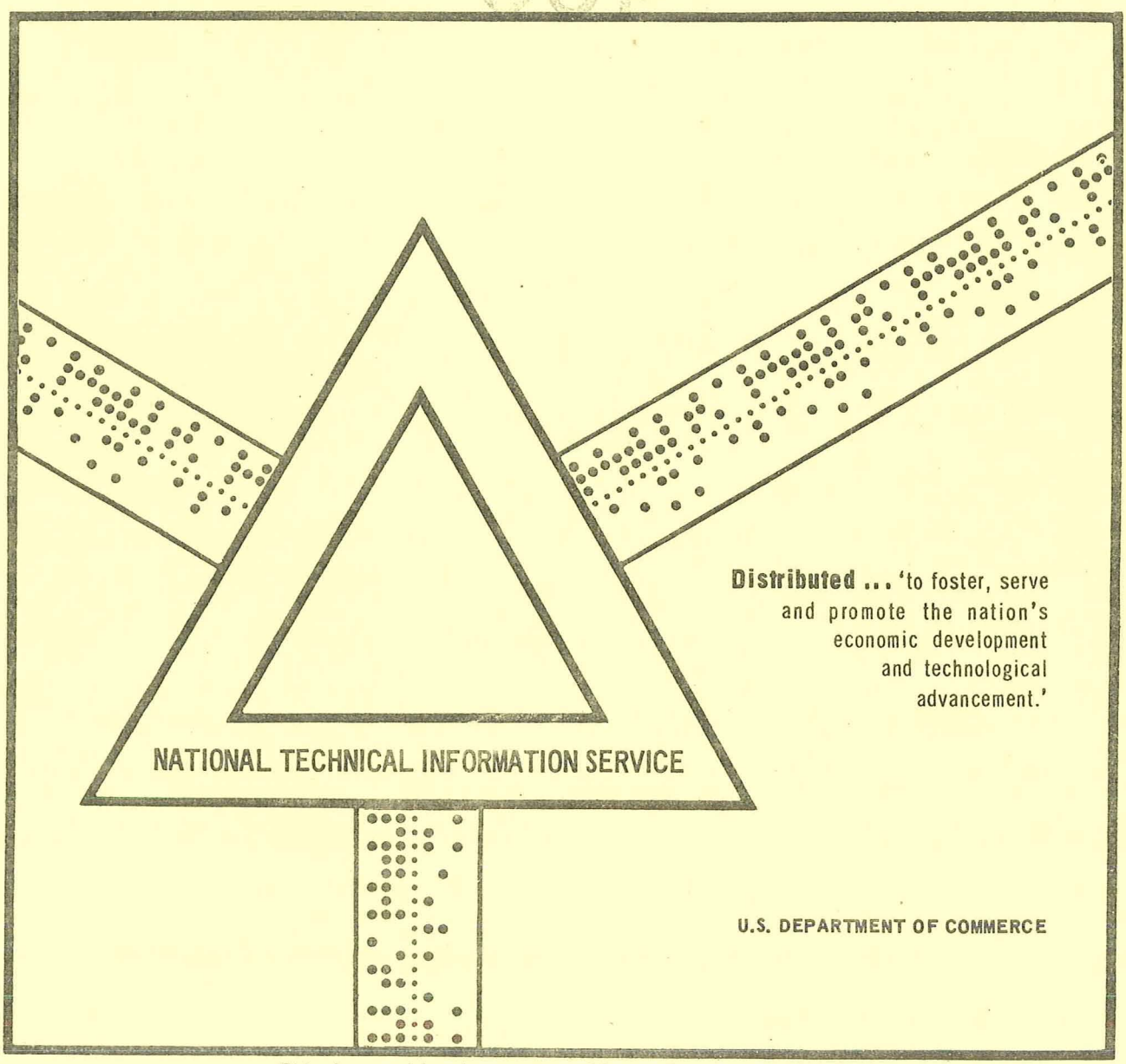
A SELF-STUDY COURSE IN FORTRAN PROGRAMING - VOLUME I - TEXTBOOK

Valmer Norrod, et al

Computer Sciences Corporation
El Segundo, California

April 1970



NASA CONTRACTOR REPORT

NASA CR-1478, Vol. I

NASA CR-1478, Vol. I

A SELF-STUDY COURSE IN FORTRAN PROGRAMING

Volume I - Textbook

by Valmer Norrod, Sheldon Blecher, and Martha Horton

Prepared by
COMPUTER SCIENCES CORPORATION
El Segundo, Calif.
for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • APRIL 1970

NASA CR-1478, Vol. I

A SELF-STUDY COURSE IN FORTRAN PROGRAMING

Volume I - Textbook

By Valmer Norrod, Sheldon Blecher, and Martha Horton

Prepared under Contract No. NAS 5-9758 by
COMPUTER SCIENCES CORPORATION
El Segundo, Calif.

for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 - CFSTI price $3.00

Page intentionally left blank

ABSTRACT

This two-volume manual is a comprehensive course in FORTRAN programing. Beginning with number systems and basic concepts, it proceeds systematically through the elements of the FORTRAN language and concludes with a discussion of programing techniques such as flow charting and debugging. Volume I is organized as a programed textbook with frequent checkpoints and abundant examples. Exercises and answers referred to in Volume I are contained in the Workbook - Volume II. Written for training programers at the NASA Langley Research Center, the manual is based on Control Data FORTRAN 2.3, but it is generally applicable to other versions.

Page intentionally left blank

FOREWORD

This manual was developed by Computer Sciences Corporation for training FORTRAN programers at the NASA Langley Research Center. While it is a description of Control Data FORTRAN 2.3 structured for self-study, it is generally applicable to other versions of FORTRAN.

The following personnel of the Langley Research Center worked closely with the contractor and made major contributions to the document: Lessie D. Hunter, Margaret A. Ridenhour, Nancy L. Taylor, and Dorothy J. Vaughan. Prior to publication, the manual has been used by more than 100 people and has proved to be very effective.

It is suitable for use by an individual studying alone, or as the basis for informal group study. When used as a group text, experience has shown that the study is more effective if it is supplemented by periodic reviews by a course monitor and concluded with a final examination.

Parts I - IV are intended to cover the basic elements of the FORTRAN language and may be used as a course for the engineer or manager interested in acquiring only a basic knowledge of the language but not intending to progress into actual programing work.

Roger V. Butler
Head, Computational Techniques Section
Langley Research Center

Page intentionally left blank

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

A SELF-STUDY COURSE IN FORTRAN PROGRAMING

Volume I - Textbook

By Valmer Norrod, Sheldon Blecher, and Martha Horton
Computer Sciences Corporation

I.A Introduction

I.A.1

To serve as both an introduction to this manual and as a note of encouragement to the student, it can be stated with assurance that anyone who has conquered the 3 R's can learn FORTRAN. This, of course, assumes a good self-teach manual (our obligation) and reasonable effort on the student's part (your obligation).

One of the primary objectives of this introductory section is to familiarize the student with the philosophy behind this manual and to present basic computer concepts which are applied in computer programming. Much of the information presented here is not necessary in order to learn how to use FORTRAN but it will at least serve to place things in proper perspective.

One being exposed to computer programming for the first time might very well ask the following questions.

What exactly is a computer?
How does a computer work?
What is a computer program?
How does a computer use a computer program?
How does information get in and out of a computer?
What specific steps must be taken in order to get the computer to do something?

PART I
GENERAL

I.A.1
(Cont.)

The material covered in the introductory part of the course attempts to answer these questions and in so doing covers basic background information. This portion of the manual will acquaint the new programmer with the tools with which he will be operating. Included will be discussions and exercises on what a computer is, how it operates, and what it requires in order to operate. Exercises and answers are contained in Volume II and will be referred to at the appropriate points in the text. Also, in order to familiarize the student with words and expressions in common use in the computer field, a glossary of computer terms has been compiled and incorporated as part of this introductory section.

The philosophy behind this self-teach manual is to break FORTRAN down to its truly basic elements. At times it may seem that we are "begging the point". Be patient with us - at worst we are guilty of over-explanation. We have chosen over-explanation as a method by which we can avoid the more serious pitfall of insufficient explanation.

In order to insure that a point is clearly made, a question or a series of questions follows the presentation of each new fact. The questions have been arranged on the page with the answers in the margin so that you can cover them while reading the questions. If you answer a question incorrectly reread the previous statement until you understand the answer to the question.

Great pains have been taken in order to develop an orderly accumulation of facts in small steps. To fully realize the advantages of this manual - take the small steps - don't jump around.

I.B Number Systems

I.B.1 Basic to the understanding of a computer system is the understanding of number systems. All of us are familiar with the decimal number system although many of us, due to the way we were taught, have no idea of what another number system means.

I.B.2 What makes the decimal number system a decimal system is simply the fact that 10 distinct digit characters are defined within the system (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). The decimal number system can also be referred to as a number system to the base 10 since the number of digit characters defined in a number system establishes the base of the number system.

The decimal number system is a number system to the base _____. Answer: 10

I.B.3 As an aside, it should be realized that there is nothing truly logical about using the decimal number system. It simply evolved because we have 10 fingers (10 distinct digit characters) which proved convenient as a counting device (as many of us realize).

I.B.4 A number system which contains 8 distinct digit characters (0, 1, 2, 3, 4, 5, 6, 7) is referred to as an octal number system or a number system to the base 8. Similarly a number system containing 2 digit characters (0, 1) is referred to as a binary number system or a number system to the base 2.

A binary number system contains _____ digit characters. Answer: 2
An octal number system contains _____ digit characters. Answer: 8

I.B.5

Clearly, using numbers as a counting device, one must be able to count more than the number of digits contained in the number system.

As an illustration let's assume we have 37 marbles to count and that we are fortunate enough to have two 10-fingered people available to perform the count. How would we go about counting the number of marbles on these fingers? We would have person 1 count 10 marbles on his fingers. Every time person 1 has completed a count of 10, person 2 will raise a finger. At the completion of the count person 2 will have 3 fingers raised indicating that person 1 has counted 10 three times. Person 1 at the completion of the count will have 7 fingers raised. Looking at the raised fingers of these two people we would know that the total count was 37. Three 10's indicated by person 2 plus the 7 indicated by person 1.

It should be noted that in this illustrative example person 1 will never raise 10 fingers. Instead of person 1 raising his tenth finger, he lowers all his fingers and person 2 raises one of his fingers. The maximum count possible by these 2 people is 99, 9 fingers raised by both persons 1 and 2.

To count any further requires the assistance of a third person. A finger raised by this third person indicates the completion of a count of 10 tens = $10 \times 10 = 10^2 = 100$.

Let's review for a moment and observe the pattern. The digits of person 1 represent single counts (10^0). The digits of person 2 represent counts of 10 (10^1). The digits of person 3 represent counts of 100 (10^2). The digits of person 4 will represent a count of 1000 (10^3), person 5 a count of 10,000 (10^4), and so on.

Note that a number raised to a power of zero equals 1 ($10^0 = 1$).

I. B. 5
(Cont.)

The point of this whole illustration is to show the significance of digit position in a number. Although each person in the illustration has only 10 digits, the count is only limited by the number of people or analogously by the number of positions in a number - not by the number of digits available to the position.

Let's carry the analogy to a decimal number 285383. We assign positions as follows.

position →

6	5	4	3	2	1
2	8	5	3	8	3

position 1 contains 3 which represents 3 ones = 3×10^0.

position 2 contains 8 which represents 8 tens = 8×10^1.

position 3 contains 3 which represents 3 hundreds = 3×10^2.

position 4 contains 5 which represents 5 thousands = 5×10^3.

position 5 contains 8 which represents 8 ten thousands = 8×10^4.

position 6 contains 2 which represents 2 hundred thousands = 2×10^5.

The total number is the sum of all these:

$$3 \times 10^0 + 8 \times 10^1 + 3 \times 10^2 + 5 \times 10^3 + 8 \times 10^4 + 2 \times 10^5$$

$$= 3 \times 1 + 8 \times 10 + 3 \times 100 + 5 \times 1000 + 8 \times 10000 + 2 \times 100000$$

$$= 3 + 80 + 300 + 5000 + 80000 + 200000.$$

Fill the following box with the decimal number 95432.

position →

5	4	3	2	1

Answer:

9	5	4	3	2

I.B.5 (Cont.)

position 1 contains a __ which represents ___ ones = ___ x 10^0. Answer: 2, 2, 2

position 2 contains a __ which represents ___ tens = ___ x 10^1. Answer: 3, 3, 3

position 3 contains a __ which represents ___ hundreds = ___ x 10^2. Answer: 4, 4, 4

position 4 contains a __ which represents ___ thousands = ___ x 10^3. Answer: 5, 5, 5

position 5 contains a __ which represents ___ ten thousands = ___ x 10^4. Answer: 9, 9, 9

The total number is a sum of products.

___ x 10^0 + ___ x 10^1 + ___ x 10^2 + ___ x 10^3 + ___ x 10^4 Answer: 2, 3, 4, 5, 9

which equals ___ x 1 + ___ x 10 + ___ x 100 + ___ + 1000 + ___ x 10000 Answer: 2, 3, 4, 5, 9

which equals ___ + ___ + ___ + ___ + ___. Answer: 2, 30, 400, 5000, 90000

I.B.6

Let's carry our analogy a step further and assume that we wish again to count 37 marbles but only have people available with 8 fingers. This time person 1 can only count to 8 and person 2 must keep track of the number of times person 1 completes a count of 8. To count to 37 requires person 1 to complete a count of 8 four times which means that person 2 has 4 digits raised. Person 1 at the completion of the count will have 5 digits raised. So - person 2 has 4 digits raised and person 1 has 5 digits raised representing the number 45 in a number system to the base 8 (octal). A subscript after a number is commonly used to denote the base of the number. For example, 45_8 means 45 in a number system to the base 8.

A 45 in the octal number system therefore represents four 8's and five units or 1's and is equal to 37 in the decimal number system.

I. B. 6
(Cont.)

Getting back to our eight-fingered people, the addition of a third person enables us to keep track of how many times 8 eights are counted. Each digit raised by this third person therefore represents 8 eights, 3×8, 8^2 or 64.

Once again a pattern arises. Equating the people to digit positions we find that in the octal number system, digits in position 1 represent the number of ones (8^0), digits in position 2 represent the number of eights (8^1), digits in position 3 represent the number of eight eights (8^2), digits in position 4 represent the number of eight eight eights (8^3) and so on.

Let's examine the octal number 35276.

We assign positions as follows.

position → 5 4 3 2 1

3	5	2	7	6

position 1 contains 6 which represents 6 ones = 6×8^0.

position 2 contains 7 which represents 7 eights = 7×8^1

position 3 contains 2 which represents 2 eight eights = 2×8^2.

position 4 contains 5 which represents 5 eight eight eights = 5×8^3.

position 5 contains 3 which represents 3 eight eight eight eights = 3×8^4.

The total number is representable as the sum.

$6 \times 8^0 + 7 \times 8^1 + 2 \times 8^2 + 5 \times 8^3 + 3 \times 8^4$.

To evaluate the decimal equivalent to the octal number 35276 all we

I.B.6 (Cont.)

have to do is perform this summation and substitute appropriate values to the powers of 8.

$$35276_8 = 6 \times 8^0 + 7 \times 8^1 + 2 \times 8^2 + 5 \times 8^3 + 3 \times 8^4$$

$$= 6 \times 1 + 7 \times 8 + 2 \times 64 + 5 \times 512 + 3 \times 4096$$

$$= 6 + 56 + 128 + 2560 + 12288$$

$$= 15038_{10}$$

The octal number 35276 is equivalent to the decimal number 15038.

Fill the following box with the octal number 3752.

position → 4 3 2 1

Answer:

3	7	5	2

position 1 contains a ___ which represents ___ ones = ___ x 8^0. Answer: 2, 2, 2

position 2 contains a ___ which represents ___ eights = ___ x 8^1. Answer: 5, 5, 5

position 3 contains a ___ which represents ___ eight eights = ___ x 8^2. Answer: 7, 7, 7

position 4 contains a ___ which represents ___ eight eight eights = ___ x 8^3. Answer: 3, 3, 3

The octal number 3752 may be represented as the sum:

___ x 8^0 + ___ x 8^1 + ___ x 8^2 + ___ x 8^3. Answer: 2, 5, 7, 3

45_{10} is a number to the base ____. Answer: 10

45_{10} contains _____ 10's and _____ 1's. Answer: 4, 5

I.B.6 (Cont.)

45_8 is a number to the base _____. Answer: 8

45_8 contains _____ 8's and _____ 1's. Answer: 4, 5

Since 45_8 contains _____ 8's and _____ 1's, it is equivalent to the decimal number _____. Answer: 4, 5, 37

I.B.7

The base of a number system can now be seen to have significance not only as the number of digits contained in the number system but also with respect to digit position.

Expressing a whole number as a sum of products, each product is composed of the digit multiplied by the base of the number system raised to a power which is the number of positions from the right-most digit.

Restating this rule in terms of a formula we find that:

$$\text{Each Product} = \text{Digit} \times \text{Base}^{\text{Position from right-most digit}}$$

Take the octal number 456. Expressing this number as a sum of products we know immediately that since the number contains 3 digits it will be represented by the sum of 3 products. Digit 4 is two positions from the right-most digit 6 so that applying the formula we find the product to be:

$$4 \times 8^2 \leftarrow \text{Position from right-most digit}$$

↑ Digit ↑ Base

I.B.7
(Cont.)

Digit 5 is one position from the right-most digit 6 so that its product is

5×8^{1} ← Position from right-most digit

↑ Digit ↑ Base

Digit 6 is 0 positions from the right-most digit so that its product is

6×8^{0} ← Position from right-most digit

↑ *Digit* ↑ *Base*

The octal number 456 is thus expressed by the following sum of products.

$4 \times 8^{2} + 5 \times 8^{1} + 6 \times 8^{0}$

Indicate the value associated with the digit underlined in terms of its product.

$\underline{7}3215_{8}$ _____ Answer: 7×8^{4}

$340\underline{1}0_{8}$ _____ Answer: 1×8^{1}

32175_{8} _____ Answer: 5×8^{0}

$\underline{9}732_{10}$ _____ Answer: 9×10^{3}

$98\underline{5}6_{10}$ _____ Answer: 5×10^{1}

I. B. 7 (Cont.) Express the following numbers as sums of products.

37_8 _____

Answer: $3 \times 8^1 + 7 \times 8^0$

506_8 _____

Answer: $5 \times 8^2 + 0 \times 8^1 + 6 \times 8^0$

95_{10} _____

Answer: $9 \times 10^1 + 5 \times 10^0$

900_{10} _____

Answer: $9 \times 10^2 + 0 \times 10^1 + 0 \times 10^0$

I. B. 8 All this discussion about number systems has been leading up to the fact that with the use of computers we are faced with the problem of dealing with numbers to base systems other than ten. For this reason it is important to have some idea of what is meant by number systems.

The number systems often used when working with computers are decimal, octal, and binary.

The binary number system is the number system under which computers operate.

The octal number system is a convenient system by which binary numbers can be represented.

The decimal number system is, of course, the system of the user.

I.B.8 (Cont.) The use of numbers in each of these systems necessarily requires an ability to convert from one number system to another and also to perform the basic arithmetic operations of addition and subtraction in each of the systems.

Computers operate under which number system? _____ Answer: Binary

The octal number system is a convenient system by which _____ numbers can be represented. Answer: binary

I.B.9 Octal to Binary Conversion

Due to characteristics of the binary and octal number systems, octal numbers are easily converted to binary numbers and vice versa. This is due to the fact that when binary digits are taken 3 at a time, there is a one to one correspondence to octal digits as indicated in the following table.

Octal digit	Binary digits
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

I.B.9 (Cont.) Conversion from octal to binary is accomplished by simply substituting the appropriate 3-bit configuration for each octal digit.

For instance, the octal number 543 is converted by substituting for each octal digit as follows.

octal	5	4	3
binary	101	100	011

The octal number 543 is, therefore, equivalent to the binary number 101100011.

The octal number 75327 is similarly converted to binary as follows.

octal	7	5	3	2	7
binary	111	101	011	010	111

The octal number 75327 is, therefore, equivalent to the binary number 111101011010111.

Convert the following octal numbers to binary.

a.) 7523_8 _____

b.) 71_8 _____

c.) 54321_8 _____

d.) 1_8 _____

Answer: a.) 111101010011

b.) 111001

c.) 101100011010001

d.) 001

I.B.10 Binary to Octal Conversion

The conversion from binary to octal is just the converse of the octal to binary conversion. The digits of the binary number are grouped in sets of 3 from the right and appropriate octal digits are substituted. Zeros are placed preceding the binary number if necessary to make the bits an even multiple of three.

Converting the binary number 1101110011 to octal we first group the digits in sets of three from the right as follows.

$$\underbrace{00}\,\underbrace{1}\,\underbrace{101}\,\underbrace{110}\,\underbrace{011}$$

In this example two leading zeros were added to provide an even multiple of three.

Next the appropriate octal digits are substituted.

binary	$\underbrace{001}\,\underbrace{101}\,\underbrace{110}\,\underbrace{011}$
octal	1 5 6 3

The binary number 1101110011 is, therefore, equivalent to the octal number 1563.

The binary number 110000011 is converted to octal as follows.

binary	$\underbrace{110}\,\underbrace{000}\,\underbrace{011}$
octal	6 0 3

The binary number 110000011 is equivalent to the octal number 603.

I. B. 10 (Cont.) Convert the following binary numbers to octal.

a.) 1100110001 _____ Answer: a.) 1461_8

b.) 10 _____ Answer: b.) 2_8

c.) 1000011111 _____ Answer: c.) 1037_8

d.) 1010101 _____ Answer: d.) 125_8

I. B. 11 Decimal to Octal Conversion

Let's assume we wish to convert the decimal number 9543 to an octal number. We'll go through the conversion with a step by step explanation.

