I = 1 TO 57: PRINT D(I);:NEXT I
230 PRINT "LIST E": FOR J = 1 TO 57: PRINT E(J);:NEXT J
240 PRINT "LIST F": FOR K = 1 TO 57: PRINT F(K);:NEXT K
```

These three statements above will cause each array to be printed out horizontally following the identifying message which is the name of the list. The semicolon at the end of each of the three print statements will cause the numbers in the printout to be "packed". A comma in place of the semicolon would cause one number to be printed in each print zone for that PRINT statement.

The following are examples of use of PRINT USING and image statements in printing out arrays.

```
80 PRINT USING 90
90 % LIST A              LIST B              LIST C

100 FOR X = 1 TO 88
150 PRINT USING 160 A(X), B(X), C(X)
160 %###.##              ###.##              ###,##
170 NEXT X
```

EXERCISES 5.5

1. Assume that each of the arrays P, R, T, and I contain 100 numbers. Write the print statements that will cause the arrays to be printed out vertically under the respective headings PRINCIPAL, RATE, TIME, INTEREST.

2. Assume array W contains 48 numbers. Write the print statements to have the numbers printed out four to a row under the column headings WEEK 1, WEEK 2, WEEK 3, WEEK 4.

3. If array K contains the numbers, in order,

   26, 42, 38, 91, 88, 66, 77, 12, 13, 34, 56, 78

   what would be the printout of

```
90 FOR X = 1 TO 12 STEP 4
100 PRINT K(X), K(X + 1), K(X + 2), K(X + 3)
110 NEXT X
```

4. What PRINT USING and image statements would cause:

   the headings LENGTH, WIDTH, AREA to be printed beginning in columns 10, 20, 30 respectively;

   round the numbers in arrays L, W and A to the nearest whole number and printed out in vertical columns (maximum of five digits) under the appropriate column heading?

5.6 STRING VARIABLES IN ARRAYS.

String variables are read into arrays and printed out from arrays much the same way as scalar variables.

```
10 DIM A$(5), B(5)
20 FOR X = 1 TO 5
30  READ A$(X), T1, T2, T3, T4, T5
40 B(X) = (T1 + T2 + T3 + T4 + T5)/5
50 NEXT X
60 PRINT "STUDENT", "AVERAGE"
70 FOR X = 1 TO 5:  PRINT A$(X), B(X):  NEXT X
80 DATA "JOHN ADAMS", 87, 86, 80, 90, 100
81 DATA "GEORGE RAFT", 65, 76, 66, 79, 65
82 DATA "ERIC NOTT", 99, 98, 89, 95, 97
83 DATA "TERRY LADD", 88, 85, 86, 80, 79
84 DATA "THOMAS BRANDT", 77, 79, 70, 69, 55
85 END
```

This program will cause the students' names to be read into array A$. The five test scores for each student are read, added, divided by 5, and the result placed in array B. When array A$ and array B are printed out, the result will be the student's name followed by his average for the five test scores.

EXERCISES 5.6

1. The purpose of the program below is to read into array S$ the name of each student and to read into array I the grade index for that student. The numbers in array I are then examined and the greatest index determined. The name of the student with the highest index and his index are then printed out.

The program contains errors. Indicate the errors and rewrite the program correctly.

```
10 DIM S$(20), I(20)
20 M = -1.0E+60
30 FOR X = 1 TO ~~20~~ 12:  READ S$(X), I(X)
50 IF I(X) > M THEN 70:  GO TO 90      IF I(X) <= M THEN 90
70 M = I(X):  N$ = S$(X)
90 NEXT X
100 PRINT "STUDENT", "INDEX":  PRINT N$, M
120 DATA LUCILLE HENRY, 1.456,  ADAM WILLIAMS,  3.879
121 DATA HAROLD HOLISTER,  2.367,  DANIEL MCALSTER, 2.987
122 DATA MARY LAMB, 3.334, JOAN ALLEN, 3.583
123 DATA KURT WHISKER, 3.997, HELEN ALLEN, 2.955
124 DATA GEORGE WEEIL, 1.580,  SUSAN WHITE, 3.888
125 DATA MILDRED CARON, 3.400, JOHN MARS, 3.750

130 END
```

5.7 SHARING COMMON ARRAYS.

```
10 COM A(20), B$(10), D, E
```

If a common variable is an array variable, the COM statement must provide array definition the same as a DIM statement would. The DIM statement for common array variables is then not needed. A COM statement may contain both array variables and scalar variables, as shown above.

5.8 THE TAB FUNCTION.

```
100 PRINT TAB(10); "SALESMAN";TAB(23);"SALES";TAB(33)"COMMISSION"
200 PRINT TAB(5);A(X);TAB(23);S;TAB(35);C
```

The TAB function permits tabulated formatting. The statement in line 100 above will cause the column headings SALESMAN, SALES, and COMMISSION to be printed beginning in columns 10, 23, 33 respectively. The statement in line 200 above will cause the salesman's name to be printed beginning in column 5, the value of S to be printed beginning in column 23, and the value of C to be printed beginning in column 35.

Positions are numbered 0 to 69 on the teletype. The value of the expression in the parentheses is computed and the integer part taken. The typewriter is then moved to this position. If it has already passed this position, the TAB is ignored.

If the value of the expression is greater than 69, the typewriter will move to the beginning of the next line. The following program will cause the plotting of the sine curve for all angles from zero to 360 degrees in steps of 10 degrees.

```
100 FOR D = 0 TO 360 STEP 10
110 R = D/57.2958
120 PRINT TAB (30 + 30* SIN(R)); "*"
130 NEXT D
```

The measure of the angle in degrees is converted to measure in radians. The value of the sine of that angle is computed. The value of the sine of an angle ranges from -1 to +1 inclusive. Multiplying this by 30 gives a value from -30 to + 30 inclusive. Since the positions to which the typewriter can tab go only from 0 to 69, at least 30 must be added to this computed value so that the value will ne non-negative. When the value of the sin is zero, the typewriter will tab to column 30. When its value is -1, it will tab to column 0, and when the value is +1, it will tab to column 60.

The above program may be modified to provide for an axis to be printed in column 30.

```
10 FOR D = 0 TO 360 STEP 10
20 R = D/57.2958
30 S = (30 + 30*SIN(R))
40 IF S <30 THEN 60
50 PRINT TAB(30);"."; TAB(S); "*";  NEXT D
55 GO TO 70
60 PRINT TAB(S); "*"; TAB(30): ".";  NEXT D
70 END
```

The value of the sine of the angle will be indicated by an asterisk. The axis in column 30 will be indicated by periods. Since the values increase from left to right, after the graph is plotted, it must be rotated 90°. The axis that was printed vertically in column 30 will then become the horizontal x-axis.

The following statement would cause another axis to be printed out at right angles to the x-axis.

```
1  A$ = "..................": B$=A$:  C$=A$:  D$=A$
2  PRINT A$, B$, C$, D$
```

Although this will be printed horizontally on the page, after it is rotated 90° it will serve as the vertical y-axis.

If an entire graphing program can be entered on one line, the program may be executed in immediate mode.

```
FOR D=0 TO 360:PRINT TAB(30+30*SIN(D/57.2958));"*":NEXT D
```

EXERCISES 5.8

1. Complete the following program below using one or more PRINT TAB and TAB statements so that column headings VALUE OF X, X SQUARED, and X CUBED will be printed out in columns 0, 20, and 40 respectively, and arrays A, B, and C will be printed out vertically under the column headings beginning in columns 4, 23, and 40 respectively.

```
5 DIM A(100), B(100), C(100)
10 FOR X = 1 TO 100: A(X) =:  B(X) = X↑2:  C(X) = X↑3
20 NEXT X
```

2. What letter of the alphabet would be printed out by the followimg program?