Step 1 Divide the number 9543 by 8.

$$8 \overline{)9543} \quad 1192 \quad \text{remainder} = 7$$

This division indicates that the decimal number 9543 contains 1192 eights with 7 left over. As you recall each digit position of an octal number represents from right to left increasing multiples of eight. The right-most position represents the number of units. Since the decimal number 9543 contains 1192 full cycles of 8 with 7 left over, the right-most digit of the converted octal number is 7.

I.B.11
(Cont.)

Step 2 Divide the number 1192 by 8.

$$8\overline{)1192}\quad 149 \qquad \text{remainder} = 0$$

This division indicates that the number of eight eights or 8^2 in the decimal number 9543 is 149 with nothing left over. The remainder in this division represents the number of full cycles of eight remaining after multiples of 8^2 are taken out which is in fact the second position from the right of the converted octal number.

The second position from the right of the octal number is, therefore, 0.

Step 3 Divide the number 149 by 8.

$$8\overline{)149}\quad 18 \qquad \text{remainder} = 5$$

This division indicates that the decimal number 9543 contains 18 full cycles of 8^3 with the remainder of 5. This remainder represents the multiple of 8^2 remaining after multiples of 8^3 are taken out. This is the third right-most position of the converted octal number.

The third position from the right of the octal number is 5.

Step 4 Divide the number 18 by 8.

$$8\overline{)18}\quad 2 \qquad \text{remainder} = 2$$

This division indicates that the decimal number 9543 contains 2 full cycles of 8^4 with a remainder of 2. The remainder represents multiples of 8^3 remaining after multiples of 8^4 are taken out and is the fourth position in the converted octal number.

I. B. 11
(Cont.)

The fourth position from the right of the octal number is 2.

Step 5

The 2 remaining in the quotient of step 5 is the number of times the decimal number 9543 cycles through 8^4.

The fifth position from the right of the octal number is 2.

From steps 1-5 the decimal number 9543 is equivalent to the octal number 22507.

Let's go through steps 1-5 once more in a more convenient manner. We'll perform continuous divisions by 8 and indicate each remainder to the side as follows.

Remember that each remainder represents a converted octal digit, the first remainder being the right-most digit, the second the next right-most and so on.

We'll again convert the decimal number 9543 to octal.

		remainder	
division 1	8 \9543		
division 2	8 \1192	7	(right-most octal digit)
division 3	8 \149	0	(2nd octal digit from right)
division 4	8\18	5	(3rd octal digit from right)
division 5	8\2	2	(4th octal digit from right)
		2	(5th octal digit from right)

The decimal number 9543 is, therefore, equivalent to the octal number 22507.

I.B.11 (Cont.) As one more illustration, lets convert the decimal number 792 to octal.

	remainder
8 \ 792	
8 \ 99	0
8 \ 12	3
8 \ 1	4
	1

The decimal number 792 is, therefore, equivalent to the octal number 1430.

Convert the following decimal numbers to octal.

a.) 38 _____

b.) 999 _____

c.) 9 _____

d.) 522 _____

e.) 6 _____

Answer:

a.) 46_8

b.) 1747_8

c.) 11_8

d.) 1012_8

e.) 6_8

I.B.12 Octal to Decimal Conversion

Conversion from octal to decimal can be accomplished by either representing the octal number as a sum of products and simply evaluating the summation or more rapidly through a mechanical procedure of multiplications and additions.

The first way can be illustrated as follows. Convert the octal number 2357 to decimal.

The octal number can be represented by a summation which when evaluated yields its decimal equivalent.

$$2357_8 = 7 \times 8^0 + 5 \times 8^1 + 3 \times 8^2 + 2 \times 8^3$$

$$= 7 \times 1 + 5 \times 8 + 3 \times 64 + 2 \times 512$$

$$= 7 + 40 + 192 + 1024$$

$$= 1263_{10}$$

The octal number 2357 is, therefore, equivalent to the decimal number 1263.

The second method of converting from octal to decimal is arrived at by factoring the equation which represents an octal number as a summation of products.

To be more specific, an octal number $n_3n_2n_1n_0$, where n_0, n_1, n_2, and n_3 are octal digits, can be represented as a sum of products.

I.B.12
(Cont.)

$$n_3n_2n_1n_0 = n_0 \times 8^0 + n_1 \times 8^1 + n_2 \times 8^2 + n_3 \times 8^3$$

This equation can be factored out as follows:

$$n_3n_2n_1n_0 = ((8n_3 + n_2) 8 + n_1) 8 + n_0$$

This equation sets up and defines the following conversion procedure.

The procedure is as follows:

1.	Multiply the left-most octal digit by 8	$8n_3$
2.	Add to this product the next octal digit	$8n_3 + n_2$
3.	Multiply this sum by 8	$(8n_3 + n_2)8$
4.	Add to this product the next octal digit	$(8n_3 + n_2)8 + n_1$
5.	Multiply this sum by 8	$((8n_3 + n_2)8 + n_1)8$
6.	Add to this sum the next octal digit	$((8n_3 + n_2)8 + n_1)8 + n_0$

I.B.12 (Cont.)

In order to illustrate this procedure, let's assume we wish to convert the octal number 22507 to decimal. The procedure is as follows.

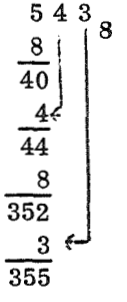
Operation	Illustration
1. Multiply the left-most octal digit by 8	2 2 5 0 7 8 16
2. Add the next octal digit to this product	2 18
3. Multiply this sum by 8	8 144
4. Add the next octal digit to this product	5 149
5. Multiply this by 8	8 1192
6. Add the next octal digit	0 1192
7. Multiply by 8	8 9536
8. Add the next octal digit	7 9543

The octal number 22507 is, therefore, equivalent to the decimal number 9543.

I.B.12
(Cont.)

Let's convert the octal number 543 to decimal.

1. Multiply the first digit by 8
2. Add the second digit
3. Multiply by 8
4. Add the third digit



The octal number 543 is equivalent to the decimal number 355.

Convert the following octal numbers to decimal.

a.) 12_8 _____

b.) 310_8 _____

c.) 5374_8 _____

d.) 77_8 _____

Answer:
a.) 10

b.) 200

c.) 2812

d.) 63

I.B.13 The rules for addition and subtraction in number systems, other than decimal, are essentially the same as those for decimal. The primary difficulty encountered in performing additions and subtractions in other number systems involves overcoming decimal number habits. We're all so accustomed to adding and subtracting in decimal that we are, in a way, faced with a psychological block when we have to perform these operations in any other number system. Of course prolonged exposure to octal addition and subtraction will eventually lead to a high degree of proficiency. It would become as mechanical or as natural as performing these operations in decimal. However, since our use of octal addition and subtraction will be extremely limited, we'll discuss octal operations in terms of decimal. In other words, although we'll add and subtract in octal, we'll think in decimal.

I.B.14 Octal Addition

In performing additions in octal, the same procedure is followed as in decimal with the following exception. When an octal digit is decimally added to an octal digit, the sum must be converted to octal if this decimal sum is larger than or equal to 8. The conversion is given in the following table.

decimal sum	octal sum
8	10
9	11
10	12
11	13
12	14
13	15
14	16

I.B.14
(Cont.)

This conversion can be easily remembered by noting that the octal numbers are two more than the decimal numbers.

Note also that the largest summation which can be performed in octal on a digit basis is 7 plus 7.

To demonstrate octal addition, let's add 73_8 to 57_8.

1. Add 7 to 3 decimally
   $7 + 3 = 10$
2. Since 10 is greater than 7, add 2
   $10 + 2 = 12$
3. Write down 2, carry 1
4. Add 7, 5 and the 1 carried over decimally
   $7 + 5 + 1 = 13$
5. Since 13 is greater than 7, add 2
   $13 + 2 = 15$

$$\begin{array}{r} 73_8 \\ 57_8 \\ 1 \\ \hline 2 \end{array}$$

$$\begin{array}{r} 73_8 \\ 57_8 \\ 1 \\ \hline 152_8 \end{array}$$

The octal sum of 73_8 and 57_8 is 152_8.

I. B. 14
(Cont.)

Let's now add 12357_8 and 2505_8 in octal.

1. $7 + 5 = 12$ 12 is greater than 7
   $12 + 2 = 14$
   Write down 4, carry 1

$$\begin{array}{r} 12357_8 \\ 2505_8 \\ {}^{1}{}^{1}{}^{1}{}^{1}{}^{8} \\ \hline 4 \end{array}$$

2. $5 + 0 + 1$ carry $= 6$
   Write down 6, no carry

$$\begin{array}{r} 12357_8 \\ 2505_8 \\ \hline 64 \end{array}$$

3. $3 + 5 = 8$ 8 is greater than 7
   $8 + 2 = 10$
   Write down 0, carry 1

$$\begin{array}{r} 12357_8 \\ {}^{1}2505_8 \\ \hline 064 \end{array}$$

4. $2 + 2 + 1$ carry $= 5$
   Write down 5, no carry

$$\begin{array}{r} 12357_8 \\ 2505_8 \\ \hline 5064 \end{array}$$

5. $1 + 0 = 1$
   Write down 1

$$\begin{array}{r} 12357_8 \\ 2505_8 \\ \hline 15064_8 \end{array}$$

The octal sum of 12357_8 and 2505_8 is 15064_8.

Add the following numbers in octal.

$35_8 + 77_8$ _____ Answer: 134_8

$7_8 + 7_8$ _____ Answer: 16_8

$25472_8 + 237_8$ _____ Answer: 25731_8

$7576_8 + 3260_8$ _____ Answer: 13056_8

I.B.15 Octal Subtraction

The rules for octal subtraction are essentially the same as those in decimal. Again, rather than actually perform the operations in octal, we'll set up procedures by which the actual arithmetic operations are carried out in decimal, but yielding octal results.

The primary rule to remember when subtracting in octal is that when a digit subtrahend cannot be subtracted from a digit minuend, 8 is added to digit minuend and 1 is taken away from the next digit minuend.

For example, if we wish to subtract $72_8 - 37_8$ octally, our first attempt is to subtract 7 from 2. Since this can't be done, we add 8 to 2 and take 1 from the 7 in the minuend next to the 2 and subtract as follows.

$$\begin{array}{r} 72_8 \\ -\,37_8 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{r} 6\ \ 8{+}2 \\ 3\ \ 7 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{r} 6\ \ 10 \\ 3\ \ 7 \\ \hline 3\ \ 3_8 \end{array}$$

The difference between 72_8 and 37_8 in octal is, therefore, 33_8.

Let's try another example.

Subtract $15742_8 - 555_8$ in octal.

In the first digit position 5 cannot be subtracted from 2. Add 8 to 2 and take 1 from the 4 in the next digit position. Then subtract decimally.

In the second digit position the minuend is now 3, but since the subtrahend is 5, it cannot be subtracted. Add 8 to 3 and subtract 1 from the 7 in the next minuend digit position. Then we subtract decimally.

```
  310
15742
  555
-----
    5

   11
 6810
15742
  555
-----
   65
```

I.B.15 (Cont.)

The rest is a straightforward subtraction.

```
   11
  6810
 15742
   555
-------
 15165
```

The octal difference between 15742_8 and 555_8 is 15165_8.

Perform the following subtractions in octal.

a.) $7532_8 - 3721_8$ _____

b.) $35_8 - 17_8$ _____

c.) $6657_8 - 77_8$ _____

d.) $72222_8 - 63333_8$ _____

Answer: a.) 3611_8

Answer: b.) 16_8

Answer: c.) 6560_8

Answer: d.) 6667_8

I.B.16

The rules for binary addition and subtraction are essentially the same as for octal and decimal. In some respects it's even easier since only two digit characters are ever involved.

Binary Addition

When adding binary numbers we're always adding either 0's or 1's to each other in each bit position. The following table lists all the summations which can occur by bit position when adding two binary numbers.

Both bits can be 0	0 + 0 = 0	no carry
One bit can be 0, the other 1	1 + 0 = 1	no carry
Both bits can be 1	1 + 1 = 0	1 carry
Both bits can be 1 plus a carry	1 + 1 + 1 = 1	1 carry

I.B.16 (Cont.)

Assume we wish to add the number 101110 to 011100 in binary.

In bit position 1 0 + 0 = 0, no carry	101110 011100 0
In bit position 2 1 + 0 = 1, no carry	101110 011100 10
In bit position 3 1 + 1 = 0, 1 carry	101110 011100 010
In bit position 4 1 + 1 + 1 = 1, 1 carry ↑ carry	101110 011100 1010
In bit position 5 0 + 1 + 1 = 0, 1 carry ↑ carry	101110 011100 01010
In last bit position 1 + 0 + 1 = 0, 1 carry ↑ carry	101110 011100 1001010

The sum of 101110 and 011100 in binary is, therefore, 1001010.

I. B. 16
(Cont.)

Let's try one more. We'll add 11101 to 01111.

In bit position 1	11101
1 + 1 = 0, 1 carry	0111$_1$1
	0
In bit position 2	11101
0 + 1 + 1 = 0, 1 carry	011$_1$$_1$1
↑ carry	00
In bit position 3	11101
1 + 1 + 1 = 1, 1 carry	01$_1$$_1$$_1$1
↑ carry	100
In bit position 4	11101
1 + 1 + 1 = 1, 1 carry	0$_1$$_1$$_1$$_1$1
↑ carry	1100
In last bit position	11101
1 + 0 + 1 = 0, 1 carry	0$_1$$_1$$_1$$_1$1
↑ carry	101100

The sum of 11101 and 01111 is 101100 in binary.

It should be noted of course that these additions have been performed in detail for illustrative purposes only. The addition is actually performed in one step.

I.B.16 (Cont.) Perform the following additions in binary.

a.) 110011 + 1 _____ Answer: a.) 110100

b.) 110100 + 111111 _____ Answer: b.) 1110011

c.) 111 + 111 _____ Answer: c.) 1110

d.) 10001 + 11001 _____ Answer: d.) 101010

I.B.17 Binary Subtractions

Although subtraction in binary can be carried out in essentially the same manner as in decimal, that is, subtracting digit by digit, this often becomes unwieldy. The reason for this is that since in binary we are dealing with only the two digit characters 0 and 1, we must frequently in subtraction borrow through a large number of digit positions containing zero in the minuend. This can cause some difficulty and confusion.

As an alternate approach, the difference between two binary numbers can be found by complementing the subtrahend, adding this to the minuend, and then adding any resulting carry. This is particularly suitable to binary numbers since the complement of a binary number is found by simply replacing 0's with 1's and 1's with 0's.

I.B.17 (Cont.)

This method of binary subtraction is illustrated in the following examples.

a.) Subtract in binary 110001 - 101111.

The minuend is 110001.
The subtrahend is 101111.

Complement the subtrahend	010000
Add the minuend	$_1$110001
Carry 1	000001
Add the carry	1
Difference	10

The difference between 110001 and 101111 is, therefore, 10 in binary.

b.) Subtract 100101001 from 111100000 in binary.

The minuend is 111100000.
The subtrahend is 100101001.

Complement the subtrahend	011010110
Add the minuend	$_1$111100000
Carry 1	010110110
Add the carry	1
Difference	010110111

The binary difference between 111100000 and 100101001 is, therefore, 10110111.

I.B.17 (Cont.) c.) Subtract 101 from 111010000 in binary.

The minuend is 111010000.
The subtrahend is 000000101.

Complement the subtrahend	111111010
Add the minuend	111010000
Carry 1	111001010
Add the carry	1
Difference	111001011

The difference between 111010000 and 101 is, therefore, 111001011.

Perform the following subtractions in binary.

a.) 101010101 - 111 _____

b.) 111000 - 101001 _____

c.) 10111 - 10000 _____

d.) 1100 - 1 _____

Answer: a.) 101001110

Answer: b.) 001111

Answer: c.) 111

Answer: d.) 1011

I.B.18 Fractional Conversions

Conversion to Decimal

Before concluding our discussion of number systems, some brief mention will be made of fractional numbers.

We are all familiar with fractional numbers in the decimal number system and recognize that digits to the right of the decimal point define the fractional portion of the number. When we represent a decimal number as a sum of products, we find that the products obtained from digits to the right of the decimal point are a logical extension of the way products are obtained from digits to the left of the decimal point. This can be illustrated by representing the decimal number 29.35 as a sum of products.

$$29.35 = 2 \times 10^{1} + 9 \times 10^{0} + 3 \times 10^{-1} + 5 \times 10^{-2}$$

From this example, it can be seen that products are formed with digits to the right of the decimal point by multiplying the digit by 10 raised to the negative power of its position to the right of the decimal point.

Analogously, fractional numbers in other number systems are represented by sums of products in the same way.

In the octal number system, products are formed with digits to the right of the octal point by multiplying the digit by 8 raised to the negative power of its position from the octal point.

I.B.18
(Cont.)

The octal number 43.24 is represented by the following sum of products.

$$43.24_8 = 4 \times 8^1 + 3 \times 8^0 + 2 \times 8^{-1} + 4 \times 8^{-2}$$

Note that the conversion of the octal number 43.24 to decimal is simply the evaluation of the sum of products as follows:

$$43.24_8 = 4 \times 8 + 3 \times 1 + \frac{2}{8} + \frac{4}{8^2} = 32 + 3 + .25 + .0625 = 35.3125_{10}$$

Binary digits to the right of the binary points are handled in a similar manner. The binary number 101.11 is represented by the following sum of products.

$$101.11_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

Represent the following numbers as sums of products.

a.) $.95_{10}$ _____

b.) 83.2_{10} _____

c.) $.53_8$ _____

d.) $.02_8$ _____

e.) 7.1_8 _____

f.) 11.1_2 _____

g.) $.01_2$ _____

Answer:

a.) $9 \times 10^{-1} + 5 \times 10^{-2}$

b.) $8 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1}$

c.) $5 \times 8^{-1} + 3 \times 8^{-2}$

d.) $0 \times 8^{-1} + 2 \times 8^{-2}$

e.) $7 \times 8^0 + 1 \times 8^{-1}$

f.) $1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$

g.) $0 \times 2^{-1} + 1 \times 2^{-2}$

I. B. 18
(Cont.)

Decimal to Octal

Fractional decimal numbers are converted to octal by successive multiplications by 8. The decimal fraction is multiplied by 8 and the integer portion becomes the first octal digit. The fractional portion of the product is then multiplied by 8 and the integer result becomes the second octal digit. This is continued as far as desired. The following example illustrates this process.

Convert the decimal fraction .596 to octal.

Step 1	Multiply by 8. 4 is the first converted octal digit after the octal point.	.596 8 ④.768
Step 2	Multiply the fractional portion by 8. 6 is the second octal digit.	.768 8 ⑥.144
Step 3	Multiply the fractional portion by 8. 1 is the third octal digit.	.144 8 ①.152
Step 4	Multiply the fractional portion by 8. 1 is the fourth octal digit.	.152 8 ①.216

The result so far is $.596_{10} = .4611_8$

By continuing this process the converted octal number can be carried out to as many places as desired.

Convert the following decimal fractions to octal.

a.) $.5_{10}$ _____

b.) $.3125_{10}$ _____

c.) $.0625_{10}$ _____

Answer: a.) $.4_8$

b.) $.24_8$

c.) $.04_8$

I.C Basic Computer Concepts

I.C.1 Although the base of a number system can be any number equal to 2 or more, decimal, octal, and binary number systems are of particular importance when related to computers.

The decimal number system is of importance simply because that's the system we're all used to and expect to use in basic communication with the computer.

The binary number system is the number system of a computer.

The octal number system is a convenient system in which to represent numbers in the binary number system.

The three number systems of particular importance in relation to computers are_____, _____, and _____.

Answer: decimal
octal
binary

I.C.2 Why is the binary number system the number system of computers?

The binary number system as we now know contains only 2 digits. This is significant!

How can the 2 digits 0 and 1 be represented in the binary number system?

Of course we can just write it, but physically how can we represent it?

How about up and down--up is 1 and down is 0.

How about wet and dry--wet is 1 and dry is 0.

White and black, open and closed, right and left, full and empty, on and off--all of these conditions can be used to represent a 0 and 1 in the binary number system.

A computer uses an on and off condition to represent the binary digits 0 and 1. 1 is an on condition and 0 is an off condition.

A binary digit in a computer is called a bit--a contraction of the words binary and digit.

The binary digit 0 in a computer is represented by an _____ condition. Answer: off

The binary digit 1 in a computer is represented by an _____ condition. Answer: on

A bit is a _____ _____. Answer: binary digit

I.C.3 Specifically, a bit position in a computer is considered in an "on" state when magnetized and in an "off" state when non-magnetized.

I.C.4 As you recall, a number is defined not only by the number of digits contained in its number system, but also by digit position.

Although we've explained how a computer recognizes the binary digits 0 and 1, we have not as yet discussed bit positions--an essential requirement for representing a number.

In a computer a certain number of bit positions are assigned to a "computer word". The number of bit positions contained in a computer word vary with computers. The CDC 6600 has a computer word made up of 60 bit positions--a word in the CDC 3200 computer contains 24 bit positions--the UNIVAC 1108 and IBM 7094 have computer words containing 36 bit positions.

The number of bit positions contained in a computer word establishes the magnitude of the number that can be represented by that computer word. The more bit positions assigned to a computer word, the larger the number that can be represented.

A computer word is composed of a fixed number of _____.

Answer: bits or bit positions

I.C.5 Computer words are the basic structure of any digital computer. Computer words can be recognized in one of two ways by a computer, as instructions or as numbers.

The instruction repertoire of a computer is a function of computer design. Every digital computer has its own set of basic instructions defined in terms of a fixed bit configuration within a computer word.

Computer words are recognized by computers as either _____ or _____.

Answer: instructions, numbers

I.C.6 Before we discuss in any more detail the role played by computer words, let's back away for a moment and discuss, in general, how a computer operates.

An analogy can be made between computers and the human brain. The term "electronic brain" is commonly applied to computers and justifiably so. A computer is composed of memory cells which can retain or provide information through a suitable arrangement of instructions. This arrangement of instructions is called a computer program.

This is not dissimilar to the brain. It too, in a sense, has available memory cells which through the learning process become programmed to perform various functions.

The computer, like the brain, requires input data to operate on. Input data are transmitted to the brain through 5 senses which are collected by physical devices hooked on to the body--eyes, ears, nose, mouth, etc.

I.C.6 (Cont.)

So too with a computer. Physical devices such as optical scanners, punched cards, paper tape, magnetic tape, magnetic disks, etc. are all available for transmitting information to the computer. These are called input devices--devices transmitting information to the computer.

Output devices used by the brain include speech, writing, body movement, facial expressions, etc. Output devices available to computers include printers, plotters, magnetic tapes, oscilloscopes, typewriters, etc.

So, in order for a computer to work it must have:

1. input capability
2. a resident computer program
3. output capability

Let's follow through the steps taken when a problem is posed for computer solution.

To make these steps more meaningful we'll assume an actual problem and illustrate the steps taken in terms of this problem.

Assume the following problem:

Given a quadratic equation $0 = c + 12x - 15x^2$, solve for x for all values of "c" from 1 to 100 in steps of 1.

I.C.6
(Cont.)

Step 1

Surprisingly enough, the first step is to solve the problem. It must be remembered that computers do not solve problems, programs do! All computers do is to provide a means by which instructions can be carried out rapidly, repetitively, and consistently. The programmer, knowing the instructions available to a computer, must decide how a problem is to be solved. A single problem can be solved in many ways on the same computer. The method of solution is left to the discretion of the programmer.

Problems are solved by _____ _____.

Answer: computer programs

Computer programs provide instructions to be carried out by _____.

Answer: computers

The solution to a quadratic equation, as we recall from high school algebra is:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Substituting values from our sample quadratic equation, we find the solution to be:

$$x = \frac{-12 \pm \sqrt{144 + 60c}}{-30}$$

This is the solution to the problem and it is this equation that we will program. It is very important to realize that solutions are programmed, not problems.

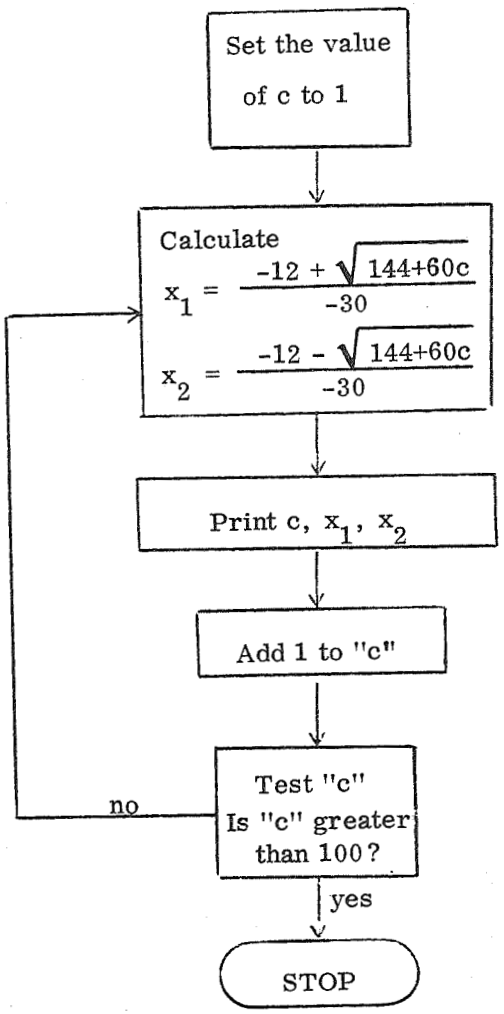
I. C. 6
(Cont.)

Step 2

This step establishes the logic to be employed in solving the problem. It is at this point that a flow chart is drawn to illustrate the method of solution. Flow charts will be discussed in more detail further on in the course.

The solution to our sample problem could be flowcharted as follows. It should be noted here that this flow chart does not conform to any standard convention, but is presented for simple illustrative purposes.

I. C. 6
(Cont.)


Set the value of c to 1
Calculate
$x_1 = \frac{-12 + \sqrt{144+60c}}{-30}$
$x_2 = \frac{-12 - \sqrt{144+60c}}{-30}$
Print c, x_1, x_2
Add 1 to "c"
Test "c"
Is "c" greater than 100?
no
yes
STOP

I.C.6
(Cont.)

If you examine this flow chart, you'll notice that in order to solve our problem we must

1.) perform the calculation
2.) printout the results
3.) repeat the calculation and printout for a total of 100 values of c.

These are, in general terms, the functions to be performed by the program. How these functions are specifically performed depends on the instruction repertoire of the computer or the computer language being used.

For now let's accept the fact that somehow instructions can be written to perform all the functions designated in the flow chart and let's see what happens next.

Step 3

Now that we know how we are going to solve our problem, we are prepared to write our program on coding sheets. Programs are written on coding sheets for convenience since they indicate how our written instructions are to be punched onto cards.

Coding sheets are designated to simulate computer cards. On a coding sheet each line is divided into 80 columns corresponding to the 80 columns contained on a card. What is written on one line of a coding sheet will be punched in the designated columns, on one computer card.

I.C.6 (Cont.)

How do we know what columns to write our instructions in?

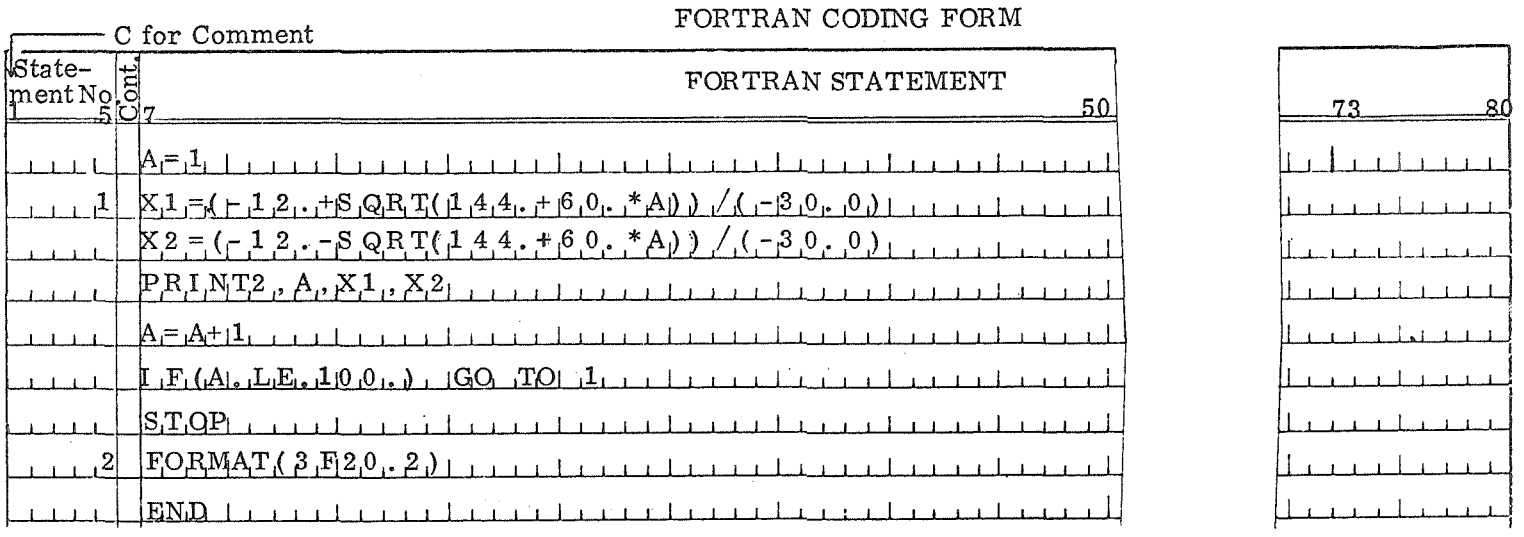
This is strictly a function of the rules of the language in which we're writing. In FORTRAN very specific rules are defined with respect to the columns in which information can be punched. This will be discussed later on in the course.

Computer cards contain _____ columns. Answer: 80

A line on a coding sheet is divided into _____ columns. Answer: 80

Information written on 3 lines of a coding sheet will be punched onto how many computer cards? _____ Answer: 3

Back to our sample problem! We're ready to write the program. Let's take the plunge and write the program in FORTRAN--a preview of things to come. In our program, the symbol A is used to represent c in the formula.



I.C.6
(Cont.)

These 9 lines of FORTRAN instructions constitute an "honest to goodness" FORTRAN program. It follows quite faithfully the logic we've outlined in our flow chart.

Don't panic now! Recognize this for what it is--a preliminary exposure to a FORTRAN program. It is meant to give you some idea of what a FORTRAN program looks like.

Let's examine this program line by line and get a general idea of what these instructions mean. Whatever you get out of this preliminary exercise will prove worthwhile as an aid in later reading.

Line 1 of our program, A = 1, instructs the computer to assign a value of 1 to the symbol A.

Line 2 performs the calculation and assigns the result, the first root of the equation, to the symbol X1.

Line 3 performs the calculation and assigns the result, the second root of the equation, to the symbol X2.

Line 4 prints out A, X1, and X2 on a printer attached to the computer. The results are printed out on the page according to a format defined on line 8. Line 8 simply tells the computer how to physically print out results on the output page.

Line 5 adds 1 to the value of A. A will now be 1 larger than it was before.

Line 6 tests to see if A is less than or equal to 100. If it is, it tells the computer to go back to line 2 and repeat lines 2, 3, 4, and 5 until the value of A is greater than 100. When A is greater than 100, the computer is instructed to go to line 7.

I.C.6 (Cont.)

Line 7 tells the computer to stop executing instructions for this program.

Line 8 provides format instructions for the print instruction on line 4.

Line 9 contains an instruction, END, which indicates to the computer that this is the last card of the program.

After the program has been written on a coding sheet, the next step is to have the program punched onto computer cards, our means of communication with the computer.

Step 4

The computer cards are now placed in the card hopper of the card reader attached to the computer. Some buttons are pressed on the computer console and each card is sequentially read by the card reader, information from each card being transmitted by the card reader to the computer which in turn stores the information into appropriate memory locations.

After all of the program has been read into the computer, the computer is instructed to execute the instructions of the program which now resides in the computer. As a result of the execution of the program, answers are output by the computer on the printer.

I.C.6 (Cont.)

Summarizing all of this we find that in order to solve a problem on a computer, four basic steps are required. They are:

Step 1	The solving of the problem
Step 2	The flowcharting of the solution
Step 3	The writing of the program on coding sheets and the punching of the program instructions onto computer cards.
Step 4	The input of information contained on the cards to the computer and the execution of the program.

A problem must be _____ before it can be programmed.

Answer: solved

A _____ should precede the writing of a program.

Answer: flow chart

A program is written on _____ _____.

Answer: coding sheets

From coding sheets instructions are _____ onto computer cards.

Answer: punched

Computer cards are read and transmitted to a computer by a _____ _____.

Answer: card reader

Before a program can be executed, the cards containing the program must be _____ _____ _____ _____.

Answer: read into the computer.

I.C.7 The address of a computer word is the location of the computer word in memory. All computer words have an address in computer memory.

Computer instructions are contained in computer words in successive addresses and are normally executed sequentially, calling for data, as necessary by address, from other areas of memory.

The location of a computer word in memory is called the _____ of the computer word.

Answer: address

I.C.8 As it turns out, machine language programming can prove quite tedious. This has led to the generation of program processors which convert programs written in one language to machine language programs.

One type of program processor is called a compiler. FORTRAN compilers fit into this category.

A program written in FORTRAN is called a source program. It is on this source program that the FORTRAN compiler operates in order to produce a machine language program called the object program.

Before a FORTRAN program can be executed, it must be converted to a machine language program by the FORTRAN _____.

Answer: compiler

A FORTRAN program is called a _____ program.

Answer: source

The machine language program generated by the FORTRAN compiler is called the _____ program.

Answer: object

I.C.9 Assuming that cards containing a FORTRAN source program followed by appropriate data are prepared for input to the computer, the following steps are entered into; (Note: automatically, as far as the programmer is concerned).

1.) The FORTRAN compiler is read into memory.

2.) The cards containing the FORTRAN source program are read into memory and converted to a machine language object program by the FORTRAN compiler.

3.) The machine language object program is then executed.

The FORTRAN compiler performs quite a feat for the programmer when one considers the amount of bookkeeping, writing, and complexity normally found in machine language programming.

It should be added here that in addition to relieving the programmer of machine language programming, the compiler also provides a large store of diagnostic messages for the programmer to help him find program errors, commonly called "bugs". The act of finding and correcting errors in a program is called "debugging a program".

I.C.10 Communication with the computer on a computer word basis is accomplished by using octal numbers. That is, if one desires to find out what is contained in a specific computer word or if one wants a computer word to contain a specific binary number, octal numbers are used.

Octal numbers are used because they are equivalent to binary numbers, each octal digit representing three binary digits. Twenty octal digits are required to represent a 60-bit CDC 6600 computer word.

For illustrative purposes, let's assume we're working with a 12-bit computer word. This 12-bit word requires 4 octal digits to represent it; again, each octal digit covering 3 bits.

Let's represent the binary number 111011001010 in octal. This is done by taking the bits in sets of three and substituting the appropriate octal digit for each set as follows:

binary number	111	011	001	010
octal number	7	3	1	2

The octal number 7312, therefore, represents the binary number 111011001010.

Trying it the other way we find that the octal number 2176 represents the binary number 010001111110. This is obtained as follows:

octal number	2	1	7	6
binary number	010	001	111	110

I. C. 10 (Cont.)

Write the following binary numbers in octal.

101001 _____ Answer: 51

110001000 _____ Answer: 610

100001111010 _____ Answer: 4172

Write the following octal numbers in binary

42 _____ Answer: 100010

730 _____ Answer: 111011000

7777 _____ Answer: 111111111111

I.C.11 Character information, such as letters, symbols, and numbers, is recognized by computers in a special way. It should be realized that the use of character information is extremely important. Not only is a computer called on to perform calculations, but also it is required to process character information in order to provide messages, to label information, to identify items, etc.

The way a computer is designed to recognize characters is by means of a 6-bit code. The CDC 6600 defines its 6-bit coding of characters by a console display code. Some other computers call their character coding BCD which stands for binary coded decimal. This character coding is not standard so that a 6-bit code which represents a character on one computer is not necessarily the code for the same character on another computer.

Let's cite an example. A CDC 6600 computer word containing the word "TABLE" in console display code would contain the following.

55555555552401021405 (octal)
b b b b b T A B L E

These 20 octal digits represent the 60 bits contained in a CDC 6600 computer word.

I.C.11 (Cont.)

A table is provided at the end of the CDC FORTRAN Reference Manual which lists the console display codes and the characters to which they are equivalent. From this table the following was obtained.

Character	Console Display Code
blank	55
T	24
A	01
B	02
L	14
E	05

Looking at the computer word containing the word "TABLE" we find that the first 6 bits are 55_8 (binary 101101) which is a blank in console display code. This is followed by four more blanks. After the five blanks comes the letter "T". in console display code a 24_8 (binary 010100), followed by an "A", in console display code a 01_8 (binary 000001), then a "B", "L", and finally "E".

I.C.11 (Cont.) Write the following words in console display code for the CDC 6600 using the table of codes provided. Use leading blanks to fill up the 60-bit word.

Console Display Code	Character
01	A
02	B
03	C
04	D
05	E
06	F
07	G
10	H
11	I
55	blank

ACHE _____ Answer: 5555555555501031005

CHIEF _____ Answer: 5555555550310110506

BAGGAGE _____ Answer: 5555502010707010705

GLOSSARY

Address	The location of a memory cell or a symbol representing that location.
ADP	Automatic Data Processing
Algorithm	A rule of procedure for solving a problem.
Allocation	The allotment of storage for data and other information.
Alphanumeric	This word is a contraction of Alphabetic-Numeric and refers to the character set consisting of the letters of the alphabet and the ten decimal digits.
Analog	The representation of numerical quantities by means of physical variables; e. g., translation, rotation, voltage, or resistance. Contrasted with "digital".
AND	A logical operation defined in Boolean algebra.
Argument	The independent variable of a function.
Array	A series of items arranged in a meaningful pattern.
Assemble	To translate a program written in a synthetic language into machine language and to assign storage for instructions and data.
Auxiliary Equipment	Equipment not under direct control of the central processing unit.
BCD	Abbreviation for Binary-Coded-Decimal. (See section I.C.11)
Bit	Contraction of binary digit.
Block	A group of consecutive records, words, or characters handled as one unit.
Block Transfer	Movement of a number of consecutive computer words from one position in storage to another.

GLOSSARY

Boolean	Related to the logical arithmetic developed by George Boole.
Buffer Storage	A device which temporarily stores information during a transfer of information.
Bug	An error in a program.
Byte	A term indicating a fixed number of consecutive binary digits.
Calling Sequence	The set of instructions used to link a subroutine with a main routine.
Card	A machine processable information storage medium of special quality paper stock.
Card Punch	A device to record information in cards by punching holes in the cards to represent letters, digits, and special characters.
Card Reader	A device which senses and translates into internal form the holes in punched cards.
Card-To-Tape	Refers to the transfer of information from punched cards to magnetic tape.
Central Processing Unit	That component of a computing system which contains the arithmetic, logical, and control circuits of the basic system.
Chain	A series of items linked together.
Checkout	The application of diagnostic or testing procedures to produce a properly working program.
Checksum	A summation of digits or bits used primarily for checking purposes.

GLOSSARY

Compiler	A program processor which translates from a synthetic language such as FORTRAN to machine language.
Complex Number	A number consisting of a real and an imaginary part.
Console Display Code	(See section I.C.11)
Control Card	A control card provides information to the monitor or compiler.
Core Storage	A form of high speed storage using magnetic cores.
Data Processing	Manipulating data to achieve desired results.
Data Reduction	The process of transforming raw data to a form more suitable for analysis. This frequently requires smoothing, adjusting, scaling, and ordering of the raw data.
Debug	Same as checkout.
Deck	A set or pack of cards.
Diagnosis	The process of locating and explaining errors in a computer program.
Digitize	To convert from analog to digital form.
Disk Storage	A storage device which uses magnetic recordings on flat rotating disks.
Display Unit	A device which provides visual representation of data.
Double Precision	Pertains to the quantity containing twice the number of digits normally carried.
Double Punch	A term which refers to more than one punch in any one card column.

GLOSSARY

Downtime	The elapsed time occurring due to machine failure.
Dump	To print out the contents of part or all of some storage medium.
EDP	Abbreviation for Electronic Data Processing.
End of File Mark	A one character indicator on tape designating an end of file.
File	A systematic collection of information often consisting of a number of records.
File Protect Ring	A ring which when placed on a tape reel enables the tape to be written on. With this ring off, the tape can be read, but not written on.
Fixed Point Number	A fixed point number in FORTRAN refers to an integer.
Flag	A symbol used to provide a signal or indication of some condition (e. g., good, bad, or questionable data).
Floating Point Number	A number which is represented by the digits of the number plus an indicator denoting the location of the decimal point.
Flow Chart	A graphic representation of a problem in terms of data flow, procedures, methods, etc.
Format	A predetermined arrangement of characters, fields, lines, punctuation, page numbers, etc.
FORTRAN	A contraction of the words "formula translator". This algebraic type language is widely used and can be compiled on many different computers.
Hardware	The mechanical, magnetic, and electronic components of a computer.
Header Record	A record containing identifying or explanatory information for a group of records which follow.

Hollerith Code	An encoding scheme by which any one of a set of 50 characters may be represented in one column of a card. Named for Herman Hollerith, the originator.
Initialize	To set certain counters, switches, or addresses at specified times in a computer program.
Input	Information transferred from auxiliary or external storage into internal storage.
Input-Output	Commonly called I/O which refers to equipment or data involved with the information transferred into the computer and information transferred out.
Iteration	The continued repetition of the same operation or group of operations.
Left Adjusted	The placing of information such that any unused area is on the right.
Library Routine	A routine placed on file and available for general use.
Linear Programming	A technique used in mathematics and operations research to find a best solution for a certain class of problems.
Location	A place in storage where a unit of data may be stored.
Logical Operation	An operation or instruction which operates on the independent bits of a computer word.
Loop	A coding technique in which a group of instructions are repeated.
Magnetic Core	A small doughnut-shaped ferrite designed and constructed for on or off magnetization and used to store information in the computer.
Mask	A fixed word pattern of bits used for the purpose of selecting or eliminating bit positions from other words.
Matrix	An array of quantities in a prescribed form. The elements are usually arranged in rows and columns.