```
10 PRINT TAB(10); "X"; TAB(18); "X"
11 PRINT TAB(12); "X"; TAB(16); "X"
12 PRINT TAB (14); "X"
13 PRINT TAB (12); "X"; TAB (16); "X"
14 PRINT TAB (10); "X"; TAB (18): "X"
```

3. What letter of the alphabet would be printed out by the following program?

```
10 PRINT TAB (10); ".........."
20 FOR X = 20 TO 10:  PRINT TAB (X); ".":  NEXT X
30 PRINT TAB (10); ".........."
```

3.
```
10 FOR D = 0 TO 360 STEP 5
20 R = D/57.2958
30 C = COS(R)
```

a) Complete the above program using one or more PRINT TAB and TAB statements so that an asterisk will be printed in column 25 when the value of the cosine is zero, in column 50 when the value of the cosine is +1, in column zero when the value of the cosine is -1, and in the appropriate column when the cosine takes on the other calculated values.

b) Modify the program so that an axis consisting of periods is printed in column 25.

c) Modify the program further so that the values -1, -.5, 0, +.5, and +1 are printed above the appropriate columns.

3. Write a program that will plot the values of sine x, cosine x, and an axis all on the same graph for all values of x from zero to 360 degrees in steps of 10 degrees.

4. The program below was intended for printing a plot of the graph of the cosine curve for all values of x from 10 to 15 degrees in increments of one minute of degree. Describe the errors in the program.

```
10 FOR X = 10 TO 15 STEP 1/60: C = COS(X/57.2958)
20 PRINT TAB(C); "*":  NEXT X
```

5.9 ARRANGING ARRAY ITEMS IN SEQUENCE.

There are several methods of taking array items and arranging them in numeric sequence or alphabetical order. One method of arranging them in numeric sequence is as follows:

1. Read in the data items into array A.
2. Set array B identical to array A.
3. Determine the greatest number in array A. Place this number in position C(1). Substitute -1.0E+60 for this greatest number in array A.
4. Determine the greatest remaining number in array A. Place this number in the next available position in array C. Substitute -1.0E+60 for this greatest remaining number in array A.
5. Repeat step 4 until all the numbers in array A have been replaced with -1.0E+60 and each number in array A has been transferred to array C in descending order.
6. Array A will now consist entirely of -1.0E+60 in every position. Array.B will be identical to the original array A. Array C will contain in descending order all the numbers that were originally in Array A.
7. Print out array B and array C.

Below is a program for ordering up to 100 numbers in sequence. N is the number of numbers to be ordered. P is the position of the greatest remaining number in array A. In the example below, there are 87 numbers to be ordered.

```
10 DIM a(100), B(100), C(100)
20 READ N
30 FOR X = 1 TO N: READ A(X):  B(X) = A(X):  NEXT X
40 FOR J = 1 TO N
50     M = A(1)
60     P = 1
70          FOR I = 2 TO N
80               IF A(I)~~>~~<=M THEN ~~100~~ 120
90               ~~GO TO 120~~
100              M = A(I)
110              P = I
120         NEXT I
130     C(J) = M
140     A(P) = -1.0E+60
150 NEXT J
160 FOR K = 1 TO N:  PRINT B(K), C(K)
170   DATA 87
171 DATA (87 numeric data items)
200 END
```

If the array items are alphanumeric, the previous program may be easily adapted to arranging the items in alphabetical order.

```
10 DIM A$(100), B$(100), C$(100)
20 READ N
30 FOR X = 1 TO N:  READ A$(X), B$(X):  B$(X) = A$(X): N EXT X
40 FOR J = 1 TO N
50      M$ = A$(1)
60      P = 1
70           FOR I = 2 TO N
80                IF A$(I)<M$ THEN 100
90                GO TO 120
100               M$ = A$(I)
110               P = I
120          NEXT I
130     C$(J) = M$
140     A$(P) = "AAAAA"
150 NEXT J
160 FOR K = 1 TO N:  PRINT B$(K), C$(K)
170 DATA 87
171 DATA (87 alphanumeric data items)
200 END
```

5.10 WRITING A LEVEL FIVE PROGRAM.

A level five program contains one or more arrays.

A typical level five program is as follows:

```
10 REM    MARY ANDERSON    LEVEL FIVE    12/8/72

20 REM    USE OF ONE-DIMENSIONAL ARRAYS
25 DIM A(10), B(10), C(10)
30 PRINT TAB(10);"LIST A";TAB(20);"list B";TAB(30);"LIST C"

40 FOR X = 1 TO 10

50     READ A(X), B(X)

60      J = 11 - X

70     C(X) = A(X) * B(J)

80 NEXT X

90 FOR K = 1 TO 10

100    PRINT TAB(10);A(X);TAB(20);B(X);TAB(30);C(X)

110 NEXT K

120 END

115 DATA 1234, 56, 3456, 53, 4582, 68, 1753, 88, 1000, 10

116 DATA 4739, 97, 2567, 44, 6541, 85, 2439, 9375, 99, 86
```

CHAPTER 6

TWO-DIMENSIONAL ARRAYS

WRITING A LEVEL SIX PROGRAM

6.1 NAMING TWO-DIMENSIONAL ARRAYS.

A two-dimensional array can be considered to consist of horizontal rows and vertical columns. It may be called a table. The name of an array consists of a single letter. The variable name for any one position in the array or table consists of the array name followed by parentheses containing two expressions, called subscripts, separated by a comma. The first subscript refers to the row number and the second subscript refers to the column number.

S(3,5) refers to the position in array S that is in both the third row and the fifth column. If array S consists of three rows and five columns, the positions of the items are as follows:

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

This is the same convention as that used in identifying locations in determinants and matrices.

Subscripts may be expressions. Examples of valid names for subscripts are:

```
S(x+1,Y-2)    B(W+G,V*R)    T(9,V↑2)
```

6.2 DIMENSIONING AN ARRAY.

A DIM statement statement must appear before use of a subscripted variable in a program. The DIM statement reserves space for the array. The DIM statement below would reserve space for a two-dimensional array of four rows and nine columns, called U; for a two-dimensional array of ten rows and ten columns, called V; and for a one-dimensional array having 56 elements, called W.

```
10 DIM U(4,9), V(10,10), W(56)
```

Arrays that are common to subsequent or previous programs are dimensioned in a COM statement and not in a DIM statement:

```
10 COM A(34), B(9,12)
```

6.3 READING IN A TWO-DIMENSIONAL ARRAY.

```
10 DIM A(5,3)                       10 DIM A(5,3)

20 FOR I = 1 TO 5                   20 FOR J = 1 TO 3

30    FOR J = 1 TO 3                30    FOR I = 1 TO 5

40         READ(I,J)                40         READ A(I,J)

50    NEXT J                        50    NEXT I

60 NEXT I                           60 NEXT J

70 DATA 8,98,765,45                 70 DATA 8,98,765,45

71 DATA 42,58,69,101                71 DATA 42,58,69,101

72 DATA 333,454,676,99              72 DATA 333, 454,676
73 DATA 112,377,100                 73 DATA 112,377,100
80 END                              80 END
```

Two-dimensional arrays may be read in by use of nested FOR loops as shown above. In the example at the left, the numbers are read in one <u>row</u> at a time. In the example at the right, the numbers are read in one <u>column</u> at a time.

Use of the program at the left would result in the following array:

```
  8       98      765

 45       42       58

 69      101      333

454      676       99

112      377      100
```

Use of the program at the right would result in the following array:

```
  8       58      676
 98       69       99
765      101      112
 45      333      377
 42      454      100
```

Care must be taken in reading in an array to be sure the numbers will be located in the manner the user wants.

A program such as the following may be used to set all the values of an array to a given value such as zero:

```
10 DIM F(5,8)
20 FOR I = 1 TO 5
30    FOR J = 1 TO 8
40        F(I,J) = 0
50    NEXT J
60 NEXT I
```

Statements 20 through 60 could all be written on one line:

```
20 FOR I=1 TO 5: FOR J=1 TO 8: F(I,J)=0: NEXT J: NEXT I
```

EXERCISES 6.3

1. Write a program that would cause the data in data statements 100 through 103 to be read into array A in the positions indicated below:

```
100 DATA 18, 24, 63
101 DATA 19, 66, 84, 77, 56, 18
102 DATA 23, 44, 36, 17
103 DATA 45, 89, 65, 74, 86
```

18	24	63	19	66	84
77	56	18	23	44	36
17	45	88	65	74	86

2. Write a program that would cause the data in data Statements 100 through 103 in Exercise #1 to be read into array A in the psoitions indicated below:

18	77	17
24	56	45
63	18	88
19	23	65
66	44	74
84	36	86

6.4 CALCULATING WITH DOUBLE SUBSCRIPTED VARIABLES.

The index variable of a FOR loop may be used as a subscript, as a value, or as both. In the following program, each element in array D is multiplied by the corresponding element in array E. Twice the row number and three times the column number are then added to this product. The result is stored in array F.

```
10 DIM D(3,4), E(3,4), F(3,4)
20 FOR K = 1 TO 3
30    FOR L = 1 TO 4
40        LET F(K,L) = D(K,L) * E(K,L) + 2*K + 3*L
50    NEXT L
60 NEXT K
```

Remember to set up and dimension with a DIM statement the array where the result of the calculation is to be placed.

If the arrays are quite large, and there is a possibility that there is insufficient core storage to set an array in which to store the results of the calculations, it may be possible to use one of the given arrays for storing the result if each element in one of the given arrays is no longer needed after it has been involved in the calculations. For example, line 40 in the above program could have been written:

```
40 LET D(K,L) = D(K,L) * E(K,L) + 2*K + 3*L
```

The result of the calculations would then be stored back in the given array D. The numbess in the original array D would of course then be lost for any further calculations or printout.

Another method to use when insufficient core storage is available is not to store the calculated value but to print it out immediately.

6.5 PRINTING OUT TWO-DIMENSIONAL ARRAYS.

Two-dimensional arrays are usually printed out row by row. The TAB function may be used to cause the rows of numbers to be printed out in neat columns that are convenient to read. For example:

```
70 FOR I = 1 TO 3
80    PRINT TAB(0);F(I,1);TAB(10);F(I,2);TAB(20);F(I,3);
TAB(30);F(I,4)
90 NEXT I
```

If when you are typing a PRINT TAB statement and it cannot be typed all on one line, the typing may be continued onto a second line. Simply keep typing (or spacing) until the carriage return automatically causes return to column one. Then continue typing.

When the PRINT TAB function is used in printing out arrays in neat columns, the numbers in the columns will be left justified.in the column.

To form columns of numbers in which the numbers are printed right justified in the columns, PRINT USING statements are used. For example,

```
70 FOR I = 1 TO 3
80     PRINT USING 90, F(I,1),F(I,2),F(I,3),F(I,4)
90     % #####    #####    #####    #####
100 NEXT I
```

6.6 STRING VARIABLES IN TWO-DIMENSION ARRAYS.

The elements in a two-dimensional array may be string variables. Each element may consist of up to 18 characters, each being any valid BASIC alphanumeric character, including a letter, digit, or special character. There is nothing prohibiting the entire string from being digits. However, no arithmetic operations can be performed on string variables

The following program shows how to read in and print out a two-dimensional array consisting of string variables. The data is not shown.

```
10 DIM Z$(5,3)
20 FOR R = 1 to 5: FOR C = 1 TO 3: READ Z$(R,C):NEXT C: NEXT R
30 FOR R = 1 TO 5: PRINT TAB(0);Z$(R,1):TAB(22);Z$(R,2);
TAB(44);Z$(R,3)
40 NEXT R
```

EXERCISES 6.5

Describe the differences in the two printoûts from the following program:

```
10 DIM A(3,4)
20 FOR X = 1 TO 3:  FOR Y = 1 TO 4: READ A(X,Y): NEXT Y:NEXT X
30 FOR X = 1 TO 3
35 PRINT TAB(0);A(X,1);TAB(10);A(X,2);TAB(20);A(X,3);TAB
(40);A(X,4)
38 NEXT X
50 FOR X = 1 TO 3
55 PRINT USING 58,A(X,1),A(X,2),A(X,3),A(X,4)
58 %#####     #####     #####     #####
59 NEXT X
100 DATA 123, 456, 789, 684, 733, 451
101 DATA 831, 769, 155, 676, 852, 557
999 END
```

6.7 ESTIMATING PROGRAM LIMITS.

An estimate of the partition size required to run a particular BASIC program is given by the following formula:

Partition size needed = C + 5(N) + 18(A) + E

where C = the number of characters in the program, including spaces, carriage returns, etc.

N = the number of numeric variables. Each element of an array is one variable.

A = the number of alphanumeric variables.

E = the space required for FOR loop processing and expression evaluation. This is about 200 bytes for an average program but varies with program complexity.

6.8 RUNNING PROGRAMS THAT EXCEED PARTITION CAPACITY.

If the program to be run exceeds the capacity of the partition of the computer available, the program may be broken down into two or more smaller programs.

Variables that are common to both smaller programs are listed in a COM statement. The common array variables are also dimensioned in the COM statement. When the second program is to be run, the command RESTART is used instead of the START command.

The example below shows a program for performing matrix multiplication, that has been broken down into three programs. The three smaller programs are connected by the COM statements and RESTART commands.

```
:START
:10 COM A(10,10), B(10,10)
:20 FOR I = 1 TO 10: FOR J = 1 TO 10:READ A(I,J),B(I,J)
:30 NEXT J:  NEXT I
:40 DATA (100 numbers for array A and 100 for array B)
:99 END
:RUN
END PROGRAM
:RESTART
:10 COM A(10,10), B(10,10)
:20 COM C(10,10)
:30 FOR I = 1 TO 10:  FOR J = 1 TO 10: C(I,J) = 0
:40     FOR K = 1 TO 10: C(I,J) = C(I,J) + A(I,K)*B(K,J)
:50     NEXT K
: 60 NEXT J:  NEXT I
: 70 END
:RUN
END PROGRAM
:RESTART
:10 COM C(10,10)
:20 FO I = 1 TO 10
:30   PRINT USING 35,C(I,1),C(I,2),C(I,3),C(I,4),C(I,5),
C(I,6),C(I,7),C(I,8),C(I,9),C(I,10)
:35 % ####  ####  ####  ####  ####  ####  ####  ####  ####  ####
:40 NEXT I: END
```

6.9 THE CHAIN AND CHAINR COMMANDS.

```
100 CHAINR
```

The CHAINR program statement prodûces in effect an automatic combination of the following:

STOP (stop current program)

RESTART (clear user area,saving Common Data)

LOAD (load new program from terminal tape reader)

RUN (run the program)

This command can be used for loading and executing one or more segmented jobs with a minimum of intervention.