GLOSSARY

Memory	A memory is a device in which information can be stored. The term "memory" is normally used with reference to quick access devices such as magnetic cores.
Mnemonic	A mnemonic in coding refers to a symbol or word chosen to be similar to the name of the item it represents (e. g., THETA for θ, TWOPI for 2π , etc.).
Mode	A computer system of data representation; e. g., the binary mode.
Modulo	A mathematical operator which yields the non-negative remainder function of division.
Monitor	A program processor which exercises supervisory control over some other program or collection of programs.
Multiprocessing	Solution of a problem by coordinated action of several computers.
Multiprogramming	Solution of several problems simultaneously on a single computer.
Nesting	Including a routine or block of data within a larger routine or block of data.
Number System	The representation of a quantity by a positional value to a given number base.
Numerical Analysis	The study of methods for obtaining numerical answers to mathematically stated problems.
Object Program	The machine language program which is the output after translation from the source program.
Off-line	Pertaining to the operation of input-output devices or auxiliary equipment not under direct control of the central processing unit.
On-line	Operation of an input-output device as a component of the computer under programmed control.
Open Shop	A computing installation at which computer programming is performed by any qualified employee.

GLOSSARY

OR	A logical operation defined in Boolean algebra.
Output	Information transferred from internal storage to external storage or to an output device.
Parameter	A quantity to which arbitrary values may be assigned.
Parity Check	A special form of validity check in which one bit of the binary code is reserved as a parity bit and is set equal to either zero or one to make the total number of one-bits in the code consistently even (even parity check) or consistently odd (odd parity check).
Printer	A device which prints output.
Program	A series of instructions written in order to solve a problem on a computer.
Reader	A device which senses information from one form of storage and converts and transmits it to information in another form of storage.
Real Number	A real number in FORTRAN refers to a single precision (as opposed to double precision) floating point number in the real domain.
Record	A group of related facts or fields of information treated as a unit. On magnetic tape, separated by a record gap.
Record Gap	A space between records on a magnetic tape.
Right Adjusted	The placing of information such that any unused area is on the left.

GLOSSARY

Routine	A series of instructions which carry out a well defined function.
Scratch Tape	A magnetic tape used for intermediate results rather than for input, output, or filing.
Shift	A movement of bits, digits, or characters to the left or right.
Sign Bit	A bit stored with a binary number to indicate the algebraic sign.
Sort	To arrange the items of a file in a specified order.
Source Language	The form in which the program is written on the coding sheet.
Storage	A general term for any device capable of retaining information.
Subroutine	The set of instructions necessary to direct the computer to carry out a well defined mathematical or logical operation at the request of another routine.
Table	One or more lists containing organized information.
Track	A single longitudinal path as on magnetic tape.
Transfer	An instruction which can alter the regular sequence of instruction execution.
Turnaround Time	The amount of time which elapses from submission of inputs for a computer run until the output for that run is available.
Word Length	The number of bits or characters handled as a physical unit by the computer.

PART II
FORTRAN BASICS

II.A FORTRAN Arithmetic Statements

II.A.1 The FORTRAN arithmetic statement is designed to have the form of a mathematical formula. The formula $a = x + y$, for example, is written in FORTRAN as A = X + Y.

There is one arithmetic operation performed in the statement A = X + Y. It is _____.

Answer: addition

II.A.2 Addition is one of the basic arithmetic operations. Others are subtraction, multiplication, and division. The FORTRAN language uses the symbol + to indicate addition and -, *, and / for subtraction, multiplication, and division, respectively.

Each of the following FORTRAN expressions use two basic arithmetic operations. Indicate the missing operation in each case.

A + B + C	addition and _____
A + B - C	addition and _____
A - B / C	subtraction and _____
A * B - C	_____ and subtraction

Answer: addition
subtraction
division
multiplication

II.A.3 The FORTRAN expression A * B causes the value of A to be multiplied by the value of B. The expression A * B - C will multiply the value of A by the value of B then subtract the value of C from the result.

If the value of A is 20.0, B is 5.0, and C is 1.0, the result of the expression A * B - C is _____.

Answer: 99.0

II.A.4 The = in the FORTRAN arithmetic statement means to take the result of the expression on the right and assign that value to the symbol on the left. The symbol on the left then retains that value until it is assigned a new value by another statement.

If the value of A is 3.0, then the value of X after the statement X = A is 3.0. If the value of A is 8.0, what is the value of X after the statement X = A + A? _____

Answer: 16.0

II.A.5 Another basic arithmetic operation permitted in FORTRAN is exponentiation. This operation is indicated by two *'s. The expression A * * K raises A to the Kth power.

List the symbols which correspond to the five operations listed below. These symbols are called operators.

addition _____
exponentiation _____
multiplication _____
subtraction _____
division _____

Answer:
+
* *
*
-
/

II.A.6 If you had any difficulty with the last problem, start PART II over again. See your advisor if it is not clear after the second pass through the material. If you understand the material to this point, you may continue.

II.A.7 A FORTRAN statement may contain several different operations. For example, the statement D = X * * 2 + Y * * 2 + Z * * 2 contains both exponentiation and addition. The statement D = X * X + Y * Y + Z * Z is equivalent to the one above but contains multiplication and addition.

The statement A + D * * 3 + B / X contains the three operations _____, _____, and division.

Answer: exponentiation
addition

II.A.8 The order in which the operations are performed in a particular FORTRAN expression is determined by a few basic rules. The importance of knowing these rules is illustrated by the FORTRAN expression W * G - P / Q * R. Without a definite order, this expression could represent any of the following arithmetic formulas.

1) (wg - p) / (qr)
2) w (g - p) / (qr)
3) ((wg - p)/q) r
4) wg - (p / q) r
5) wg - (p / (qr))

A FORTRAN arithmetic expression must represent one and only one arithmetic formula. Therefore, it is necessary to have a set of rules which govern the _____ in which the calculations are performed.

Answer: order

II.A.9 In a FORTRAN expression, all exponentials are evaluated before the other operations are performed. The exponentials are evaluated in the order they occur from left to right.

In the expression X * * 2 + D * Y , the first operation performed is _____.

Answer: X * * 2

II.A.10 After all exponentials are evaluated, the multiplications and divisions are then performed in the order they occur from left to right.

In the expression A * X - Y / D, the second operation performed is _____.

Answer: Y / D

II.A.11 After all exponentials, multiplications, and divisions have been performed, the additions and subtractions are performed in the order they occur from left to right. Thus, the three-level hierarchy of operations is:

1) exponentiation
2) multiplication and division
3) addition and subtraction

Indicate the first operation which will be performed in each of the following statements:

A = R + B - C _____
C = B * X * * 2 - Q / D _____
X = Y * * 2 + Z * * 2 _____
Y = A * B / C _____

Answer: R + B (addition)
X * * 2 (exponentiation)
Y * * 2 (exponentiation)
A * B (multiplication)

II.A.12 If you had no difficulty with the last problem, give yourself a gold star and go to II.A.14. Otherwise, continue.

II.A.13 Now let us consider the operational hierarchy of each statement in II.A.11.

1.) A = R + B - C

There are no exponentiations, multiplications, or divisions. Thus, the only operations involved all belong to the third level of the operational hierarchy and are performed in the order encountered from left to right.

2.) C = B * X ** 2 - Q/D

In this example, there are operations from all levels of the hierarchy. The operation which is performed first in this case is the exponential, X ** 2. Next the multiplications and divisions are performed in the order they occur from left to right. Next the additions and subtractions are performed. In this case only one subtraction is encountered.

3.) X = Y * * 2 + Z * * 2

In this case, there are two exponentials to be performed and they are carried out in the order they occur from left to right. Thus, the operation Y * * 2 is performed first. After the second exponential, Z * * 2, a search is made for multiplications or divisions. Since there are none, a third scan is made to find additions and subtractions. Thus, the last operation is the addition of the two exponentials.

4.) Y = A * B / C

Here all operations belong to the second level of the operational hierarchy. Therefore, they are performed in the order they occur from left to right. In this case, we have the operation A * B. The next operation divides this result by C.

II.A.14 The statement B = X / A * C always corresponds to the arithmetic formula $b = \frac{x}{a} \cdot c$. This is true because multiplication and division have equal priority and the division occurs first in the statement.

Write the arithmetic formula which corresponds to the FORTRAN statement X = G / T * * 2 * H. _____

Answer: $x = \frac{g}{t^2} \cdot h$

II.A.15 Parentheses may be used to change the order of operations. Think about expressing $s = \frac{a}{db}$ as a FORTRAN statement.

The FORTRAN statement S= A/(D*B) corresponds to the arithmetic formula $s = \frac{a}{db}$.

The order in which the operations are performed may be changed by the use of _____.

Answer: parentheses

II.A.16 The use of the parentheses in FORTRAN expressions is essentially the same as the use of parentheses in arithmetic formulas. The first and most important thing to remember is that parentheses must be used in pairs.

The number of left parentheses in a FORTRAN statement must equal the number of _____ parentheses.

Answer: right

II.A.17 Nesting of parentheses is permitted. This means that a pair of parentheses may lie entirely within another pair. When parentheses are nested, the expression within the innermost pair of parentheses is evaluated first.

In the statement A = ((B - C) * D) /R, which operation is performed first? _____

Answer: B - C

II.A.18 The expression within the innermost set of parentheses is evaluated according to the hierarchy of operations discussed earlier.

The three levels of the operational hierarchy are _____, _____ and _____, and _____ and _____.

Answer: exponentiation, multiplication, division, addition, subtraction

II.A.19 Once the value of the expression within a set of parentheses has been obtained, that value is then used in evaluating the expression contained within the next outward set of parentheses.

The expression (X + Y) / (A + D) will add the value of X to the value of Y and then divide the result by the sum of _____ and D.

Answer: A

II.A.20 It is usually obvious when parentheses are needed to obtain the desired result. Some situations arise, however, when it is not immediately apparent that parentheses are required. A good rule to follow is to use parentheses if there is any doubt. No harm results from extraneous parentheses as long as they are used in a meaningful way.

Pick the expression where a set of parentheses are used incorrectly.

1. (A + B) * C 2. ((A + B)) * C 3. ((A + B) *) C

Answer: 3

II.A.21 If your answer was 1 , go back to II.A.14. If your answer was 2 , remember that too many parentheses are not harmful as long as the expression is meaningful. Expression 3 has no meaning and is, therefore, an incorrect use of parentheses.

II.A.22 One common error in FORTRAN is the omission of the multiplication symbol. In FORTRAN, all operations must be indicated by the use of the proper symbol.

The expression (A + B) (-D) is incorrect because no operation _____ appears between the two parentheses.

Answer: symbol

II.A.23 The expression (-D) is a proper use of the negative operator. Care must be taken, however, to avoid the use of two operators next to each other.

The statement X = -A+B is correct. The expression R * - S is not correct. Write the expression in its correct form.

Answer: R*(-S)

II.A.24 Previously, only one letter has been used to identify each variable. If we continue in this manner, a program would be restricted to twenty-six variables. To remove this restriction, each variable may be defined with any combination of 1-7 characters (numerals or letters) beginning with a letter.

A variable name must not contain more than _____ characters. The first character of a variable name must always be a _____.

Answer: 7, letter

II.A.25 There is one exception to the above rule. The letter O followed by six numerals is not a valid variable name (to be explained later).

Pick the three incorrect symbols (or identifiers) from the following list: _____, _____, _____

1) AX
2) B1C3
3) CZKR1LPS
4) O342312
5) 12AC

Answer: 3, 4, 5

II.A.26 If you answered II.A.25 correctly, go to II.A.28. Otherwise, continue.

II.A.27 The third symbol is not correct because it has more than seven characters. The fourth begins with the letter O and is followed by six digits. The last begins with a numeral.

II.A.28 It is often useful to make the program identifiers resemble the actual quantity. An illustration is given in the following computation of centripetal force.

```
FORCE = MASS * (VEL * * 2/RADIUS)
```

In the above statement, the numerical value of FORCE becomes _____ if MASS is 16, VEL is 10 and RADIUS is 5.

Answer: 320.0

II.A.29 Up to now, we have only considered individual FORTRAN statements. In practice, many statements are usually required to solve a particular problem. Unless otherwise directed by methods you will learn later, the FORTRAN statements are executed sequentially.

The statements

```
ACCEL = VEL * * 2/RADIUS
FORCE = MASS * ACCEL
```

are equivalent to the one statement in II.A.28. Assuming the same values as in II.A.28, what value is assigned to the variable ACCEL?

Answer: 20.0

II.A.30 The advantage of using two statements is that the acceleration, ACCEL, is available for use in other statements.

If the value of A is 20.0 and the value of B is 5.0, what are the values of A and B <u>after</u> the two statements?

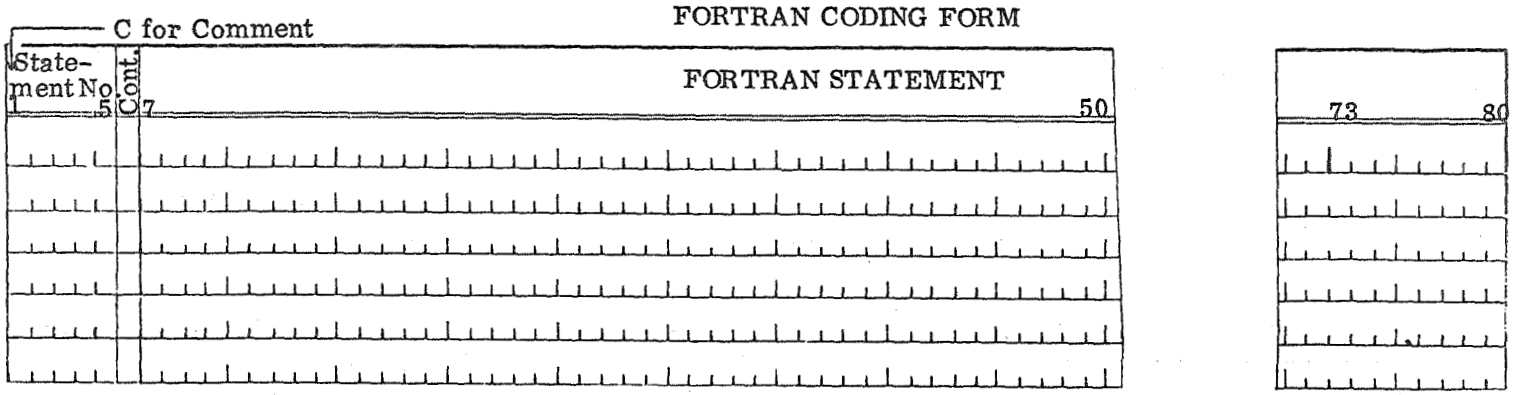
A = A + B _____

B = B + A _____

Answer: 25.0, 30.0

II.A.31 The first statement above assigned a new value to A. This value is 25.0. It is obtained by adding the current values of A and B. The new value of A is then used in the second statement to obtain the new value of B.

II.A.32 You will be provided with FORTRAN coding forms for use in writing your programs. These forms are not all alike but the basic format remains the same. The following illustration shows the format of a typical coding form.



II.A.33 Notice that the columns on the coding form are numbered from 1-80. These columns correspond to the 80 columns on the computer cards which are used to input information to the computer.

There are _____ columns on the FORTRAN coding form.

Answer: 80

II.A.34 After the FORTRAN program is written, the information on the coding forms is keypunched into computer cards. The information in each non-blank line on the coding form is punched into one computer card.

After a program is written, keypunch operators transfer the information from the _____ _____ to computer cards.

Answer: coding forms

II.A.35 Columns 7-72 are used for writing the FORTRAN statements. Normally, a statement will start in column 7 and use as many columns as are needed to complete the statement.

FORTRAN statements must not start before column _____ or extend beyond column _____.

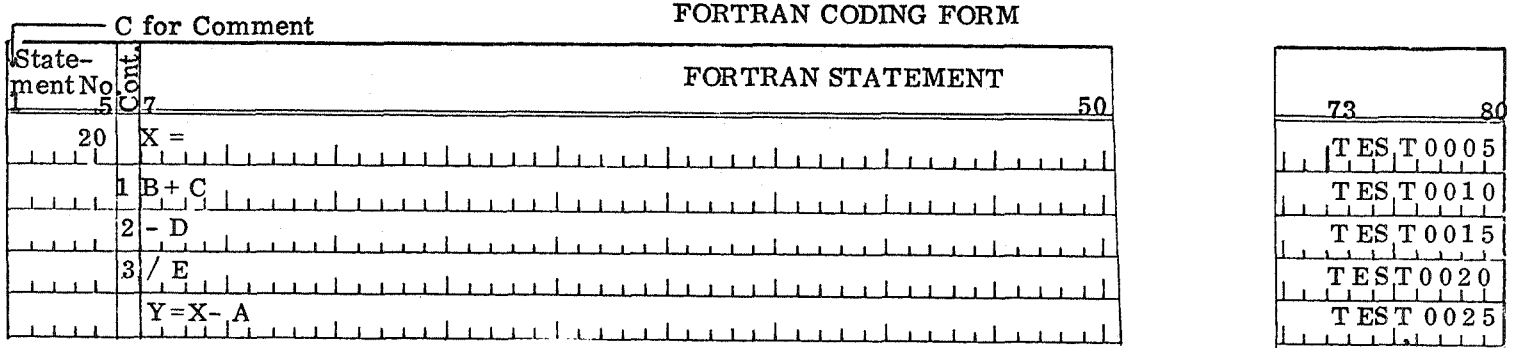
Answer: 7, 72

II.A.36 Suppose we get to column 73 and have not completed the statement. In this case, the next card must be marked as a continuation card and the statement continued in columns 7-72. Any card which has a non-zero punch in column 6 is considered to be a continuation of the previous card.

A limit of 19 continuation cards may be used for a single statement by punching a letter or non-zero numeral in column _____ of each continuation card.

Answer: 6

II.A.37 Continuation cards may be used when necessary on all FORTRAN statements. Statement 20 in the example below is punched on a total of four computer cards.



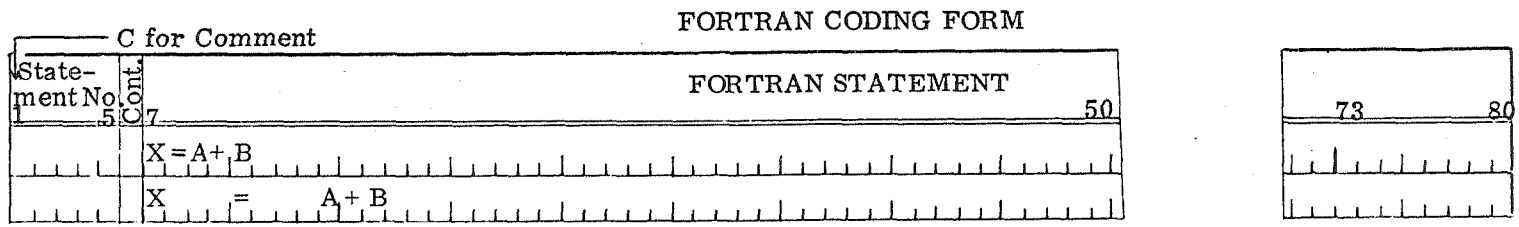
Statement 20 is an exaggerated use of continuation cards. Actually, the computation of X would normally require only _____ card(s).

Answer: one

II.A.38 Except for a special case which will be considered later, all blanks which occur in columns 7-72 are ignored by the compiler.

The following two statements are equivalent. True or false? _____

Answer: True



II.A.39 The two statements in II.A. 38 are equivalent because the blanks which appear in columns 7-72 are ignored by the FORTRAN compiler.

II.A.40 Columns 1-5 may be used to identify a particular statement by assigning it a unique statement number. This statement number can be any integer from 1 through 99999.

The same statement number must not be assigned to more than one statement. True or false? _____

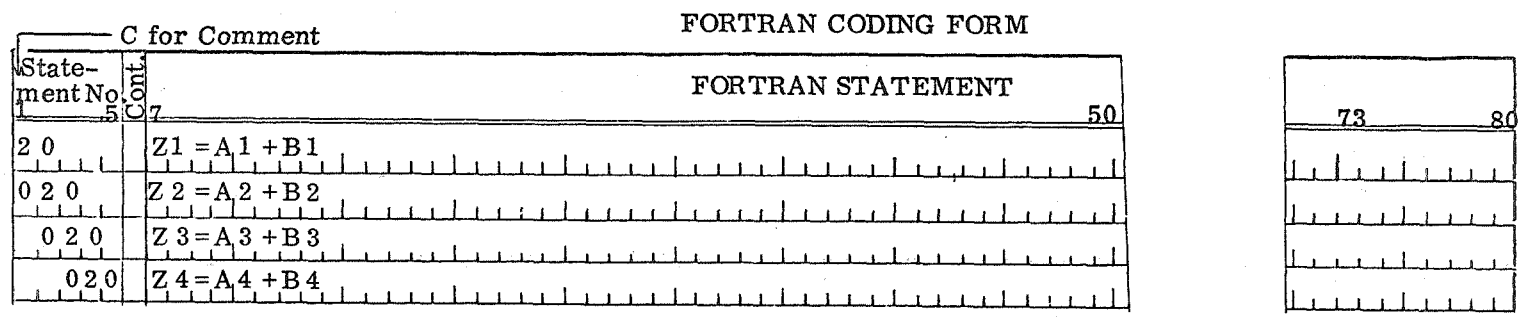
Answer: True

II. A.41 There is no need to assign every statement a number. In fact, only a small percentage of the statements will require statement numbers. The reasons for assigning statement numbers will become apparent later.