```
200 CHAIN
```

The CHAIN program statement produces in effect an automatic combination of the following:

STOP

START (clear user area)

LOAD

RUN

The CHAINR statement clears the user area but saves common data.

The CHAIN statement clears the user completely. This is useful in loading and running a series of independent programs (those having no common data) that have been saved on paper tape.

The use of the CHAIN and CHAINR statements in loading and executing programs saved on casettes and disks is presented in a later chapter.

6.10 WRITING A LEVEL SIX PROGRAM.

A level six program is one that involves reading in, calculating with, and printing out, one or more two-dimensional arrays. A typical program would be as follows:

```
10 REM   MICHAEL DELORENZO   LEVEL SIX   12/13/72
20 REM TWO-DIMENSIONAL ARRAYS
30 DIM A(10,5), B(10,5), C(10), D(10,5)
40 FOR X=1 TO 10:  FOR Y = 1 TO 5: READ A(X,Y),B(X,Y)
50 NEXT Y:  NEXT X
60 REM READ IN ARRAY C
70 FOR X = 1 TO 10:  READ C(X):  NEXT X
80 REM  PERFORM CALCULATIONS AND STORE IN ARRAY C
90 FOR X = 1 TO 10:  FOR Y = 1 TO 5
100   C(X,Y) = A(X,Y)* B(X,Y)*C(X)
110 NEXT Y:  NEXT X
120 REM   PRINT OUT ARRAY C
130 FOR X = 1 TO 10
140   PRINT USING 145,C(X,1),C(X,2),C(X,3),C(X,4),C(X,5)
145   % #####    #####    #####    #####    #####
150 NEXT X
160 DATA (110 numbers)
170 END
```

CHAPTER 7

USER-DEFINED FUNCTIONS AND SUBROUTINES

WRITING A LEVEL SEVEN PROGRAM

7.1 USER-DEFINED FUNCTIONS.

When a particular formula that can be entered on one line is required at several points in a program, the user may define this function and then use it anywhere and as often as needed in the program.

The function must be defined by use of a DEF statement which consists of a line number, the letters DEF, the letters FN followed by a letter or a digit which names the function, followed by parentheses that contain a scalar variable (a letter or a letter followed by a digit), followed by an equals sign, and then an expression which defines the function. For example:

```
60  DEF  FNA(H) = H↑2 + 2*H - 17.9
```

After the function has been defined, it is used in much the same way as any of the library functions. The statement

```
100 PRINT V, X, FNA(8), FNA(V*X)
```

will cause the system to print out the values of V, X, of $8^2 + (2)(8) - 17.9$ and of $(V*X)^2 + (2)(V*X) - 17.9$.

The DEF statement may refer to other functions either library functions or user-defined functions. For example:

```
120 DEF FNB(C) = C↑3 + FNA(C) - COS(C)
```

Up to five levels of function nesting is permitted. For example:

```
210 DEF FNL(X) = X↑3 + X↑2/9.77
220 DEF FNM(Y) = 6.17*Y - FNL(1.18)
230 A = FNM(13.91) + SQR(ABS(FNM(1.19)))
```

A function cannot refer to itself.

Two functions cannot refer to each other, as this would form an endless loop.

A reference cannot be made to an outside DEF FN statement from an immediate mode statement.

The DEF FN statement may be placed anywhere in the program.

The scalar variable in a DEF FN statement is called a dummy variable. It may have a variable name identical to a real variable used elsewhere in the program or in other DEF FN statements. Current values of these variables will not be affected during FN evaluation.

The user-defined function in BASIC can contain only one argument. The statement
80 DEF FNJ(A,B) = A↑2 + B↑2 is invalid. The parentheses may contain only a single scalar variable.

There is no purpose in defining a function if it is only going to be used once in a program.

7.2 SUBROUTINES.

A user-defined function can be no more than one program line in length. If a group of several statements are needed several times in a program, these statements do not have to be written over and over again. They may be written once and the system can then branch to them whenever they are needed. One way of doing this is by use of a subroutine.

A subroutine is really an independent program to be executed when called by another program

The first program line in the subroutine may be any BASIC statement. For example, below is a subroutine for calculating the factorial of a number, X. It considers the possibility of X being negative, X being equal to zero, and of X being positive. If X is positive, the factorial of the number is calculated and assigned to the variable F, and control is then returned to the main program.

If X is zero, the value of X factorial is one. This number is assigned to the variable F, and control is returned to the main program.

If X is negative, the message "FACTORIAL OF A NEGATIVE NUMBER IS UNDEFINED" is printed out, and control is returned to the main program.

```
200 IF X<0 THEN 300:  IF X = 0 THEN 280
210 N = X:  A = 1:  F = N * (N - A)
220 A = A + 1:  IF A<N THEN 210:  RETURN
280 F = 1:  RETURN
300 PRINT "FACTORIAL OF NEGATIVE NUMBER IS UNDEFINED": RTURN
```

REM statements may be written anywhere in a subroutine. It is ually a good idea to have the first statement be a REM statement, explaining the purpose of the subroutine. The last executable statement in a subroutine must be a RETURN statement. However, a non-executable statement such as a REM statement may be shown on the same line. For example:

```
310 RETURN:  REM  END OF SUBROUTINE
```

To call for a subroutine, a GOSUB statement is used in the main program. In the following program, the number of permutations of N things taken R at a time is calculated using the formula

number of permutations =(N factorial)/((N-R)factorial)

```
10 REM  TO FIND PERMUTATIONS OF N THINGS TAKEN R AT A TIME
20 READ N, R:  IF N = 0 THEN 100
30 LET X = N:   GOSUB 200
40 REM  F IS THE VALUE OF X! FOUND IN SUBROUTINE
50 F1 = F
60 LET X = N - R:   GOSUB 200
70 F2 = F
80 REM   PERMUTATIONS EQUAL N!/(N-R)!
90 LET P = F1/F2 :   PRINT N, R, P:   GO TO 20
100 END
200     REM   SUBROUTINE FOR CALCULATING X FACTORIAL
205     IF X <0 THEN 300:   IF X = 0 THEN 280
210     N = X:  A = 1:  F = N * (N - A)
220     A = A + 1:   IF A<N THEN 210:    RETURN
280     F = 1:   RETURN
300     PRINT "FACTORIAL OF NEGATIVE NUMBERS IS UNDEFINED"
310     RETURN:   END OF SUBROUTINE
95 DATA 15, 13, 8, 7, 10, 10, 25, 2, 0, 0
```

When the system encounters a RETURN statement in a subroutine, the system is directed to the verb following the last executed GOSUB statement. The first time the subroutine in the above program is executed, it will return to statement 40. The second time the subroutine is executed, it will return to the statement on line 80.

This points out the main advantage of the use of the GOSUB and RETURN statements over using just GO TO statements. A RETURN statement cannot be replaced with a GO TO statement because each time the subroutine is executed, the system is to return to a <u>different</u> part of the program than the last time it was executed.

Subroutines may be prepared in advance of writing a program, and they may be easily stored on punched paper tape to be loaded into the overall program when the main program is typed. Preparing and saving subroutines such as those for solving quadratic equations, solving a system of linear euqtaions, calculating incomes taxes, or calculating payroll dedeuctions can be very useful.

The GOSUB and RETURN statements may be used to perform a subroutine <u>within</u> a subroutine. This would be a <u>nested</u> GOSUB.

A program may contain several subroutines and they may be located anywhere in the program. However, great care must be taken to ensure that the system does <u>not</u> begin executing a subroutine without having been directed there by a GOSUB statement. If this should happen, the system would be completely confused in attempting to execute a RETURN statement in the subroutine. It would not know where to return to since it was not sent there by a GOSUB. It is wise to place all subroutines at the end of a program, <u>preceded</u> by an END statement.

7.3 THE GOTO ON STATEMENT.

```
50 GOTO 200, 250, 280, 300 ON (X + Y)
```

If the truncated integer value of X + Y is 1, control is transferred to the statement on line 200. If the value is 2, control is transferred to line 250; if 3 to line 280; if 4, to line 300. If the value is greater than 4, control is transferred to the next statement in the program.

If the truncated integer value of the expression within the parentheses is less than one, an error message is printed out and execution termiaated.

Statement 50 above may be considered as a replacement for the four following IF THEN statements:

```
50  IF INT(X+Y) = 1 THEN 200
51  IF INT(X+Y) = 2 THEN 250
52  IF INT(X+Y) = 3 THEN 280
53  IF INT(X+Y) = 4 THEN 300
```

On the next page is an example of a program utilizing GOTO ON statements.

Exercise 25 on page 140 involves the calculating of Federal income tax on taxable incomes between $6000 and $16,000. Below is an example of how such a program could be written utilizing the GOTO ON statements.