Not all statements require statement numbers. True or false? Answer: True

II. A.42 When the statement number is interpreted by the FORTRAN compiler, blanks and leading zeros are ignored.

The following statements all have the same statement number. True or false ? Answer: True



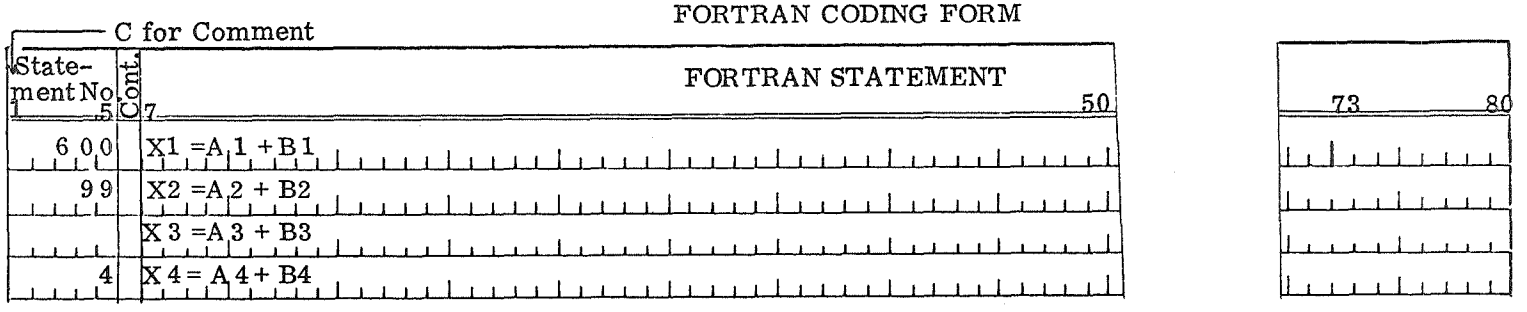
II. A.43 In the last example, we drop all zeros and blanks which occur before finding a non-zero numeral. Consecutive blanks on the right end are also dropped. Therefore, the statement number associated with each statement is 20.

The four statements shown in the last example must not appear in the same computer program. This is true because they all have the same _____ _____.

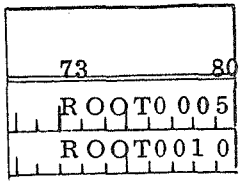
Answer: statement number

II. A.44 Statement numbers may be assigned in any order. Since a statement number is used as an identifier for one particular statement, the value of a statement number is not related to its position in the program.

The example below shows some valid statement number assignments.



II.A.45 Columns 73-80 are not scanned by the FORTRAN compiler. It is suggested that these columns be used to identify your program and to provide sequence numbers in case of dropped cards, etc. For example, the first two statements of a program to calculate the roots of a polynomial might contain the following in columns 73-80.



Numbering by fives or some other increment larger than one provides flexibility for later program changes.

Columns 73-80 are used for punching the FORTRAN statement. True or false? _____

Answer: False

II.A.46 It is often convenient to write comments which describe the computations being performed. These comments may be written on the coding forms by putting the letter C in column 1. These comments will appear on computer listings.

A C in column 1 indicates that the card contains _____.

Answer: comments

II.A.47 Before continuing with section B, work exercise II.A in your workbook.

II.B Data

II.B.1 In the expression X**2, the quantity 2 is used in obtaining the square of X. When the number itself is used in a FORTRAN expression, it is called a constant.

The 2 in the expression X**2 is called a _____. Answer: constant

II.B.2 In the expression 3.0*X+6.0, the values 3.0 and 6.0 are constants. Constants are used in FORTRAN expressions in exactly the same manner as variable names.

In the statement A=20.0*B+4.0, the value of B is multiplied by 20.0 and the result is added to _____. Answer: 4.0

II.B.3 You may have noticed that some of the constants in previous examples were written with decimal points and some were written without decimal points. This was done to illustrate the two principal types of constants. The constant written without the decimal point is called an integer constant. The name describes the set of values which can be represented by this type of constant (integers).

In the FORTRAN statement L = 5+7*J, there are two integer constants. What are they? _____ _____ Answer: 5, 7

II.B.4 The constants used in previous examples which included a decimal point are called real constants. Here again the name suggests the range of values which can be represented by this type of constant (real numbers).

The expression X**2-5.4*C contains both an _____ constant and a _____ constant.

Answer: integer, real

II.B.5 Real constants may be written in several different forms. All of these forms have two common characteristics, however. They are:

1) Every real constant must contain at least one and not more than 15 decimal digits.
2) Every real constant must have a decimal point.

Select the one value which is not classified as a real constant.

1) 45.2
2) .28
3) 36
4) 150.
5) 6.0

Answer: 3)

II.B.6 The answer 3 is correct because the value 36 does not contain a decimal point.

The value 36 is called an _____ constant.

Answer: integer

II. B. 7 Constants used in scientific work are often expressed as a value multiplied by a power of ten. For example, $.34 \times 10^{-6}$ is a shorthand method for writing .00000034. Real constants in FORTRAN may be written in a similar form. The value $.34 \times 10^{-6}$ is written in FORTRAN as .34E-6 or 3.4E-7 or .034E-5 or etc.

Write the actual value of each of the following real constants.

1) .10E2 _____
2) 10.E-2 _____
3) .004E+4 _____

Answer: 10.0
.1
40.0

II. B. 8 Notice that the power of ten is always written as a positive or negative integer. If no sign is indicated, the value is assumed to be positive.

Select the real constant which is not written correctly.

1) .32
2) 40.0E-8
3) 50.0
4) .20E-4.0
5) .0006 _____

Answer: 4)

II.B.9 In number II.B.8, 4), the power of ten multiplier is written as a real constant. This value is restricted to positive or negative integers.

Classify each of the following constants as either real or integer.

1) 24 _____
2) 24.0 _____
3) 256.5 _____
4) 10 _____
5) .03E2 _____
6) 13. _____

Answer: integer
real
real
integer
real
real

II.B.10 Real constants may be zero or any positive or negative value between 10.0E-294 and 10.0E322.

The value 25.0E400 is not a valid real constant because it is too _____.

Answer: large

II.B.11 A variable is a quantity (represented by a symbol) which may have different values assigned to it during execution of the program.

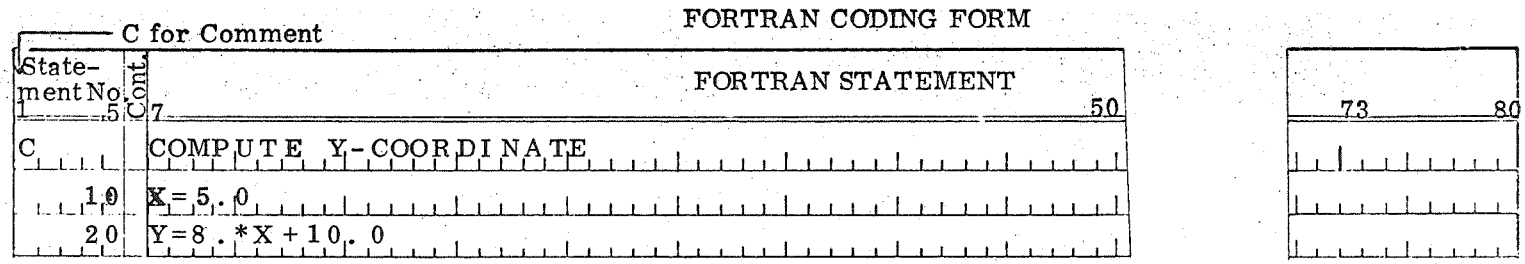
The value of a variable may be changed at any point in the program while the value of a constant always remains fixed. True or false? _____

Answer: True

II.B.12 The value of a variable may be changed as often as necessary to obtain the desired result.

In the following example, the value assigned to the variable X after statement 10 has been executed is _____. The value of Y after statements 10 and 20 have been executed is _____.

Answer: 5.0, 50.0



II.B.13 The two principal types of variables used in FORTRAN statements are integer variables and real variables. Integer variables, like integer constants, can only be assigned integer values. Real variables can be assigned the value zero or any positive or negative value between 10.0E-294 and 10.0E322.

The two principal types of variables are _____ variables and _____ variables.

Answer: integer, real

II. B. 14 Integer constants are distinguished from real constants by the presence or absence of a decimal point. The usual way of distinguishing between integer variables and real variables is by proper selection of the variable name. Variable names beginning with any of the letters I, J, K, L, M, or N are considered to be integer variables.

Classify each of the following variables as either real or integer.

1) RADIUS _____
2) JCOUNT _____
3) ACCEL _____
4) X _____
5) K _____

Answer: real
integer
real
real
integer

II. B. 15 Integer constants and variables are stored in the computer with the binary point assumed to be at the right hand end of the computer word. The mode of operation involving integer constants and variables is called fixed-point or integer arithmetic.

All arithmetic in the statement K = J - L is performed in the _____ mode.

Answer: fixed-point

II.B.16 Real constants and variables permit fractional values and, therefore, must be stored in computer memory along with an indicator which provides information on the position of the binary point. The binary points must be aligned by the computer before the operation can take place. The mode of operation involving real variables and constants is called floating-point arithmetic.

Since most applications require fractional values, calculations are generally performed in the _____ mode.

Answer: floating-point

II.B.17 Remember that fractional values cannot be carried in the fixed-point mode. Therefore, a division of two fixed-point values will result in the loss of any remainder; this is called truncation.

If the value of J is 7 and L is 5, the result of the expression J/L is _____.

Answer: 1

II.B.18 Work exercise II.B in your workbook before starting section II.C.

II.C Mixed-Mode Expressions

II.C.1 In some instances, it is convenient to write expressions which contain a mixture of real and integer quantities. If no exponentiation is involved, the two modes may be mixed without any restrictions. Generally speaking, mixed-mode and fixed-point mode arithmetic require more computer time and memory location than does floating-point mode arithmetic.

Integer and real quantities may be mixed freely in the same expression as long as there is no exponentiation. True or false?_____

Answer: True

II.C.2 Within a pair of parentheses where the mode is constant, all computation within those parentheses will be carried out in the mode of the quantities present.

In the statement X = Z*(4-J), the mode of (4-J) is _____.

Answer: fixed-point

II.C.3 If the mode is mixed and no parentheses are present, the integer quantities are converted to real and the calculations are performed in the floating-point mode.

Suppose each of the following expressions is evaluated. Give the mode of the result of each expression.

1) (L-K) _____
2) (X-3*P) _____
3) Y-L/K _____
4) Y-(L/K) _____

Answer: fixed-point
floating-point
floating-point
floating-point

II. C. 4 Let us examine each expression in II. C. 3 when

K = 4 P = 5.3
L = 7 X = 20.5
Y = 3.35

Number 1 is composed entirely of integer quantities. Therefore, it will be evaluated in the fixed-point mode.

II. C. 3 1.) (L-K) = 3

Numbers 2 and 3 are mixed expressions with no parentheses within the expression and the integers are converted to floating-point form before the expressions are evaluated.

II. C. 3 2.) (X-3*P) = 20.5 - 3. * 5.3
= 20.5 - 15.9 = 4.6

In II. C. 3 3.) notice that both L and K are converted before the division takes place and the result of the division is in floating-point form.

II. C. 3 3.) Y-L/K = 3.35 - 7.0/4.0
= 3.35 - 1.75
= 1.6

In this case L/K is not truncated.

II.C.4 (Cont.)

Number 4 is also mixed mode, but it contains an all fixed-point expression within a set of parentheses. This fixed-point expression is evaluated first giving a fixed-point result. This result is then converted to floating-point form and subtracted from Y. In this case the result of L/K is truncated.

II.C.3 4.) Y-(L/K) = 3.35-(7/4)
= 3.35-(1)
= 3.35-1.0 = 2.35

II.C.5

When an expression contains exponentiation, the following rules determine the type of the result obtained.

1) Integer to an integer power gives an integer result.
2) Real to a real power gives a real result.
3) Real to an integer power gives a real result.
4) Integer to a real power is not allowed.

Indicate the type of result given by each of the following expressions.

1) X**2 _____
2) Y**(J+K) _____
3) L**N _____
4) R**Y _____

Answer: real
real
integer
real

II.C.6 Now let us consider an exponential as part of a larger mixed mode expression. First, determine the mode of the result of the exponentiation. Next, consider this result as a quantity to be used in the larger expression and apply the rules for mixed mode expressions.

One of the following expressions is not valid. Indicate the type of result given by the following valid expressions.

1) L-J**3 _____
2) R-J**B _____
3) K+R**Y _____

Answer: integer
invalid
real

II.C.7 In the previous expressions, the second is not valid since an integer raised to a real power is not allowed. In the remaining two examples, the type of result obtained from the exponentials is determined first. In the first expression, the result is an integer which is then subtracted from an integer. Thus, an integer value is obtained. In expression 3, the result of the exponential is a real quantity. The expression is now mixed since it contains a real quantity and an integer quantity. Therefore, the integer is converted to real form and the addition is performed in the floating-point mode yielding a real result.

II. C. 8 In a FORTRAN arithmetic statement, the evaluation of the expression on the right hand side of the = results in a single value. The type of this result depends on the operations involved and the types of the different variables which appear in the expression. If the variable on the left of the = is not the same type as the result of the expression on the right of the = , the result of the expression is converted to the type of the variable on the left.

For example, I = X+Y will convert the result of the expression X+Y to integer before assigning the value to I. This results in the loss of any fractional part. The conversion (if needed) across the = is always to the type of the variable on the _____ of the = .

Answer: left

II. C. 9 Work exercise II. C in your workbook.

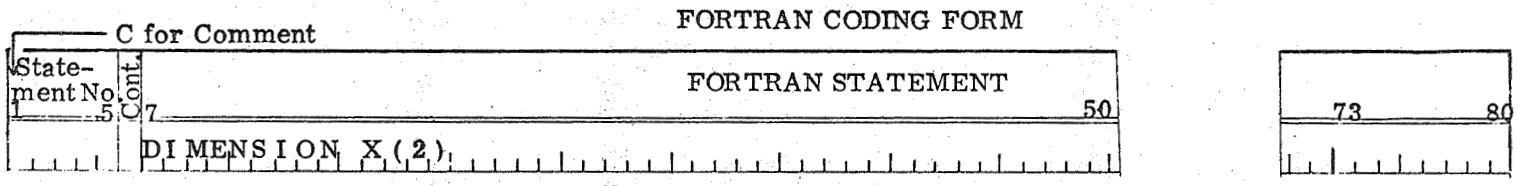
II.D Arrays and Subscripted Variables

II.D.1 Sometimes it is necessary to have several values of one variable available for computation. For example, values from a table of hourly temperatures over a 24 hour period are referenced as T (1), T (2), T(3), . . ., T(24). The variable T in this case is called a subscripted variable.

A subscripted variable provides a means for associating several values with a single variable. True or false? _____

Answer: True

II.D.2 A variable is defined as a subscripted variable by use of a DIMENSION statement. The following statement defines the variable X as a subscripted variable.



A variable which appears in a DIMENSION statement is called a _____ variable.

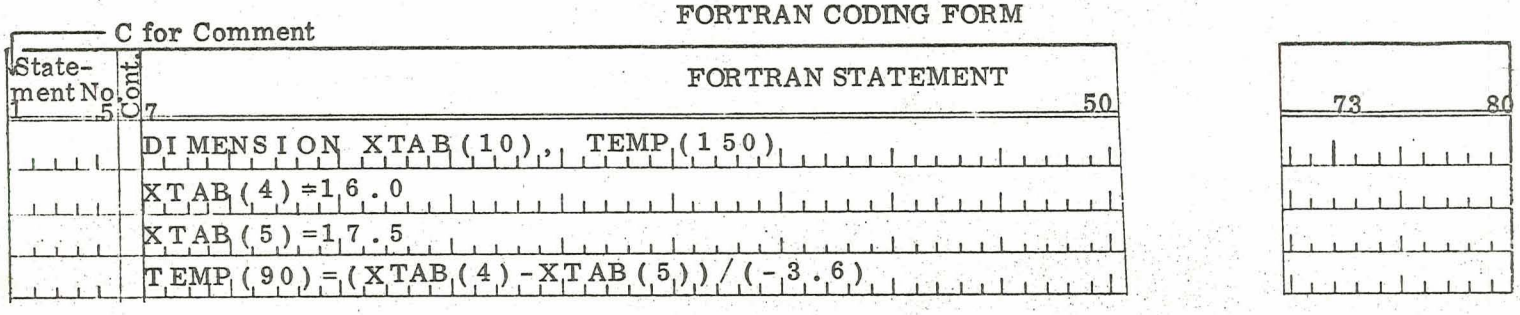
Answer: subscripted

II.D.3 The value 2 in the statement DIMENSION X(2) indicates that the data array X may contain a maximum of two values. The maximum values of subscripts which appear in DIMENSION statements are restricted to integer constants except in one situation which will be discussed later.

The statement DIMENSION A(20), Y(300) defines A and Y as subscripted variables and reserves computer memory for up to _____ values of A and _____ values of Y.

Answer: 20, 300

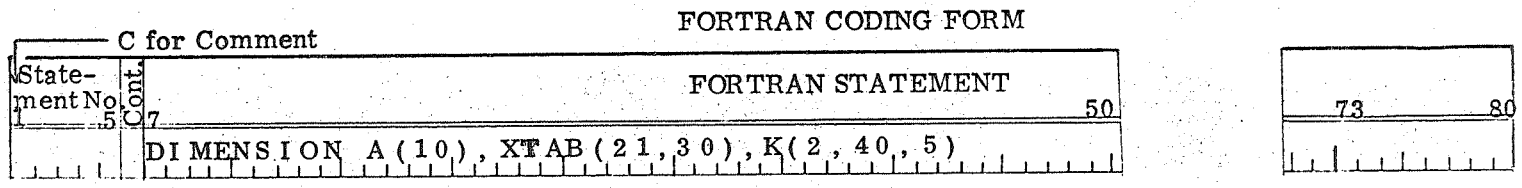
II.D.4 The value of the subscript determines which quantity in the data array is referenced. For example, the following statements store values into the fourth and fifth locations of the array XTAB and into location ninety of TEMP.



The type of a subscripted variable is determined in the same manner as the type of a simple variable. What is the type of the subscripted variable TEMP? _____

Answer: Real

II.D.5 In the previous examples, only subscripted variables with one subscript have been shown. It is also permissible to define variables with two and three subscripts. The following DIMENSION statement defines variables with one, two, and three subscripts.



A subscripted variable may be defined with one, two or three subscripts. True or false? _____

Answer: True

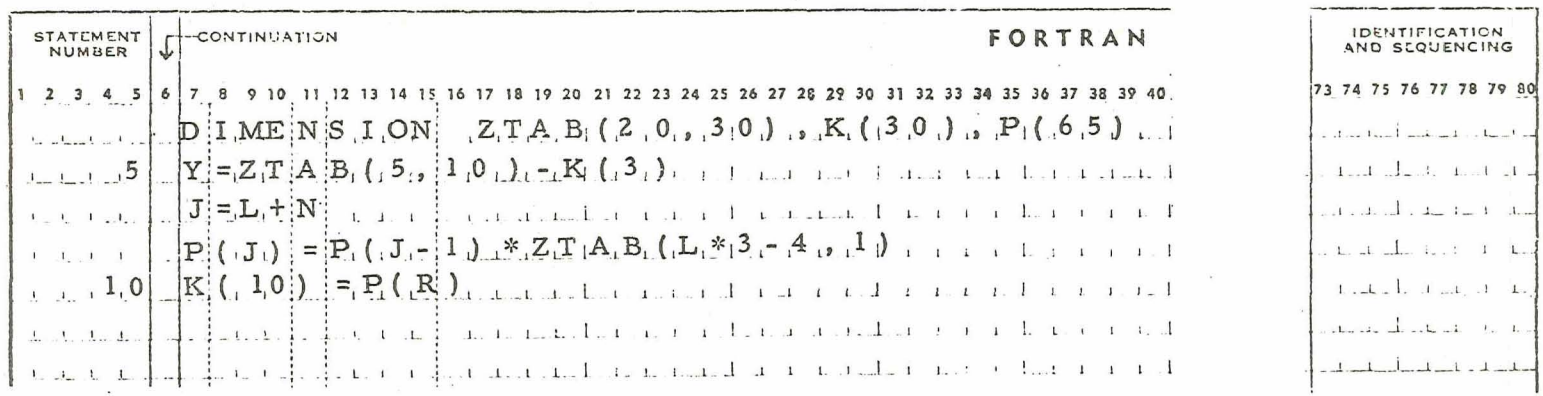
II.D.6 A data array defined with one subscript is referred to as a one-dimensional array. The terms two and three-dimensional arrays are used to refer to variables with two and three subscripts, respectively.

A subscripted variable with two subscripts is referred to as a _____ dimensional array.

Answer: two-

II. D. 7 The value of a particular subscript is called an index. The index must be an integer constant, an integer variable, or the integer result of simple arithmetic operations.

Pick the one statement with an invalid subscript.



Answer: Statement 10

II. D. 8 Let us examine the FORTRAN statements used in the last question. First, the DIMENSION statement defines three arrays. The first is two-dimensional and the second and third are one-dimensional. Statement 5 computes Y using values from ZTAB and K. J is then computed using existing values of L and N. The next statement illustrates the use of integer variables and simple arithmetic expression as indices. Statement 10 is not valid because a real variable cannot be used as an index.

II.D.9 In the statement, DIMENSION ZTAB (20, 30), the values 20 and 30 represent the maximum values of the first and second subscript respectively. This statement causes the reservation of enough computer memory for a maximum of 600 values of the variable ZTAB. The maximum number of values is always the product of the maximum values of the subscripts.

Determine the maximum number of values which may be assigned to each of the variables in the following DIMENSION statement.

DIMENSION X(3, 30,4) , Y(20, 60), K(2,3), P(10)

Answer: 360, 1200, 6, 10

II.D.10 If a subscripted variable is referenced as a simple variable, all subscripts will be assigned the value 1. For example, the expression Y*ZTAB+B will be interpreted as Y*ZTAB(1, 1) +B(1) if B and ZTAB have been defined as one- and two-dimensional variables, respectively.