```
10 REM   FEDERAL INCOME TAX ON GROSS INCOMES $6000 - $16,000
20 PRINT "GROSS INCOME", "DEPENDENTS", "TAXABLE INCOME", "TAX"
    30 REM   G IS GROSS INCOME
    31 REM   D IS TOTAL NUMBER OF DEPENDENTS AND EXEMPTIONS
    40 READ G, D:   IF G = 0 THEN 400
    45 REM   I  IS TAXABLE INCOME
    50 I = G - 625*D - .10*G
    60 GOTO 110, 110, 200, 201, 202, 203, 204   ON (I/2000)
    110 PRINT "TAX CANNOT BE COMPUTED BY THIS PROGRAM": GO TO 40
    200 T = 1130 + .25*(I-6000):  GO TO 300
    201 T = 1630 + .28*(I-8000):  GO TO 300
    202 T = 2190 + .32*(I - 10000):   GO TO 300
    203 T = 2830 + .36*(I-12000):   GO TO 300
    204 T = 3550 + .39*(I-14000)
    300 PRINT G, D, I, T:   GO TO 40
    350 DATA 16000,2,   15570,1,   15250,4,   14780,2
    351 DATA 13500,2  13,125,7,   12800,2,   14819,4
    352 DATA 8000,6,   7250,1,   10777,5   0,0
    400 END
```

7.4 THE TRACE ON AND TRACE OFF STATEMENTS.

The TRACE statements provide for the tracing of the execution of a BASIC program. This is often useful in debugging a program that does not seem to be accomplishing its intended purpose.

TRACE mode is turned on TRACE or TRACE ON is executed and is turned off when TRACE OFF is executed.

When the TRACE mode is on, the printouts wiill be produced when any program variable received a new value during the execution of a LET, READ, or FOR statement. Printouts will also be produced when a program transfer is made to another sequence of statements by a GO TO, GOSUB, or IF statement.

```
      Program                         Printout
30 TRACE ON                       A = 0
40 READ A, B                      B = 90
50 X = A + (SIN(B/57.2958))       X = 1
60 IF X = 4 THEN 180              TRANSFER TO 0200
70 GOSUB 200                      C = 9
80 C = F                          TRANSFER TO 0040
90 GO TO 40                       A = 4
95 TRACE OFF                      B = 360
100 DATA 0,90,4,360,2,130         X = 4
180 END                           TRANSFER TO 180
200 (Subroutine)                  3300 BASIC READY
```

7.5 LET STATEMENTS WITH MULTIPLE VARIABLES.

If two or more variables are to be assigned the same value, this may be accomplished by writing the variables separated by commas to the left of the equals sign in a LET statement. For example:

```
10 LET X, Y = 3.49
```

This statement will cause the value of 3.49 to be assigned to both the variable x and the variable Y.

```
20 LET X, Y, Z(12) = A + SIN(B)/C
```

This statement will cause the system to calculate the value of A + SIN(B)/C and assign this value to each of the variables X, Y, and Z(12).

7.6 WRITING A LEVEL SEVEN PROGRAM.

A level seven program is one that makes use of a user-defined function and also use of either a subroutine or use of the GOTO ON statement. A portion or all of the program is to be executed in TRACE mode.

A typical level seven program is on the next page.

CHAPTER 8

MATRIX OPERATIONS

WRITING A LEVEL EIGHT PROGRAM

8 1 MATRICES.

A rectangular array of numbers subject to certain rules of operations may be called a **matrix**. The matrix

$$\begin{array}{ccc} 23.4 & 18.6 & 12.9 \\ -5.6 & -7.8 & 10.1 \end{array}$$

could be considered as the coefficient matrix of the system of homogeneous linear equations

$$23.4x + 18.6y + 12.9z = 0$$

$$-5.6x - 7.8y + 10.1z = 0$$

The matrix may be used in obtaining solutions to such systems of equations.

The numbers in a matrix are arranged in rows and columns. A matrix having three rows and five columns may be referred to as a "3X5" matrix which is read "3 by 5." A square matrix would have the same number of rows and columns.

A matrix may consist of a single row, and would be called a **row matrix** or a **row vector**. A matrix consisting of a single column would be called a **column matrix** or a column vector.

A matrix may consist of just a single element

Since a matrix is simply an array, a matrix name may be any one of the 26 letters of the alphabet.

Before any of the BASIC operations can be used with matrices, each matrix must be declared in a DIM statement to reserve computer space for it.

8.2 READING IN A MATRIX.

```
10 DIM A(3,4), B(10,10), C$(8)
20 MAT READ A, B(3,5), C$
180 DATA (12 numbers for matrix A, 15 numbers for matrix B
and 8 sets of alphanumeric characters for matrix C.)
```

The MAT READ statement causes the matrices to be filled in order with the values in the DATA statements. Each matrix is filled row by row.

In the example above, the first 12 numbers in the DATA statements will be assigned to matrix A. Notice that matrix B has been redimensioned to a 3 X 5 matrix by the MAT READ statement. Any matrix may be redimensioned in a MAT READ statement.

The next 15 numbers will be assigned to matrix B, and the eight sets of alphanumeric characters will be assigned to matrix C$. The elements of a matrix may be alphanumeric strings as long as the matrix name consists of a letter followed by a dollar sign. Remember, no arithmetic operations can be performed on alphanumeric strings.

8.3 MATRIX ADDITION AND SUBTRACTION.

```
10 DIM A(3,8), B(3,8), C(3,8), D(3,8)
20 MAT READ A, B
30 MAT C = A + B
40 MAT D = C - B
```

Matrix addition or subtraction may be performed on two matrices having the same dimensions. In matrix addition, each element in matrix A is added to the corresponding element in matrix B and the result is placed in the corresponding position in matrix C. Matrix C may appear on both sides of the equals sign.

In the matrix subtraction shown above, each element in matrix B is subtracted from the corresponding element in matrix C, and the result stored in the corresponding position of matrix D. Matrix D may appear on both sides of the equals sign.

If the dimensions of the two matrices being added or subtracted are not the same, an error message will be printed out and the execution terminated. The resulting matrix will have the same dimensions as those of the two matrices being added or subtracted.

8.4 PRINTING OUT A MATRIX.

```
100 MAT PRINT A; B; C, D
```

Each matrix is printed row by row.

All the elements of a row are printed on as many lines as are required with single line spacing. A blank line separates rows. The first element of a row always starts at the beginning of a new print line.

Four values will be printed to a line (zone-form) unless the matrix name is followed by a semicolon. The semicolon will cause the printing to be in pack-form.

A vector array is printed as a column vector, double spaced.

In the example above, matrix A and matrix B will be printed in pack-form. Matrix C and matrix D will be printed in zone-form.

8.5 MAT PRINT USING.

```
100 MAT PRINT USING 110, A;
110 % +##.#!!!!
```

When a semicolon or a blank is used after the name of the matrix to be printed out, the matrix will be printed out row by row with all the elements of a row printed on as many lines as required with single line spacing. An additional blank line will separate rows. At the beginning of each row, the image statement is restarted. Rows of matrices with more than four elements can be printed on a single line in any specfied format.

If matrix A in the example just shown is a 2 X 5 matrix, the two instructions on lines 100 and 110 will result in a printoit such as:

```
+11.3E-06   -15.4E-15   +7.6E+04   +6.7E+10   -12.2E-06
```

If a comma were used after the name of the matrix in line 100, a carriage return will only be output when the image statement is used up and must be restarted. The statements

```
150 MAT PRINT USING 160, A,
110 % +##.#!!!!   +##.#!!!!   +##.#!!!!   +##.#!!!!   +##.#!!!!
```

would result in the same output as the statements on line 100 and 110 above, if matrix A were a 2 X 5 matrix.

The following is an example of an image statement used for printing out literal strings as well as the numeric value of a matrix:

```
300 MAT PRINT USING 310, C
310 % NO. ITEMS = ####    COST PER ITEM = ###.##    TOTAL = #####.
```

If matrix C were a 3 X 3 matrix, the printout might appear as:

```
NO. ITEMS =   12   COST PER ITEM = 12.23   TOTAL =   146.76
NO. ITEMS =   10   COST PER ITEM = 10.00   TOTAL =   100.00
NO. ITEMS =  100   COST PER ITEM =  5.50   TOTAL =   550.00
```

8.6 MATRIX SCALAR MULTIPLICATION.

```
10 DIM D(5,10), E(5,10), F(5,10), g(5,10)
20 MAT READ D
30 READ X,Y
40 MAT E = (7)*D
50 MAT F = (x ↑2 + Y ↑2)*D
60 MAT G = (COS(X))*D
```

The general form for matrix scalar multiplication is MAT C = (k)*A, where matrix C and matrix A have the same dimensions and k is a factor called a scalar. The scalar, k, may be a constant, a variable, or an expression. Each element of matrix A is multiplied by the scalar k and the result is stored in the corresponding position position in matrix C.

It is essential that the scalar k be enclosed in parentheses, even when it is a constant.

8.7 MATRIX MULTIPLICATION.

```
10 DIM a(3,4), B(4,3), C(3,3)
20 MAT READ A, B
30 MAT C = A*B
```

Two matrices may be multiplied if the first matrix has the same number of elements in each <u>row</u> as the second matrix has in each <u>column</u>. The number of columns of the first matrix must equal the number of rows of the second matrix.

In matrix multiplication, the product of the row elements of the first matrix and the corresponding column elements of the second matrix are added to produce the element in the product matrix whose row number and column number is the same as the row and column containing the elements being multiplied. That is,

```
C(1,1) = A(1,1)*B(1,1)+A(1,2)*B(2,1)+A(1,3)*B(3,1)+A(1,4)*B(4,1)
```

The product matrix will have the same number of rows as the first matrix and the same number of columns as the second matrix.

If the number of elements in the first matrix does not equal the number of elements in the second matrix, an error message will be printed out and execution terminated.

Matrix multiplication is not commutative.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{X} \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 16 & 19 \\ 36 & 43 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} \text{X} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 19 & 27 \\ 28 & 40 \end{bmatrix}$$

The same matrix cannot appear on both sides of the equals sign.

Exercises 8.7

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 5 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 1 & -2 \\ -4 & 5 & 2 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 2 \end{bmatrix}$$

$$D = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} \qquad E = \begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix}$$

Calculate:

1. A + B
2. A - B
3. A + C - B
4. DE
5. ED
6. AC
7. CA
8. 5D
9. 10E
10. 4A

$$F = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 5 & 6 \end{bmatrix} \qquad G = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Calculate:

11. FG
12. GF
13. 5F
14. 25G

15. Which of the following are impossible to perform, using the above matrices?

a) D + E
b) E - G
c) DF
d) GF
e) BC
f) EG
g) A + F
h) D + F
i) C + F

8.8 MATRIX EQUALITY.

```
10 DIM a(2,4), B(4,5)
20 MAT READ A
30 MAT B = A
```

Each element in matrix A is stored in the corresponding position in matrix B. Matrix B is automatically redimensioned to conform to the dimensions of matrix A.

Multiple equal statements such as MAT A,B = C are not allowed.

Two matrices are equal if and only if they have the same dimensions and each element of one is equal to the corresponding element of the other.

The names of the matrices must be numeric array names. String arrays may not be used.

8.9 MATRIX CONSTANT ONE FUNCTION.

```
10 DIM A(25), B(8,8), C(20,20)
20 MAT A = CON
30 MAT B = CON(4,7)
40 MAT C = CON(X,K)
```

The MAT CON function sets all elements of the named matrix to 1. If the matrix is to be redimensioned, the new dimensions are contained in parentheses following CON. If no parentheses follow CON, the matrix will not be redimensioned. The matrix name must be a numeric array name.

A matrix all of whose elements are 1 is sometimes called the J matrix.

8.10 IDENTITY MATRIX.

```
10 DIM A(6,6), B(9,9), C(10,10)
20 MAT A = IDN
30 MAT B = IDN(8,8)
40 MAT C = IDN(X,Y)
```

The identity or unit matrix is a square matrix all of whose elements are zeroes except those on the main diagonal (top left to bottom right) which are all ones. It is called the identity matrix because it is the multiplicative identity matrix.

As shown above, the matrix may be redimensioned at execution time.

If the specified matrix is not a square, an error message will be printed and execution terminated.

This can only be used for numeric arrays.

If matrix A is the identity matrix, the product of matrix A and a second matrix is a product matrix identical to the second matrix.

8.11 MATRIX ZERO FUNCTION.

```
10 DIM A(5,6), B(10,10), C(8,11), D(50)
20 MAT A = ZER
30 MAT B = ZER(7,7)
40 MAT C = ZER(X,Y+1)
50 MAT D = ZER(25)
```

The MAT ZER function sets all the elements of a matrix to zero. Writing two expressions separated by a comma within parentheses after ZER will redimension the matrix while zeroing it.

8.12 MATRIX TRANSPOSITION.

```
10 DIM a(2,5), C(5,2)
20 MAT C = TRN(A)
```

A matrix is transposed when the elements in the rows become the elements in the columns, and the elements in the columns become the elements in the rows of the resulting matrix.

For example, the matrix

$$\begin{bmatrix} 5 & 7 & 2 & 4 & 9 \\ 1 & 3 & 5 & 7 & 8 \end{bmatrix}$$

when transposed becomes

$$\begin{bmatrix} 5 & 1 \\ 7 & 3 \\ 2 & 5 \\ 4 & 7 \\ 9 & 8 \end{bmatrix}$$

Matrix C may not appear on both sides of the equals sign.

8.13 THE INVERSE OF A SQUARE MATRIX.

```
10 DIM A(7,7), C(9,9)
20 MAT READ A
30 MAT C = INV(A)
```

The identity matrix is any square matrix all of whose elements are zeroes except those on the main diagonal which consists of all ones. For every nonsingular square matrix there exists an inverse matrix such that the product of the matrix and its inverse is the identity matrix. For example, the inverse matrix of $\begin{matrix}3 & 5\\ 7 & 6\end{matrix}$ is $\begin{matrix}6 & -5\\ -7 & 3\end{matrix}$ because their product is the identity matrix $\begin{matrix}1 & 0\\ 0 & 1\end{matrix}$

Finding the inverse of a large matrix can be very difficult. However, with the use of BASIC and a computer, it is very simple.

Matrix C may appear on both sides of the equals sign. The dimensions of C are automatically determined by the dimensions of matrix A. Notice in the DIM statement above, matrix C was dimensioned larger than actually necessary.

Not only must matrix A be square, but it must be nonsingular. That is, its determinant must not be zero. An example of a singular matrix would be a zero matrix.

8.14 DETERMINANT OF A MATRIX.