In general, an index of 1 will be assumed for all subscripts missing on the right. KTAP(I) implies KTAP (I, 1, 1) if KTAP is a _____ dimensional array.

Answer: three-

II.D.11 Two-dimensional arrays are often thought of as a rectangular array where the first subscript represents the row number and the second subscript represents the column number.

If AMAT is a two-dimensional array, write the expression which multiplies the value in row 6 column 8 by the value in row 1 column 3.

Answer: AMAT(6, 8)*AMAT(1, 3)

II.D.12 As a review of subscripted variables, fill in the following blanks.

A subscripted variable must be defined by the use of a _____ statement. This DIMENSION statement may define several subscripted variables if they are separated by _____. As in all FORTRAN statements, continuation cards may be used if necessary. If X has been defined as a one-dimensional array, the use of the variable X without a subscript in an arithmetic statement will always reference _____. The expression X(5)*X(6) will multiply the fifth and _____ values of the array X. Of course, we have assumed here that the array was defined with at least _____ values. If AMAT has been defined as a two-dimensional array with 20 rows and 40 columns, the maximum number of values which can be assigned to AMAT is 20 X 40. The row index must never exceed _____ and the column index must never exceed 40.

Answer: DIMENSION
commas
X (1)
sixth
six
20

II.D.13 The reference manual* gives information on how data arrays are stored in computer memory. This information is not necessary for writing FORTRAN programs if the programmer remembers that he must use the same indices to reference a value as were used in storing the value in the array. This means that if a value is stored in AMAT (6, 10), then any reference to the value must have the same indices. The indices are not required to have the same form, however, The reference could be X*AMAT(I, J) as long as the value of I is 6 and J is 10.

II.D.14 **Work exercises II.D in your workbook.**

* CONTROL DATA FORTRAN REFERENCE MANUAL

II.E Data Types

II.E.1 The five data types to be discussed are:

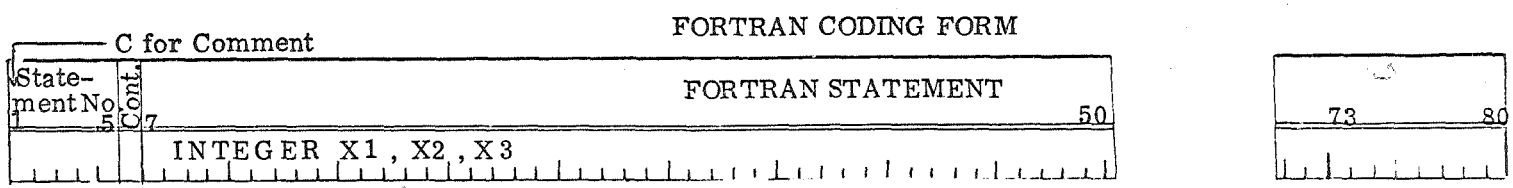
integer
real
complex
double precision
logical

II. E. 2 A provision is made which permits the type of a specific variable to be declared in the program. This is accomplished by the use of a type declaration statement. The type declaration must appear before the variable is used.

A variable named COUNT is normally a _____ variable. It may be declared an integer by use of a type declaration statement.

Answer: real

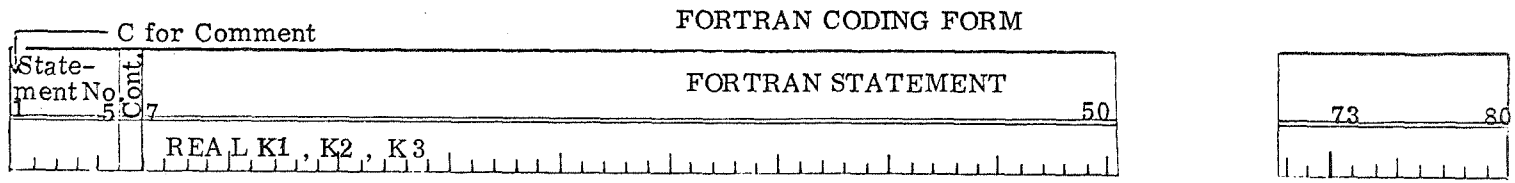
II. E. 3 The type declaration statement consists of the type followed by the variables being declared as that type. The variable names are separated by commas. The following statement will cause X1, X2, and X3 to be considered integer variables.



An integer variable does not need to appear in a type statement if the variable name begins with one of the letters _____, _____, K, L, M or _____.

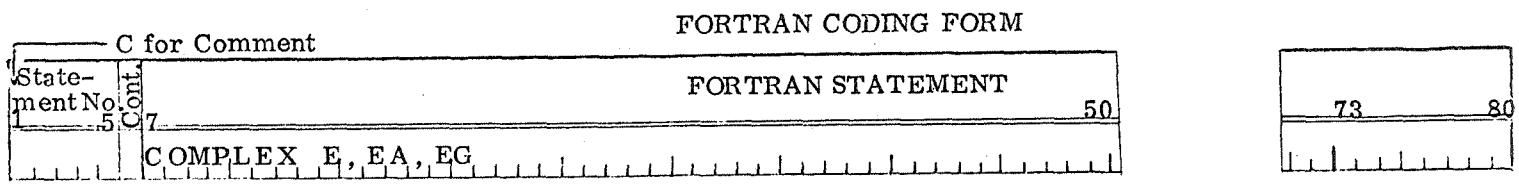
Answer: I, J, N

II. E. 4 The following statement will cause K1, K2, and K3 to be considered real variables.



COUNT is a real variable only if it is declared a real variable by use of a type statement.
True or false?_____ Answer: False

II. E. 5 All variables which are complex must appear in a type statement. Here, the word complex has the same meaning as complex numbers used in mathematics. The following statement defines E, EA, and EG as complex variables.



II.E.6 Each complex value has two parts, a real part and an imaginary part. Thus, a complex constant is composed of two parts. These two parts are separated by a comma and enclosed in parentheses. Each part must be a valid real constant. The following statements illustrate the use of complex constants and variables.

C for Comment

FORTRAN CODING FORM

Statement No. 1–5	Cont.	FORTRAN STATEMENT 7–50	73–80
		DIMENSION A(4),XTAB(20,30)	
		COMPLEX A, C, XTAB	
		A(3)=(5.0,2.5)	
		A(4)=(24.0E02,.234)	
10		XTAB(N-1,3*I)=A(3)*A(4)+(3.4,5.2)	
		C=XTAB(1,1)	

In the first statement, A and XTAB are defined as _____ variables. The variable A may be assigned a maximum of _____ values. This means four complex values since A is defined as _____ in the second statement. Remember that the real and imaginary part constitutes one complex value. The arithmetic operations performed in statement number 10 are carried out according to the usual rules of complex arithmetic. The values in each part of complex constants must conform to the rules for _____ constants.

Answer: subscripted
4
complex
real

II. E. 7 Variables and constants can also be defined in the DOUBLE PRECISION mode. This type of variable is used only when accuracy requirements demand that more than fifteen significant decimal digits be carried during arithmetic operations.

Just like complex variables, all double precision variables must be declared in a _____ statement.

Answer: type

II. E. 8 Double precision variables must be declared in type statements using either DOUBLE PRECISION or simply DOUBLE.

For example,

```
DOUBLE PRECISION A, R, Q
DOUBLE X, Y
```

will make the variables A, X, _____, _____, and _____ double precision.

Answer: R, Q, Y

II. E. 9 Double precision arithmetic requires that two computer memory locations must be used for each value. One contains the most significant part of the word and the other contains the least significant part. Special procedures have been developed to carry out arithmetic on values in this form. This does not concern the programmer except that he should remember that double precision arithmetic is considerably more complicated and, therefore, takes much more computer time.

For the 20 digit number 98989898981212121212 the most significant half of the number is $9898989898 \times 10^{10}$ and the least significant half of the number is 1212121212.

Double precision and complex arithmetic is very valuable when needed. It should be remembered, however, that execution time on the computer is much _____ in these modes.

Answer: greater

II. E. 10 A double precision constant differs from a real constant in that every double precision constant must be written with a power of ten multiplier. The power of ten in this case is denoted by a D instead of E which is used for real constants. Up to 29 digits may be used in writing double precision constants.

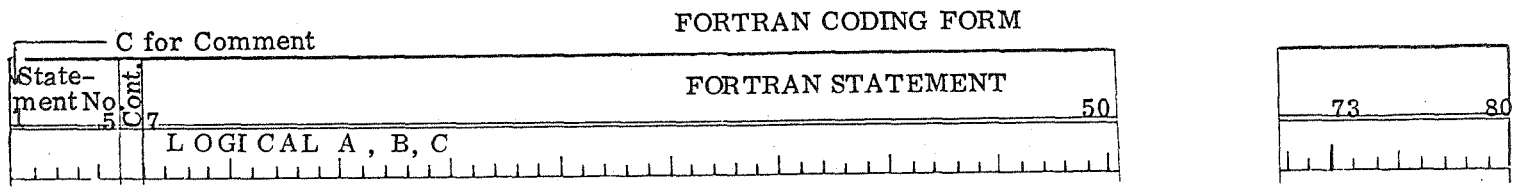
Classify each of the following constants as integer, real, or double precision.

a.) 10 _____
b.) 20.0 _____
c.) 21.00E-3 _____
d.) 266.0D-2 _____
e.) .075D3 _____

Answer: a.) integer
b.) real
c.) real
d.) double precision
e.) double precision

II. E.11 The last type of variable to be discussed is the logical variable. This variable must also be declared in a type statement.

The following statement will cause A, B, and C to be considered logical variables.



All logical variables must be declared in a _____ statement.

Answer: type

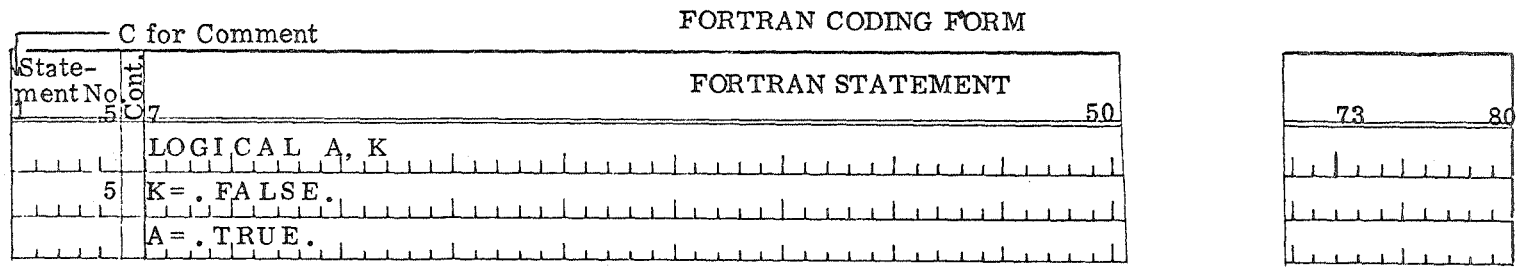
II. E.12 A logical variable is similar to an on-off switch. It is either zero or non-zero. If it is non-zero, it is said to have the value .TRUE. . If it is zero, it is said to have the value .FALSE. .

A logical variable can have the value _____ or the value _____.

Answer: .TRUE.
.FALSE.

II. E.13 In previous examples, we have always considered FORTRAN statements executed in sequence. In PART III of this manual you will see how to change the order of execution based on the results of certain computation. For example, the roots of a quadratic must be computed in one way if $b^2 - 4ac$ is positive and a different way if it is negative. If a logical variable is set to .TRUE. in one case and .FALSE. in the other, it may be interrogated later in the program to determine which computation was made.

II. E.14 Since there are only two possible values which a logical variable can assume, there are only two logical constants. They are .TRUE. and .FALSE. . The three statements below illustrate the use of logical constants.



Statement 5 will cause the variable K to be set to _____ . The next statement will set A to a non-zero value. Specifically, each binary digit will be set to a 1.

Answer: zero

II.E.15 The two logical constants may be shortened to .T. and .F. for convenience.

Logical constants may be written as .T. and .F. instead of _____ and _____.

Answer: .TRUE. , .FALSE.

II.E.16 This completes the definitions of the five variable types. Two more constants remain to be discussed, however. The first of these is the octal constant. This constant permits a number written in the octal system to be used in FORTRAN statements. This constant is generally used when a particular set of binary digits is required.

Since each octal digit converts to three binary digits, the sixty bits in a computer word is equivalent to _____ octal digits.

Answer: 20

II.E.17 An octal constant is right adjusted in the computer word. This means that if fewer than twenty octal digits are specified, then the unused binary digits on the left are set to zero.

If an octal constant contains ten digits, it will be converted to thirty binary digits which will occupy the right hand half of the computer word. The left hand half will be set to _____.

Answer: zero

II. E. 18 The first method which can be used to write an octal constant is the letter O followed by at least six octal digits and not more than twenty octal digits. The second method to write an octal constant is to follow a set of octal digits by the letter B. Here again, the number of octal digits must not exceed twenty.

One of the following is not an octal constant. Indicate which and tell why. _____

1) O027431
2) 25B
3) O653

Answer: O653 is a variable name

II. E. 19 The last constant to be defined is the Hollerith constant. This constant provides a means for converting characters (letters, numbers, and special characters) to the 6000 series console display code. Each character is represented in the computer by a unique set of six binary digits. Thus, ten characters may be stored in each computer word.

Each character is represented in the computer by a unique set of six binary digits. The word THE requires _____ binary digits.

Answer: 18

II. E. 20 The Hollerith constant is written with the number of characters, the letter H, then the characters to be converted to display code. For example,

7HTESTING

will convert TESTING to display code.

The 7 in 7HTESTING indicates the _____ of characters to be converted.

Answer: number

II. E. 21 A blank is considered a special character and has its own display code. Therefore, blanks within the count specified for a Hollerith constant are not ignored.

Indicate the proper character count for the following Hollerith constant.

___HAT_THE_ZOO

Answer: 10

II. E. 22 When less than ten characters are used, the Hollerith constant is left adjusted and blank characters are added on the right end.

If the character count is six, how many blank characters are added to the right end of the Hollerith constant? _____

Answer: 4

II. E. 23 Two variations may be used in writing Hollerith constants. These variations replace the H with the letter L or R. When ten characters are used, the same result will be obtained by using H, L, or R. If less than ten characters are used, the use of the letter L will cause the characters to be left adjusted in the computer word. The use of the letter R will cause the characters to be right adjusted in the computer word. In both cases the unused portion of the computer word is filled with binary zeros instead of blank character codes.

II. E. 23 (Cont.)

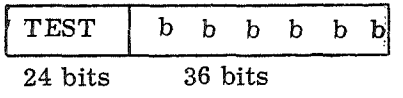
Now let us compare the three statements

a.) A = 4HTEST
b.) A = 4LTEST
c.) A = 4RTEST

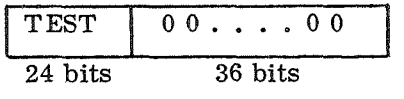
Their differences can best be illustrated by examining the contents of the computer location which contains the variable A after the execution of each instruction. Let b denote the character blank.

computer location A

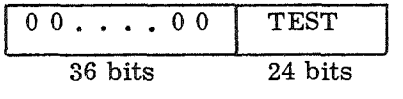
a.) A = 4HTEST



b.) A = 4LTEST



c.) A = 4RTEST



The use of the letter H causes the unused portion of the computer word to be filled with _____ characters. The use of the letter L causes binary _____ to be used for the same purpose.

Answer: blank
zeros

II. E. 24

The next section will give additional information on some uses of the octal and Hollerith constants. At this point work exercise II. E in your workbook.

II. F Logical Operations

II. F. 1 Previously we considered the different arithmetic operations. In this section the three logical operations will be discussed. The arithmetic operations were based on the operations of basic arithmetic. The logical operations are based on Boolean algebra.

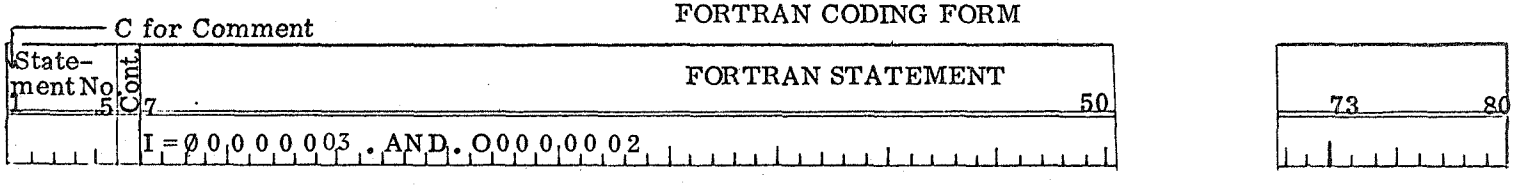
II. F. 2 The three logical operations are .AND. , .OR. , and .NOT. . Logical operations can be performed on variables, constants, or expressions of any mode.

Logical operations may be performed on octal constants. True or false? _____

Answer: True

II. F. 3 The result of a logical operation is considered to be in the logical mode. This result may be used in mixed mode expressions or assigned to a variable of any type.

Consider the FORTRAN statement

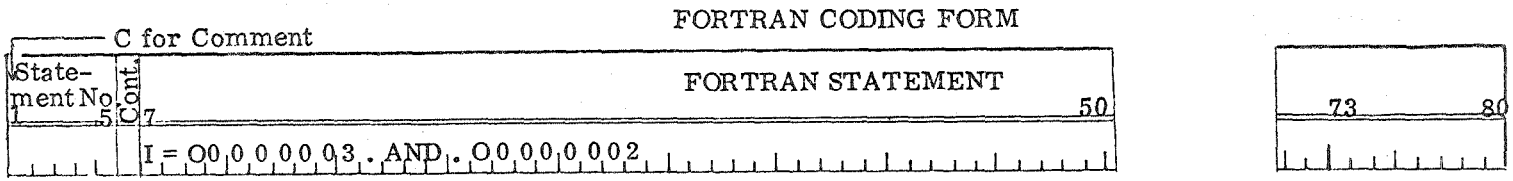


The value on the right side of the equal is the result of a logical operation and, therefore, is in the _____ mode. When this result is assigned to I, it becomes an integer value, but no_____ occurs across the equal.

Answer: logical, conversion

II. F. 4 The logical operations are performed on the binary digits. In the case of the .AND. operator, each binary digit of one value is compared with the corresponding binary digit of the other value. If the binary digit of the first value and the corresponding binary digit of the second value are both 1, the corresponding binary digit of the result is a 1. All other binary digits in the result are 0.

Consider the FORTRAN statement



What is the value of I after execution of the statement? _____

Answer: 2

II. F. 5 The previous result was obtained by examining the binary digits in the two values.

$$\text{O0000003} = 00\cdots000011_2$$

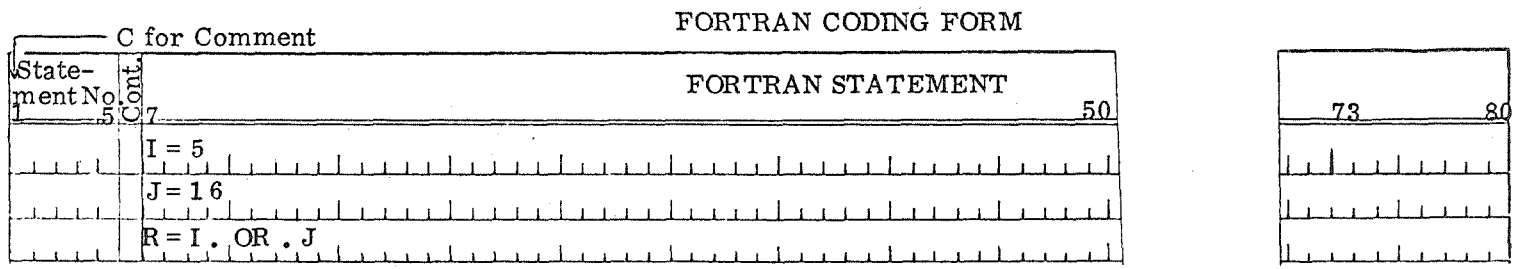
$$\text{O0000002} = 00\cdots000010_2$$

$$\text{I} = 00\cdots000010_2$$

The second binary digit counting from the right end of the value is the only position which has a 1 in both values. This, by definition, gives a 1 digit in the result. I is then set to this value without conversion and $010_2 = 2$.

II. F. 6 The logical .OR. compares corresponding binary digits and whenever a 1 is present in either operand the result is a 1.

Consider the statements



What is the value of R in octal after the execution of three statements? _____

Answer: 25_8

II. F. 7 The previous answer is determined by looking at the binary digits of the values involved.

$$I = 00\cdots000101_2$$

$$J = 00\cdots010000_2$$

$$R = 00\cdots010101_2$$

Since the result of the .OR. operation is in the logical mode, no conversion is made across the equal.

II. F. 8 The operator .NOT. operates on a single value. The result of this operation is a value which contains ones where the operand contained zeros and zeros where the operand contained ones.

Consider the statements

FORTRAN CODING FORM

C for Comment

State-ment No. 1–5	Cont. 7	FORTRAN STATEMENT 50	73–80
		LOGICAL I,A	
		I=(.NOT.A)	

If A has the value .TRUE. , what is the value of I after the execution of the statement? _____

Answer: .FALSE.

II. F. 9 In the previous example, A is .TRUE.. This means that all binary digits in A are equal to 1. Since (.NOT. A) changes all binary digits in A, the result is zero. Since I is a logical variable, a zero value is considered to be the value .FALSE. .

II. F. 10 Work exercises II. F in your workbook.

II. G More on Mixed Mode Expressions

II. G. 1 Mixed mode expressions were presented earlier for real and integer variables and constants. Now we must expand this concept to include the additional variables and constants which have been discussed since that point.

II. G. 2 Suppose the expression within the innermost set of parentheses is being evaluated. If all variables and constants are of the same type within these parentheses, no conversions are necessary and the result is of the same type.

In an earlier section, you learned that the mixture of real and integer types in an arithmetic expression gave a result in the _____ mode. Answer: real

II. G. 3 If the variables are not of the same type in an arithmetic expression, the order of dominance is:

1) complex
2) double precision
3) real
4) integer
5) logical

The result of an arithmetic expression will be the type of the most dominant variable present in the expression.

The result of the expression (J - K) is _____. Answer: integer

II. G. 4 Except for the logical type, the general rule is that the values are converted to the most dominant type present and the expression is evaluated in that mode. No conversion is performed on logical variables and constants, but the evaluation is still performed in the mode of the most dominant type present.

Consider the expression (A + K) . If A is double precision and K is integer, K is converted to _____ _____ and the addition is performed in the double precision mode.

Answer: double precision

II. G. 5 In mixed expressions, exponentials should be considered as a separate expression to be evaluated first. The result of the exponential can then be considered as a value within the original expression. The following table shows the type of the result of A**B for A and B of different types.

Type of B

Type of A

	C	D.P.	R	I	L
C	N	N	N	C	N
D.P.	N	D	D	D	N
R	N	D	R	R	N
I	N	N	N	I	N
L	N	N	N	I	N

Type of A**B

N means not allowed

From the above table, what is the type of the result when a complex value is raised to an integer power? _____

Answer: Complex

II.G.6 Work exercise II.G in your workbook.

II.H Exercise II.H in your workbook is a review of PART II. Work exercise II.H at this time

PART III

III.A Introduction

III.A.1 In the preceding section you learned to refer to constants, variables, and expressions in FORTRAN notation and to use these quantities in mathematical computations. The writing of FORTRAN statements in the proper order describes a mathematical problem which may be solved by a computer.

You learned that the computer executes statements in the order which they are written. The statements

```
FORCE = XMASS * ACCEL
DIS = 0.5 * ACCEL * TIME **2
```

cause the computer to calculate FORCE first, and then DIS. Furthermore, as this stands, the computer will execute each of these statements just once for each time you write it.

III.A.2 Now you will learn how the use of control statements will enable you to tell the computer to go to and execute certain statements out of the normal order, and whether to do the operation more than once for each time you have written the statement. You will be able to tell the computer how many times to loop back and repeat this operation; or you may choose to instruct the computer to decide how many times to repeat this loop, based on signposts (values) you will provide, and then to give you a record of action taken by means of variables you can interrogate.

Further, when you interrogate any variables, you will find the answer in a format you have specified: either a logical true/false, or a computed value within a range of specified values. You will have already written statements telling the computer what to do next based on these values, or whether to continue or to stop because your program has come to the end of the statements you have written.

III.A.3 Based on the foregoing, it can be seen that control statements give the programmer a powerful tool that frees him from unnecessary extra writing and speeds the work of the computer by providing it with advance instructions as to what to do in all predictable conditions.

If a FORTRAN program logic uses control statements properly, the programmer needs to do _____(more/less) writing.

Answer: less

III.A.4 Statement numbers identify FORTRAN statements to which control statements must refer, but are not needed on other FORTRAN statements. This is why we read in Section II.A.41 that only a small percentage of statements need numbers. (Actually, as will be seen later under the IF statement, sometimes the IF statement may send the computer to another statement merely by its position relative to the control statement without referring to the statement identification; but, otherwise, statements to which the computer is <u>referred</u> must be labelled by a statement number.)

When a control statement refers to another statement, that statement must be labelled with a _____ _____.

Answer: statement number

III.A.5 In the following example, repeated from Section II.A.44, some statements carry statement numbers in columns 1-5, and one is unnumbered.

FORTRAN CODING FORM

C for Comment

State-ment No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
600		X1 = A1 + B1	
99		X2 = A2 + B2	
		X3 = A3 + B3	
4		X4 = A4 + B4	

You would expect to find a control statement or statements elsewhere in this same program referring to which of these statements: 600, 99, unnumbered, or 4?

Answer: 600, 99, and 4

If you later, after writing the statements in III.A.5 above, wished to have a subsequent control statement refer to the third (unnumbered) statement in the example, you could label this one also, with any statement number not already used in the program. True or false?

Answer: True

III.A.6 Continue with section III.B at this time.

III.B GO TO Statements

III.B.1 The first control statement we shall study is the GO TO statement. The GO TO statement does exactly what its name implies. It tells the computer <u>not</u> to execute the next statement in the normal sequence, but rather to "go to" some numbered executable statement elsewhere and continue executing statements in the normal manner from there.

In the example:

```
      GO TO 50
30    I = 1
50    I = 2
```

what is the statement that will be executed next after "GO TO 50"? _____

Answer: I = 2

III.B.2 You have seen <u>executable</u> statements before. The arithmetic statements are a good example, such as A = B + C, because this causes the computer to do, or execute, something. An executable statement cannot be one that merely defines, such as DIMENSION and the type statements you used in Section II, and the COMMON statement you will learn about later.

I = 2 is an executable statement. True or false? _____

Answer: True

III.B.3 Which of the following sample statements could not have control transferred to it by a statement "GO TO 50"? Why?

1.) 50 A = B + C _____
2.) 50 DIMENSION A(4) _____
3.) A(4) = (24.0E02, .234) _____
4.) 50 LOGICAL _____
5.) 50 A = A.AND.B _____

Answer:

1.) OK
2.) DIMENSION statement
3.) statement is not numbered
4.) type declaration
5.) OK

III.B.4 The simplest form of the GO TO statement is called the unconditional GO TO, which has as an operand a statement number to which control should be transferred next. For example, the statement "GO TO 100" will cause statement 100 to be executed next.

What is the value of X as computed by the following statements? _____

Answer: X = 1.0

FORTRAN CODING FORM

C for Comment

Statement No. 1–5	Cont.	FORTRAN STATEMENT 7–50	73–80
50		A=3.0	
633		X=1.0	
		GO TO 100	
14		X=X+A	
100		Y=A**3	

If your answer is correct, skip to III.B.6

III.B.5 The example above would be executed by the computer as follows: Statement 50 would be performed first, setting variable A to 3.0. Next, statement number 633 would set X equal to 1.0. The next statement, GO TO 100, tells the computer to execute statement 100 rather than statement 14. Statement 14 is not executed at this time. Notice that at the completion of executing statement 100, X has not been changed from its original value of 1.0.

If the "GO TO 100" statement has not intervened, statement number 14 would have been executed, and the new value of X would be _____

Answer: X+A or 4.0

III.B.6 The GO TO statement in the example above is called an unconditional GO TO because it does not give the program any option.

The computer unconditionally executes the statement explicitly named in the GO TO statement. True or false? _____

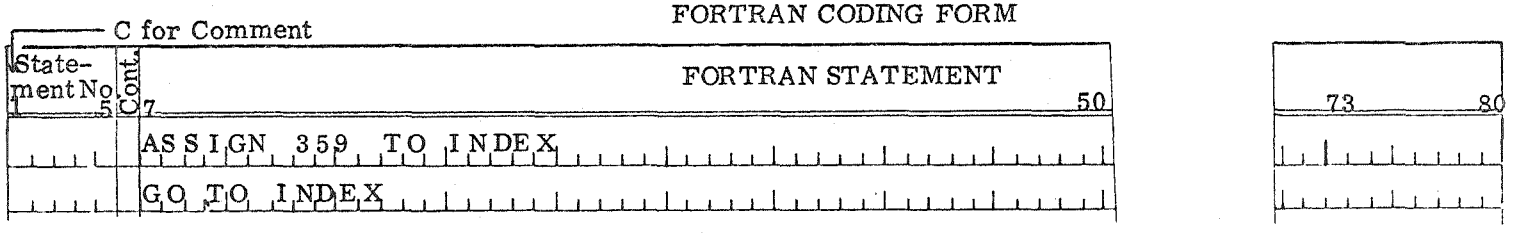
Answer: True

III.B.7 However, if the programmer wishes a GO TO statement that provides a choice of statements to which to branch, he may use two other kinds of GO TO statements. These are called the assigned GO TO statement and the computed GO TO statement. The next few sections will describe their differences in format and meaning.

The two kinds of GO TO statements that allow the programmer a choice of where to branch are called the _____ GO TO and the _____ GO TO.

Answer: assigned, computed

III.B.8 In the assigned GO TO statement, the programmer assigns a statement number to a control variable and then uses the control variable in the GO TO statement. In the example below, the assigned GO TO statement would cause transfer to statement 359.



If the programmer wished to transfer control to statement 25, he would write the first line above as: _____

Answer: ASSIGN 25 TO INDEX

III.B.9 An alternate way of using the assigned GO TO statement is to follow the control variable with a comma, then by a pair of parentheses enclosing a list of statement numbers separated by commas. This list includes all the statement numbers which the programmer assigns to the control variable within the program.

The following example illustrates this form.

III.B.9
(Cont.)

FORTRAN CODING FORM

C for Comment

Statement No. 1–5	Cont. 7	FORTRAN STATEMENT 7–50	73–80
		ASSIGN 359 TO INDEX	
10		GO TO INDEX, (15, 20, 359)	
5		ASSIGN 20 TO INDEX	
		GO TO 10	
15		X=E * P	
		GO TO 30	
359		Y=A * B + 1.5	
		GO TO 30	
20		Z=C / D - 1.5	
30		(program continues)	

In this example, where the second line "GO TO INDEX", is followed by a parenthetical list of statement numbers, a comma after the INDEX may be used. True or false? _____

Answer: True

In a program, the ASSIGN statement may assign a new statement number to the variable (such as INDEX) as many times as the programmer desires, as long as the new statement number assigned is already listed within the parentheses in the assigned GO TO statement. True or false? _____

Answer: True

III.B.10 Notice that in the example in III.B.9 the variable INDEX is an integer variable (the name starts with one of the letters I, J, K, L, M, or N). Remember that only an integer variable may be used with the assigned GO TO statement.

Is the following statement valid? _____

ASSIGN 1127 TO COUNT

Answer: No. The control variable must be in integer mode.

Would changing the name of the control variable COUNT above to ICOUNT make it an acceptable integer variable? _____

Answer: Yes

III.B.11 The only correct way to use the assigned GO TO statement is to use the ASSIGN statement to put a statement number into the variable. This tells the compiler to get the address in computer memory of the statement number and to assign that address to your variable. Do not try to give a variable the desired branching statement number in the form of a computation for use with the assigned GO TO statement.

Which of these examples of the assigned GO TO statement is incorrect?

1.) ASSIGN 1127 TO ICOUNT
GO TO ICOUNT

2.) ICOUNT = 1127
GO TO ICOUNT

_____

Answer: 2.)

III. B. 12 The computed GO TO differs from the assigned GO TO in that it relies on the value of an integer variable to select the proper statement number from a list of statement numbers enclosed in parentheses. The form of the computed GO TO is as follows:

GO TO (20, 30, 100), INDEX

III. B. 13 In the computed GO TO, the integer variable must be non-zero, and no larger than the number of statement numbers in the list. The value of this variable is set by the program by ordinary arithmetic statements and is used as a counter for selecting a statement number according to its order in the parenthetical list--first, second, third, etc.

If N = 3, what will be the next statement to be executed?_____

GO TO (20, 3, 10, 100, 4, 37), N

Answer: 10, the third statement number within the parentheses

If N = 1, what will be the next statement to be executed?_____

Answer: 20, the first number

III.B.14 In the following example, notice that the value of the variable N is preset, or initialized, in the first statement, N = 6. Then later this value is computed, reducing it, or decrementing it, by one, each time statement 30 is executed.

C for Comment

FORTRAN CODING FORM

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
		N=6	
15		GO TO (20,3,10,100,4,37),N	
37		A= B * C	
30		N= N-1	
		GO TO 15	
4		A= B/C	
		GO TO 30	
100		A= B-C	
		GO TO 30	
10		A= B + C	
200		program continues ----	

In the coding above, the value of N is not assigned but _____.

Answer: computed

III.B.15 In the preceding example, you were using a simple kind of loop, in which the various statements listed in the GO TO statement, after performing some operation, send the program back to statement 30 and to the line that follows it:

```
30  N = N - 1
    GO TO 15
```

these in turn return control to the GO TO statement, where the newly computed value of N selects another statement number out of the list. This looping will continue, statement by statement, until some exit is provided. In this case, when statement 10 is reached, it simply does not return the program to statement 30, but merely drops down to the next sequential statement written below 10.

Refer to the sample coding in III.B.14, and answer the following questions:

1.) Starting at the beginning, where N = 6, the GO TO statement will send the program next to what statement?_____

Answer: 37, the sixth number in the list within the parentheses.

2.) After statement 37 is executed, the value of A is_____, and the next statement to be executed will be_____.

Answer: B*C
30

3.) After statement 30 is executed, the value of N is _____.

Answer: 5, or 6 minus 1

4.) The next statement after 30 is executed returns control to statement 15, where N now has a value of_____, and selects statement_____to be executed next.

Answer: 5
4, the fifth statement

III.B.15 (Cont.)

5.) After statement 4 is executed, the value of A is _____, and the next statement to be executed will be_____.

Answer: B/C
GO TO 30

6.) After statement 30 is executed this time, the value of N is _____.

Answer: 4, or 5 minus 1

7.) Again, the next statement after 30 returns control to statement 15, where again, N has a new value of_____, and selects statement_____to be executed next.

Answer: 4
100 (the fourth statement)

III.B.16

There is no limit to how many statement numbers a computed GO TO may have, and the numbers may appear more than once. Care should be taken to ensure that the control variable is greater than zero and not larger than the number of statement numbers within the parentheses.

For the following statement, what is the maximum value for LIMIT?

GO TO (23, 100, 120, 23, 456, 89), LIMIT _____

Answer: 6

If LIMIT were given a value of zero, control would not transfer to any of the listed statement numbers. True or false? _____

Answer: True

If LIMIT were given a value of 7, control would not transfer to any of the listed statement numbers. True or false? _____

Answer: True

III.B.17 The only valid use of the computed GO TO statement is when the control variable's value is defined by a mathematical operation (either being calculated or set equal to a value). Using the ASSIGN statement will result in erroneous operations.

Classify the following statements as either unconditional, assigned, or computed GO TO statements.

1.) GO TO I _____
2.) GO TO 32 _____
3.) GO TO KOUNT, (44, 27, 287) _____
4.) GO TO (122, 547, 7, 945, 46), JSWITCH _____

Answer: 1.) assigned
2.) unconditional
3.) assigned
4.) computed

III.B.18 Work exercise III.B in your workbook at this time.

III. C Logical Expressions

III. C. 1 A logical expression has a value of either false or true. The value false has all bits set to zero, whereas the value true has all bits set to nonzero, or one.

The value false has all bits set to _____, the value true has all bits set to _____, or _____.

Answer: zero, nonzero, one

The value of a logical expression may be tested by the program and used as an on/off switch. True or false? _____

Answer: True

III. C. 2 Two forms of a logical expression are logical constants and logical variables.

III. C. 3 The logical constants are .TRUE. and .FALSE., which for .TRUE. have all bits set to _____ and for .FALSE. have all bits set to _____.

Answer: one, zero

III. C. 4 You have learned that a logical variable has the following characteristics:

(1) It must be declared LOGICAL in a type statement.

(2) It can be subscripted like other variables.

(3) Its value must be set to true or false by the program, either by making it equal to one of the logical constants (as shown in II. E. 14), or by other operations.

A logical variable must be declared LOGICAL in a _____ statement.

Answer: type

A logical variable can be _____ like other variables.

Answer: subscripted

The value of the bits in a logical variable must be set to true or false by the _____.

Answer: program

III. C. 5 Now we will see two other kinds of logical expressions; relational expressions, and the logical result of performing a logical operation on logical expressions. Although these two kinds of logical expressions look more involved, they still meet the basic requirement of having final values of either true or false.

All logical expressions, of whatever type, must have values of either _____ or _____.

Answer: true, false

III.C.6 Relational expressions, the next type of logical expression, show the result of comparing two arithmetic expressions by use of the following logical operators:

.EQ.	Equal to
.NE.	Not equal to
.GT.	Greater than
.GE.	Greater than or equal to
.LT.	Less than
.LE.	Less than or equal to

An example of a relational expression would be A.EQ.B, which could also be written enclosed in parentheses, as (A .EQ. B), if desired.

Two ways of writing the relational expression A is equal to B are: _____ and _____.

Answer: A .EQ. B
or
(A .EQ. B)

III. C. 7 Complete the logical operator for each definition:

		Answer:
Equal to	._____.	.EQ.
Not equal to	._____.	.NE.
Greater than	._____.	.GT.
Greater than or equal to	._____.	.GE.
Less than	._____.	.LT.
Less than or equal to	._____.	.LE.

III.C.8 In the example A .EQ. B, the relation of A to B is evaluated by asking the question: does A minus B equal zero? If A and B have the same arithmetic value, then the relational expression A .EQ. B is true, and the value of the relational expression would be true, or nonzero. However, if A should be greater or less than B, then A .EQ. B would be false, and have a value of zero.

Notice that the variables A and B above must not be logical variables, but arithmetical, that is, they must be integer, real, double precision, or complex variables. Whenever any relational expression contains more than one type of arithmetic variable, the entire expression is converted internally according to the rules of mixed-mode arithmetic you learned in Section II.C.

In the example I .LT. D, where I is an integer variable and D is a double-precision variable (which you learned in Section II is floating point), the value of I would be converted to a floating point quantity by the computer, and the resulting two floating point values would be compared, using only the most significant half (the single-precision part) of D in the comparison. True or false? _____

Answer: True

III.C.9 Indicate the value, true or false, of the following relational expressions, where K = 4 and B = 4.0.

B.EQ.K value_____ Answer: True

B.NE.K value_____ False

B.GT.K value_____ False

B.GE.K value_____ True

B.LT.K value_____ False

B.LE.K value_____ True

III.C.10 Now the fourth and last type of logical expression, which of course still has a resultant value of true or false, is what we call the logical result of performing a logical operation on logical expressions.

The logical expressions on which the logical operations are to be performed can be any of the three kinds we have already learned about: logical constants, logical variables, or relational expressions.

A logical expression can be a logical constant, a logical variable, a relational expression, or the result of a logical operation on any of these. True or false?_____ Answer: True

III.C.11 The logical operators to be used with these logical expressions are the three you used in Section II.F. However, in the present case you are not allowed to use arithmetic constants or arithmetic variables, but only logical expressions with the operators. As you remember, the operators have the following meanings:

.AND. This tests two expressions to see if they are both true. The result is true if both are true, but is false if either one is false.

.OR. This tests two expressions to see if either one or the other is true. The result is true if either one is true, but false only if both expressions compared are false.

.NOT. Indicates negation. This is true if the one value it tests is not true (value of zero), but the .NOT. operator results in a value of false if the expression tested has a value of true. In other words, the resulting value of the .NOT. operator is just the opposite of the value of the expression.

III.C.12 For example, (A .AND. B) is true if and only if A is true and B is true (that is both A and B have all bits set to ones).

On the other hand, (A .OR. B) is true (value of ones) if either A or B is true (bits set to ones).

And (.NOT. A) will be true (have value of ones) if A is false, or not true (has value of zero); but the expression (.NOT. A) will be false (have value of zero) if A is true.

From the foregoing, figure out the values of the following logical expressions:

If A is true, and B is true, then (A .AND. B) has value_____	Answer: True
If A is true, and B is false, then (A .AND. B) has value_____	False
If A is false, and B is false, then (A .AND. B) has value_____	False
If A is true, and B is true, then (A .OR. B) has value_____	True
If A is true, and B is false, then (A .OR. B) has value_____	True
If A is false, and B is true, then (A .OR. B) has value_____	True
If A is false, and B is false, then (A .OR. B) has value_____	False
If A is true, then (.NOT. A) has value_____	False
If A is false, then (.NOT. A) has value_____	True

III.C.13 An allowable combination of these logical operators is .NOT. combined with (and following) .AND. or .OR., such as in A .AND. .NOT. B, and A .OR. .NOT. B, in which cases the .NOT. is evaluated first, then the other operator is evaluated. These could also be written A .AND. (.NOT. B) and A .OR. (.NOT. B).

For example, A .AND. .NOT. B would be true (have bits set to all ones) only if A is true and B is false, or, in other words, A is true and B is not true.

.NOT. can be used with itself only in the form .NOT. (.NOT. A), or .NOT. (.NOT. (.NOT. A)), in which the value would be calculated first with the innermost parentheses, etc. For instance, if A in the above has value true, then the innermost expression (.NOT. A) has value false: the next .NOT. reverses the value to true, and the outermost .NOT. again reverses the value to false. As mentioned in III.C.6, an outer pair of parentheses around the entire expression is always permitted, but does not change the meaning.

From the foregoing, figure out the values of the following logical expressions:

If A is true and B is true, then (A .AND. .NOT. B) has value_____	Answer: False
If A is true and B is true, then (A .OR. .NOT. B) has value_____	True
If A is true and B is false, then (A .AND. .NOT. B) has value_____	True
If A is true and B is false, then (A .OR. .NOT. B) has value_____	True
If A is false and B is false, then (A .OR. .NOT. B) has value_____	True
If A is true, then (.NOT. (.NOT. A)) has value _____	True
If A is true, then (.NOT. (.NOT. (.NOT. A))) has value_____	False

III.C.14 We have seen in the preceding sections how the logical operators .AND. or .OR. and .NOT. can perform logical operations on logical expressions. We have seen that a logical expression can be any of the following: logical constants, logical variables, relational expressions, and last of all the logical result of performing a logical operation on any logical expression.