```
10 DIM A(7,7), C(9,9)
20 MAT C = INV(A)
30 D = DET
40 PRINT "DETERMINANT OF MATIRX A IS ", D
```

After the inversion of matrix A, the DET function is equal to the value of the determinant of matrix A.

8.15 MATRICES IN IMMEDIATE EXECUTION MODE.

The following matrix statements may be used in immediate execution mode:

MAT addition	MAT A = B + C
MAT CON	MAT A = CON
MAT equality	MAT A = B
MAT IDN	MAT A = IDN
MAT INV	MAT A = INV(B)
MAT multiplication	MAT A = B*C
MAT scalar multiplication	MAT A = (6)*B
MAT subtraction	MAT A = B - C
MAT TRN	MAT A = TRN(B)
MAT ZER	MAT A = ZER
MAT PRINT	MAT PRINT A,B

8.16 MATRIX INPUT.

```
10 DIM A(4), B$(3)
20 MAT INPUT A, B$(2), C(3,4)
```

The MAT INPUT statement causes the system to print a question mark when the statement is executed. The system then waits for the user to supply values for the matrices indicated. In the ecample above, four values would be entered for matrix A, two sets of alphanumeric characters would be entered for matrix B$, and 12 values would be entered for matrix C.

The MAT INPUT statement may redimension a matrix already deminsioned by a DIM statement or by another MAT statement. The MAT INPUT statement may also dimension a matrix that has not yet been dimensioned by a DIM statement or other statement. In the example above, the MAT INPUT statement has redimensioned matrix B$ and has dimensioned matrix C for the first time.

The value inputs are assigned to a matrix row by row until the matrix is filled. Excess data is ignored. Entering no data on an input line and striking the carriage return signals the system to ignore the remaining elements of the matrix currently being filled.

If there is a system-detected error in the entered data, the data must be re-entered beginning with the erroneous number. The data which precedes the error is accepted.

8.17 MULTIPLE MATRIX OPERATIONS.

Multiple matrix operations are not permitted in a single program statement. A single statement may contain only one matrix operation.

10 MAT C = A + B - C   is illegal.

This would have to be accomplished by the use of two statements:

```
10 MAT C = A + B
20 MAT C = C - D
```

8.18 AUTOMATIC DIMENSIONING OF A MATRIX.

If an array variable that has not been dimensioned is encountered in a MAT statement, it will automatically be defined as a two-dimensional matrix with dimensions 10 X 10.

8.19 REDIMENSIONING OF A MATRIX.

A matrix may be redimensioned by writing the new dimensions within parentheses following the matrix name in any of the following MAT statements:

MAT CON	MAT A = CON(5,6)
MAT IDN	MAT B = IDN(9,9)
MAT ZER	MAT C = ZER(35)
MAT INPUT	MAT INPUT D(5,5)
MAT READ	MAT READ E(5,11)

A matrix may also be redimensioned in a MAT FILEREAD statement, which is presented in chapter 9.

When a matrix is redimensioned, the total number of elements in the redimensioned matrix cannot be greater than the total number of elements specified by the original dimensions as given in a DIM or COM statement, or 10 X 10 if not specified. If matrix A is dimensioned by

```
10 DIM A(10,10)
```

it may be redimensioned to (11,9), (12,8), (5,5), (2,50), (75), (100), etc., but cannot be redimensioned in such a way as to result in more than 100 elements. If this condition is not met, an error message will be printed out.

In arithmetic matrix operations, the matrix on the left-hand side of the equals sign is automatically redimensioned, receiving the dimensions of the resulting matrix.

```
10 DIM A(3,3), B(3,3), C(20)
20 MAT C = A * B
```

Matiz C will automatically be redimensioned to (3,3).

8.20 USE OF STRING VARIABLES IN MATRICES.

Arithmetic operations may not be performed on string variables. Therefore, only the following matrix statements may contain string variables:

MAT INPUT
MAT PRINT
MAT PRINT USING
MAT READ

8.21 SOLVING A SET OF SIMULTANEOUS LINEAR EQUATIONS.

$$X + 2Y + 3Z = 26$$

$$3X + 5Y + 2Z = 39$$

$$2X + 4Y + Z = 27$$

The above set of simultaneous linear equations may be represented by the matrix equation

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 5 & 2 \\ 2 & 4 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 26 \\ 39 \\ 27 \end{bmatrix}$$

The inverse of $\begin{bmatrix} 1 & 2 & 3 \\ 3 & 5 & 2 \\ 2 & 4 & 1 \end{bmatrix}$ is $\begin{bmatrix} -.6 & 2 & -2.2 \\ .2 & -1 & 1.4 \\ .4 & -1.39698 \times 10^{-9} & -.2 \end{bmatrix}$

If both sides of the above matrix equation were multiplied by the inverse of $\begin{bmatrix} 1 & 2 & 3 \\ 3 & 5 & 2 \\ 2 & 4 & 1 \end{bmatrix}$

the resulting matrix equation would be

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\end{bmatrix} = \begin{bmatrix}-.6 & 2 & -2.2\\ .2 & -1 & 1.4\\ .4 & -1.39698 \times 10^{-9} & -.2\end{bmatrix}\begin{bmatrix}26\\ 39\\ 27\end{bmatrix}$$

or

$$\begin{bmatrix}X\\ Y\\ Z\end{bmatrix} = \begin{bmatrix}3\\ 4\\ 5\end{bmatrix}$$

Therefore, the solution to the set of simultaneous linear equations is X = 3, Y = 4, Z = 5.

The BASIC program designed to perform these computations would be:

```
10 DIM A(3,3), B(3,1), D(3,3), E(3,1), F(1,3)
20 MAT READ A, B
30 MAT D = INV(A)
40 MAT E = D*B
45 MAT F = TRN(E)
50 PRINT "X", "Y", "Z"
60 MAT PRINT F
70 DATA 1, 2, 3, 3, 5, 2, 2, 4, 1
80 DATA 26, 39, 27
90 END
```

EXERCISES 8.21

Write the necessary programs and solve each of the following sets of simultaneous linear equations.

1. $$\begin{aligned} x - y &= 73 \\ 2x + 7y &= 29 \end{aligned}$$

2. $$\begin{aligned} 2a + 5b &= 18 \\ 3a + 4b &= 7 \end{aligned}$$

3. $$\begin{aligned} 2x - y + z &= 3 \\ x + 2y - z &= 3 \\ 3x - 4y + 2z &= -1 \end{aligned}$$

4. $$\begin{aligned} x + y - 3z &= 8 \\ 2x - 3y + 2z &= -6 \\ 3x + 4y - 2z &= 20 \end{aligned}$$

5. $$\begin{aligned} 12.418v - 1.061w + 2.669x + 4.361y - 0.119z &= 4.508 \\ -1.501v + 19.832w + 0.694x - 4.816y + 2.274z &= -8.449 \\ 2.308v + 1.728w - 15.165x - 2.023y + 1.104z &= -33.031 \\ 3.359v - 0.913w - 6.441x + 27.864y + 3.737z &= -106.909 \\ -1.562v + 1.168w - 2.004x + 1.818y + 9.490z &= 30.268 \end{aligned}$$

Answers: 1. $x = 60$, $y = -13$

3. $x = 1$, $y = 3$, $z = 4$

5. $v = 1$, $w = -2$, $x = 3$, $y = -4$, $z = 5$

CHAPTER 9

DATA FILE OPERATIONS

WRITING A LEVEL NINE PROGRAM

9.1 THE FILES STATEMENT.

Data files are sets of numeric and/or alphanumeric data which are stored on paper tape, casettes, or disk.

The FILES statement is used to assign a physical unit to each file designator referenced in the program. All files designators used in a program must contain a corresponding unit assignment in the FILES statement list.

The files are assigned sequentially, the first item in the list assigns file #1, the second assigns file #2, etc.

The file designator #0 when used in a program always implies the teletype terminal paper tape reader and punch. Therefore, a FILES statement is <u>not</u> necessary for paper tape files.

Casette tape files are identified by numbers 1 through 32. If there is more than one file on a casette tape, the casette number is followed by a slash and the number of the file location on that tape.

```
10 FILES 3, 5/4
```

According this statement, file #1 is to be casette tape 3 and file #2 is to be casette tape 5, fourth file on that tape.

Disk files already catalogued are identified by their name within quotation marks.

Disk files to be catalogued are identified by their name within quotation marks, followed by a slash and the number og disk storage units required for the file. Disk storage units are multiples of 1024 characters.

```
10 FILES "HENRY","MARY"/5
```

According to this statement, file #1 is to be a disk file already catalogued under the name "HENRY."

File #2 is to be a new disk file to be catalogued under the name "MARY" and is to be allocated 5 disk storage units, which is equal to 5 times 1024 or 5120 characters.

The FILES Statement must appear in the program before any file operations. It may, however, be modified after a program is loaded and before it is run. This permits a program to be written with file operations. The actual units to be used for the files can be assigned at execution time.

Up to eight disk or casette files can be assigned in one program (#1 through #8).

9.2 THE FILEREAD STATEMENT.

```
10 FILES 3, 5/2, "MARY", "MARK"/5
20 FILEREAD #0, A, B, C$
30 FILEREAD #2, D, E, F
40 FILEREAD #3, G$, H$, K
```

The FILEREAD statement on line 20 causes data to be read from the designated file and sequentially assigned to each variable in the list. Both numeric and alphanumeric data may be read in the same FILEREAD statement.

The FILEREAD statement on line 20 above will cause two numeric values and a literal string to be read from the paper tape file and assigned to the variables A, B, and C$ respectively.

The FILEREAD statement on line 30 will cause the first three values from file #2 to be read and assigned to the variables D, E, and F respectively. The FILES statement indicates that file #2 is the second file on casette tape number 5.

The FILEREAD statement on line 40 will cause the first three values from file #3 to be read and assigned to the variables G$, H$, and K respectively. The first two of these values will be literal strings and the third will be numeric. The FILES statement indicates that file #3 is the file on the disk which is catalogued as "MARY."

If a file is exhausted before all variables in the list are satisfied, the remaining variables are ignored.

9.3 THE MAT FILEREAD STATEMENT.

```
10 DIM A(12), B(5,5), C(10,6), D$(20)
20 FILES 6, "JOE"
30 MAT FILEREAD #0, A
40 MAT FILEREAD #2, B
50 MAT FILEREAD #1, C(3,3), D$(5)
```

The MAT FILEREAD statement is used when the values being read from the files are matrices, either numeric or alphamumeric.

Each file element in each array will be assigned a value from the file, row by row.

If optional dimensions appear in the list following the variable name, the array will automatically be redimensioned to the new dimensions.

The difference in meaning of C(3,4) contained in a FILEREAD statement and the meaning of C(3,4) contained in a MAT FILEREAD statement should be clearly understood.

```
100 FILEREAD #0, C(3,4)
110 MAT FILEREAD #0, C(3,4)
```

The FILEREAD statement on line number 100 will cause one number to be read from the paper tape file and assigned to the variable C(3,4).

The MAT FILEREAD statement on line 110 will cause matrix C to be redimensioned and will cause 12 mumbers to be read from the paper tape file and assigned to the 12 positions in matrix C.

9.4 THE FILEWRITE STATEMENT.

```
100 FILEWRITE #0, A, B(3,3)
110 FILEWRITE #4, C$, "ABC3F", D
120 FILEWRITE #3, B(1,3), "ANSWER", SQR(1.133)
```

The FILEWRITE statement causes the numeric or alphanumeric charcater strings to be sequentially written onto the file indicated.

The file element may be any legal numeric variable, alphanumeric variable, numeric array element, alphanumeric array element, alphanumeric literal, or numeric expression.

Alphanumeric variables will be written identically to the character string data they contain, except that quotation marks will be inserted immediately before and after the string.

An output file can be written and reread in the same program, but it must be first terminated by a FILEEND statement after it is written and before it is read. See section 9.7.

After all the data on a file is read, subsequent FILEREAD operations for that file will be ignored. The file will have an end of file status, which can be tested by an IF END statement. See section 9.6.

9.5 THE MAT FILEWRITE STATEMENT.

```
200 MAT FILEWRITE #0, A, B$
210 MAT FILEWRITE #3, C
```

The MAT FILEWRITE statement causes the values of the specified numeric or alphanumeric arrays to be written on the file indicated without referencing each member individually. Only the array name need be indicated.

Each array is written row by row with values separated by commas.

Alphanumeric values are written as character strings enclosed by quotation marks.

9.6 THE IF END STATEMENT.

```
100 FILEREAD #2, A, B, C
110 IF END #2 THEN 150
    ...
200 MAT FILEREAD #0, E
210 IF END #0  THEN 270
```

The IF END statement is used to test for end of file condition following FILEREAD and MAT FILEREAD statements. If end of file condition exists, the system will transfer to the specified statement number.

A file has an end of file condition when all the data on the file has been read.

9.7 THE FILEEND STATEMENT.

```
200 FILEEND #0
300 FILEEND #3
```

The FILEEND statement is used to close or terminate an output file that has just been written and must be reread in the same program. It is also used to bypass the remainder of an input file.

The FILEEND statement causes the following action to occur:

Output paper tape - an end of file is punched followed by 50 blank frames.

Output casette tape - An end of file sequence is written and the tape rewound and positioned at the start of that file.

Output disk - An end of file is written and the file is redefined as an input file. The file pointer is reset to the first file element.

Input paper tape, casette, or disk - The remainder of the file is read and ignored until an end of file is encountered.

9.8 THE FILEMOD STATEMENT.

```
10 FILES 2, "JOE", "FILE-K"
160 FILEMOD #2, "FILE-A"
170 FILEMOD #3, SCRATCH
```

The FILEMOD statement is used to rename a file catalogued on disk. The new name is enclosed in by quotation marks.

If the word SCRATCH appears at the end of the statement, the FILEMOD statement will cause the file to be permanently removed from the disk.

The statement on line number 160 will cause the file catalogued as "JOE" to be renamed "FILE-A". The statement on line number 170 will cause the file catalogued as "FILE-K" to be permanently removed from the disk.

9.9 FILESAVE STATEMENT.

```
200 FILESAVE #2
```

FILESAVE is used when an output file just written is to be permanently saved.

Paper tape - An end of file and 50 blank frames are punched.

Casette tape - An end of file is written and the tape remains positioned where it is. Another file may subsequently be written.

Disk - The file is catalogued and stored on disk.

9.10 CHAIN AND CHAINR STATEMENTS.

```
200 CHAIN
200 CHAIN R
200 CHAIN 3
200 CHAIN "KEEPIT"
```

The CHAIN and CHAINR statements permit segmented jobs to be run automatically without normal intervention. The CHAIN statement produces an automatic combination of

```
STOP
START
LOAD
RUN
```

The CHAINR statement produces the same effect except RESTART is substituted for START.

If nothing follows CHAIN or CHAINR, the loading will be done from the teletype paper tape reader.

If a number from 1 through 16 follows the CHAIN or CHAINR, loading will be from the casette whose number is indicated.

If a name in quotation marks follows CHAIN or CHAINR, the loading will be from the disk file.

9.11 IMMEDIATE EXECUTION MODE.

MAT FILEREAD and MAT FILEWRITE may not be used in immediate execution mode.

File operations are not legal in immediate execution mode.

9.12 GENERATING PAPER TAPES OFF-LINE.

Since files are in alphanumeric format, paper tape files can also be generated off-line on a teletype in the following manner:

Key in each variable followed by a

CR/LF/RUBOUT/RUBOUT

An end of file is designated by an

X-OFF/CR/LF