In all cases, the logical expression operated on had a value of true or false, and the resulting value after using the logical operators had a value of_____or _____.

Answer: true, false

III.C.15 The next, and obvious, step would be to use any desired kind of logical expression with one of the logical operators (.AND. or .OR. or .NOT.). In the form

(logical expression) .AND. (logical expression)

we could substitute any logical expression that we have been writing:

1.) A, B, G, K, etc., if the variable has been declared LOGICAL in a type statement.

2.) .TRUE. or .FALSE.

3.)

A .EQ. B-D	I .NE. K(N)	(Remember, in relational
A .GT. 16.0	I .NE. 7-K	expressions, the operands--variables
K .LT. 16	INDEX(N) .LT. 10	or constants--are mathematical, not
R(I) .GE. R(I-1)	SUM .EQ. PARAM	logical; only the result is logical.)
3.0*X+6.0 .LE. D-E	ISUM .EQ. IPARAM	

III. C. 16 Considering the definitions above, which of the following is not a valid logical expression? (Assume that none of the variables are logical.)

1.) A .EQ. B-D

2.) A = B-D

3.) PAY .EQ. TIME * RATE

4.) I .LT. 2

5.) I .LT. 2.6

6.) A .EQ. .FALSE.

7.) I .EQ. .31415E01

8.) I .GT. -3.1415E+01

9.) I .LT. 31415927D0

10.) I .EQ. (15., 16.7)

Answer: Not valid:
2.) A = B-D
6.) A .EQ. .FALSE.

III.C.17 If you answered correctly the question above, skip to III.C.19. If not, let's look at each of the above examples, remembering the definitions.

1.) A .EQ. B-D is a valid relational expression, using a valid relational operator and arithmetic real variables.

2.) A = B-D is not a valid relational expression because the = sign is not a valid relational operator. This expression does not have a resulting value of false (zero) or true (all ones), but rather is an arithmetic replacement, replacing the value in A with another value that is the difference of B-D.

3.) PAY .EQ. TIME * RATE is valid, using a valid relational operator, and arithmetic real variables.

4.) I .LT. 2 is valid, using arithmetic integer variable and constant.

5.) I .LT. 2.6 is valid, using arithmetic expressions, I being an integer variable and 2.6 a real constant.

6.) A .EQ. .FALSE. is not a valid relational expression, because both operands are not arithmetic expressions, since .FALSE. is a logical constant. (Please review Section II.E.14, where A and K are declared logical variables, and then K = .FALSE.)

7.) I .EQ. .31415E01 is valid, using an integer variable and a real constant.

8.) I .GT. -3.1415E+01 is valid, using an integer variable and a real constant.

9.) I .LT. 31415927D0 is valid, using an integer variable and a double precision constant.

10.) I .EQ. (15., 16.7) is valid, using an integer variable and a complex constant. The comparison will take place between the variable and the real part of the complex constant.

III.C.18 Let's put together some typical examples of logical expressions and see how the true or false value of each is calculated:

1.) A type statement has declared A a logical variable and another statement has set A = .TRUE. ; now consider the logical expression (A) .AND. (K .LT. 16).

The expression (K .LT. 16) could have the value true or false, depending on the arithmetic value of K. Next, the true value of A is compared with the true/false value of (K .LT. 16); if both are true, then the logical value of the entire expression is true, or bits are set to all ones; however, if (K .LT. 16) is false, then the entire expression has the value false, or zero.

2.) Another case occurs when relational expressions compare arithmetic variables that have previously been given values by the program. For example, assume that the program has set J, K, and L equal to integer values. In each of the following logical expressions, the relational expressions would be evaluated first, then their values would be compared by the logical operators. Obviously, the possible combinations in this type are numerous.

```
J .EQ. 6 .AND. J .GE. K+L          (J .GE. K+L) .AND. (.NOT. (L .EQ. K) )
J .EQ. K .AND. K .GT. L            J .EQ. K .OR. J .GT. L
J .LT. K .OR. J .NE. L             J .LT. 12 .AND. J .GE. 9
```

III.C.19 Indicate the true or false value of the logical expression when the variables assume the values given:

If B = 8, K = 1, then (B .GT. 6 .AND. K .LT. 4) has value_____	Answer: True
If B = 3, K = 1, " " " " " " " has value_____	Answer: False
If B = 3, K = 4, " " " " " " " has value_____	Answer: False
If B = 7, K = 4, " " " " " " " has value_____	Answer: False
If B = 7, K = -3, " " " " " " " has value_____	Answer: True
If J = 6, K = 4, L = 2, then (J .EQ. 6 .AND. J .GE. K + L) is _____	Answer: True
If J = 6, K = 4, L = 2, then (J .EQ. 4 .AND. K .GT. L) is_____	Answer: False
If J = 6, K = 4, L = 2, then (J .EQ. K .OR. J .GT. L) is_____	Answer: True
If logical variable A has been set equal to true, then (.NOT. A) is_____	Answer: False

III.C.20 Work exercise III.C in your workbook before starting section III.D.

III.D IF Statements

III.D.1 The IF statement is a versatile control statement that can both test a value and then direct the computer to choose one of various courses of action, as determined by the value tested. In other words, the IF statement can cause the computer to branch to some other statement instead of executing the next sequential statement.

The IF statement can cause the computer to skip the _____ _____ _____, and _____to some other statement.

Answer: next sequential statement, branch

III.D.2 You have already seen how the programmer can use the GO TO statement for branching. Now you will see how the IF statement determines branching by testing arithmetic or logical expressions for some value.

The IF statement tests the values of_____or _____ expressions to determine branching.

Answer: arithmetic or logical

III.D.3 To review briefly these expressions that can be tested by the IF statement, and how they get their values, fill in the blanks in these lists:

1) The IF statement can test arithmetic expressions, which, as you recall, are those that express one or more of the five basic arithmetic operations:

_____
_____
_____
_____
_____

Answer: exponentiation
multiplication
division
addition
subtraction

III.D.3
(Cont.)

2) The IF statement can be written another way to test logical expressions, which you remember are those that always have a value of either true or false. The four types of logical expressions tested by the IF statement get their true or false values as follows (you fill in the blanks):

a) The two logical constants have a value of zero or non-zero and are named __.____.__ and __.____.__

Answer: .FALSE.
.TRUE.

b) The logical variable must be declared in a_____ statement, and may be subscripted like other variables. It is given a _____or_____value by

Answer: type
true, false

1. Setting it equal to one of the_____constants, or by

Answer: logical

2. Giving it the_____value of a logical or relational expression, or by

Answer: logical

3. Setting it equal to the_____or_____ value of an arithmetic expression.

Answer: zero
non-zero

c) Relational expressions compare _____expressions by means of the logical operators .EQ. , .NE. , .GT., _____, .LT. and _____.

Answer: arithmetic
.GE.
.LE.

d) Other logical expressions are those expressions whose value is the logical result of performing a _____ operation on logical expressions.

Answer: logical

III.D.4 Now let us see how the IF statement tests these expressions to determine branching, and how the IF statement can utilize the GO TO statement to give it greater flexibility.

III.D.5 When the programmer uses the IF statement, the computer is told in one statement what to test, what values to test for, the choice of alternate statement numbers it may execute, and which value will direct the computer to which statement number.

To test the value of a FORTRAN expression, either arithmetic or logical, the programmer uses the_____statement.

Answer: IF

The same IF statement will both test the expression and tell the computer which _____to execute next.

Answer: statement

III.D.6 There are three kinds of IF statements, one using an arithmetic control expression, and two using a logical control expression. They are:

1) The three-branch arithmetic IF, as its name implies, has an _____ control expression, and can cause the computer to branch to any of the _____ different statements.

Answer: arithmetic
three

A possible example could be:

IF (A-2.0) 20, 30, 40

2) The two-branch logical IF, as its name implies, has a _____ control expression; and, like the arithmetic IF, offers a choice of branching statements. However, it differs from the three-branch arithmetic IF in that it gives the computer a choice of only _____ different statement numbers for branching. A typical example would be:

Answer: logical
two

IF(A.LE.B) 20, 30

3) The one-branch logical IF, like the two-branch logical IF, has a _____ control expression. However, as the name implies, the computer is given only one choice; whether or not to execute the last part of the IF statement, which is some executable statement. If the computer is told by the control expression not to execute this, then the computer will merely go on to the _____ sequential statement. A typical example would be:

Answer: logical
next

IF (A.LE.B) I = I + 1

III.D.7 The Three-Branch Arithmetic IF.

III.D.8 The IF statement is very simple to use. The word IF is written followed by an arithmetic control expression in parentheses and then three statement numbers separated by commas.

IF(expression) N(-), N(0), N(+)

Which of the following examples of three-branch arithmetic IF statements are coded incorrectly?

1) IF (A-2) 20 30 40
2) IF A-2, 20 30 40
3) IF (A-2) 20, 30, 40
4) IF (A-2), 20, 30, 40

Answer: 1, 2, and 4

III.D.9 The IF statement causes the computer to evaluate the control expression, and, if the expression is negative, the first of three branch options is chosen. If the expression is equal to zero, the second option is taken. The third option is chosen for a positive value. N(-) is the statement number which will be executed next if the expression's value is less than zero, N(0) if it is equal to zero, and N(+) if it is greater than zero.

For X = 5.0, which statement number will the computer branch to after executing the following IF statement?_____

IF (X - 5.0) 100,200,300

Answer: 200

If you wished to go to statement 300, you would have to set the value of X to_____.

Answer: X > 5.0

If your answers were correct, skip to III.D.11.

III.D.10 Since the first thing the computer does is to evaluate the control expression, (X - 5.0) would be evaluated. For X = 5.0, this result is_____. The three statement numbers represent the three options for branching. The first statement number (100) is the branch for a_____value in the control expression, but in our example the control expression (X - 5.0) equals zero, so the computer branches to statement number_____.

Answer: zero

negative

200

III.D.11 The control expression may be as simple or as complex as desired. It may be a variable that has previously been given a numerical value, or it may be longer and contain its own sets of parentheses. It must, however, be completely contained within the IF statement parentheses.

The following are valid arithmetic IF statements:

```
IF (X) 254, 100, 400

IF ((X + Y)**3 / (Z + X/2.)**2) 300, 400, 500

IF (X + Y) 111, 200, 200
```

Answer: True

True or false? _____

III.D.12 Notice the last IF statement in the above question. The statement numbers for the zero and positive options are the same. This means that statement number 200 will be executed if the control expression is greater than or equal to zero.

Write the IF statement which will branch to statement number 100 if (X + Y) is less than or equal to zero and branch to 200 if (X + Y) is greater than zero. _____

Answer: IF(X + Y) 100, 100, 200

III.D.13 The IF statement is valuable because it allows the computer to operate in its most efficient mode, that is, in the performance of repetitive operation. A simple example of this is to use the IF statement to control the number of repetitions in developing some quantity.

You might wish, for example, to do some calculations repeatedly until the resultant value reaches some desired limit, regardless of the number of loops (repetitions of the calculations). In this case, the IF control expression must contain this calculated value, such as:

```
 95  IF (X * Y) 111, 300, 300

111  X = X + 1.0

     GO TO 95

300 (program continues)
```

In the example above, the calculation that will be repeated is _____.

Answer: X * Y

If the resultant value of this calculation after each loop is _____, the program goes next to statement 111, where the value of X is increased by _____.

Answer: negative

1.0

The IF statement will send the computer to statement 300 when the result of X*Y is _____ or _____.

Answer: zero or greater

III.D.14 Another kind of repetitive operation with the arithmetic IF uses a different control variable that is not the result of your calculations. Suppose that you wish to repeat the calculations a specified number of times, regardless of the final value. In this case, you will use a counter for loop control, which will be incremented (or decremented if you wish) each time the loop is repeated until the counter, when tested, shows that you have completed the desired number of loops. In this case, the counter will be the IF control expression, as in this example where I is the counter:

```
      DIMENSION A(3)

      I = 0
20    I = I + 1

25 A(I) = (X + Y) **3/(Z + Y/2.0)**2

   IF (I - 3) 20, 30, 30

30 (program continues)
```

In the example here, when statement 25 has been executed the first time, the value of (I) is _____, and the value of the control expression (I - 3) is_____. This will cause the computer to go to statement_____.

Answer: 1
-2 , 20

After statement 25 has been executed the third time, I = _____, and the value of the control expression I - 3 will be_____, which will cause the computer to go to statement_____.

Answer: 3
0
30

III.D.15 Suppose you wish to use the three-branch arithmetic IF statement to tell the computer to add up the first 100 numbers out of a large array of real numbers, and to accumulate the total in a variable named SUM. Then, after this is done, you wish the computer to go on with the rest of the program.

Look at the following example:

C for Comment — FORTRAN CODING FORM

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
		DIMENSION XNMBERS(400)	
2		SUM = 0.0	
5		I = 1	
10		IF (I - 100) 20,20,40	
20		SUM = SUM+XNMBERS(I)	
30		I = I+1	
		GO TO 10	
40		(program continues)	

In this problem, XNMBERS is the source array from which the computer will get _____ number at a time for adding in SUM. Now your counter (control variable) I will do double duty: at the beginning of the first loop, I = _____, and since I tells the computer to get the number from the "Ith" location in the array XNMBERS(I), the computer will now get the _____ number from the array. Likewise, during the fourth loop, the _____ number will be added, and during the hundredth loop the _____ number will be added.

Answer: one

1

first

fourth

hundredth

III.D.16 Still referring to the example in III.D.15, notice that at the beginning, SUM was initialized to 0.0 before you started adding numbers to it. Then the control variable I was "initialized" to 1. This ensures that when statement 20 is executed for the first time, SUM is set to the value of _____ plus the value of the _____ number in XNMBERS.

Answer: zero
first

During the 50th execution of statement 20, SUM (containing the accumulated total of the first _____ numbers) has the _____ th number added to it, raising the SUM to the new accumulated total.

Answer: 49
50

During the execution of the loop for the 99th time, the control variable will have the value of _____, and statement 20 will add the _____ th value into SUM.

Answer: 99
99

Then after statement 20 adds the 99th value into SUM, statement 30 increments the control variable from 99 to _____.

Answer: 100

Next the computer is sent back to statement 10, the IF statement, to start the 100th loop. Now the value of the control expression (I - 100) is _____, so the computer is directed to statement _____, where the _____ th number is added to SUM.

Answer: 0, 20
100

Notice that now, after the 100th number is added to SUM, the computer again increments the control variable I, so that it now equals 101. But, the 100 desired numbers have already been totalled in SUM, so you wish to get out of the loop and go to the rest of the program. Will you? Yes or no? _____

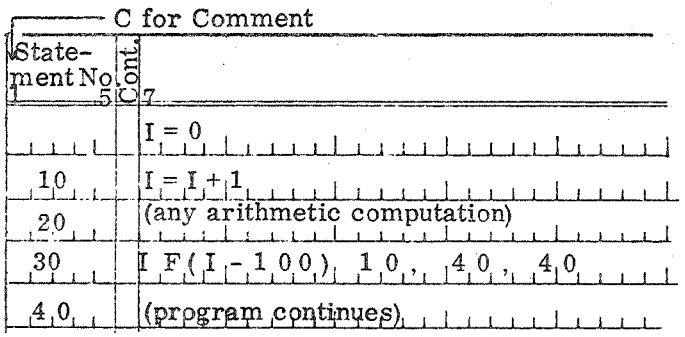
Answer: yes

Follow the coding. You next go back to statement 10, the IF statement, where now I - 100 has the value of _____, so the computer will branch to statement _____. Does this take you out of the loop? _____

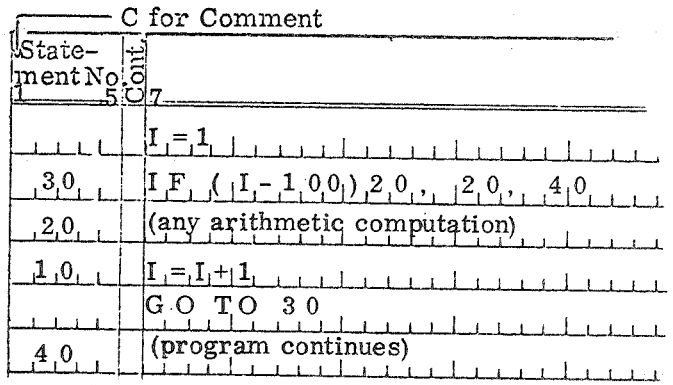
Answer: +1
40, yes

III.D.17 This brings up the related question of how to initialize a counter for a loop, and where to place your statement that increments the counter. Look at the two examples below. Example 1 is organized like the sample in III.D.14 whereas Example 2 is like the coding in III.D.15. Either plan will work, but look at their effect on the IF statement and its statement number used for branching if the control variable equals 0 (the second option).

Example 1



Example 2



In both examples here, notice that the variable I correctly has the value of_____ before executing the first iteration of the computation and it likewise has the value of _____ before executing the 100th computation. But, in the two examples, the computer tests the value of I at a different time relative to incrementing the counter.

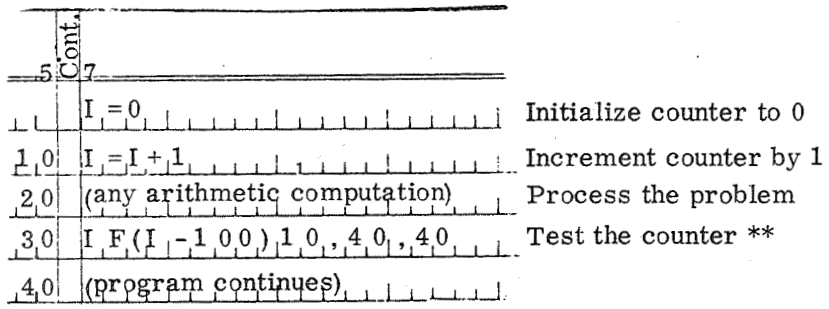
Answer: 1
100

In Example 1, the IF statement's second branch (if I - 100 = 0) is to statement _____, but in Example 2 the second branch is to_____.

Answer: 40, 20

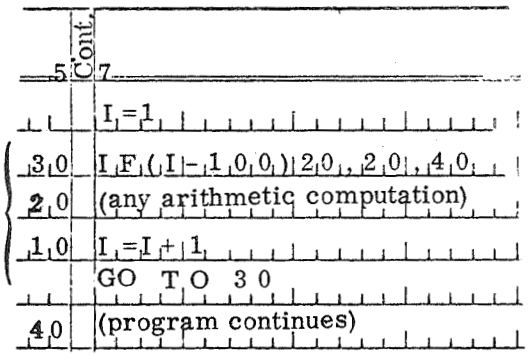
III.D.18 The sequence of statements in the two examples in III.D.17, repeated here for convenience, differs as shown here:

Example 1



Initialize counter to 0
Increment counter by 1
Process the problem
Test the counter **

Example 2



Initialize counter to 1
Test the counter ***
Process the problem
Increment counter by 1

**If (I-100) is negative, continue the loop, go to 10.
If (I-100) is 0 or positive, exit from loop, go to 40.

***If (I-100) is negative or 0, continue the loop, go to 20.
If (I-100) is positive, exit from loop, go to 40.

Look again at the two examples in III.D.17 and 18 and answer these questions:

1) Suppose that statement 20 (the processing step in each example) has just been performed for the first time, with the counter I set to the value of 1. In each example, follow what happens after statement 20: the very next time the IF statement tests the counter, in Example 1, I = _____, but in Example 2, I = _____.

Answer: 1, 2

III.D.18 (Cont.)

2) Continuing this line of reasoning, suppose that the processing (computation) has just been performed for the 100th time (with I = 100): the very next time the IF statement tests the counter, in Example 1, I = _____, therefore, the control expression I - 100 has the value of _____, which causes the computer to branch to statement_____, which in turn (does?/does not?) lead to processing another time.

Answer: 100
0
40, does not

In the same situation, in Example 2, after statement 20 has processed for the 100th time, with I = 100, the very next time the IF statement tests the counter, I = _____, therefore, I - 100 has the value of _____, which causes the computer to branch to statement_____, which (does ?/ does not?) lead to processing another time.

Answer: 101
+1
40
does not

III.D.19

The two different situations in the examples just described reveal that, if you test the control variable after you increment it, you must write the branch statement options in the IF statement so that you exit from the loop when your control variable value is_____ _____zero.

Answer: equal to or greater than

However, if you test the control variable before you increment it, you must write the branch statement options in the IF statement so that you exit from the loop when your control variable value is_____ _____zero.

Answer: greater than

III.D.20 Notice that in the examples we have studied of the IF statement, the result from the evaluation of the control variable was not saved. However, if the programmer later needs the value of the expression, he will have to write a separate statement setting the expression equal to a variable. In the example in III.D.9, if the programmer needed the result of the control expression (X - 5.0) for some other calculation, he would write

```
Y = X - 5.0
IF (Y) 100, 200, 300
```

Then he could later refer to Y in other statements, because he would have saved the value of X - 5.0 in it.

III.D.21 The Two-Branch Logical IF

III.D.22 Both logical IF statements test logical expressions, which, as you recall, has only two possible values, true or false. One of these IF statements, the two-branch logical IF, works the same as the three-branch arithmetic IF except that the word IF is followed by a logical expression in parentheses, and there are only two branching options following the logical control expression.

The first option is chosen if the expression is true, the second if the expression is false.

For X = 3.0 and Y = 4.0, which option will be chosen by the following logical IF statement? _____

IF (X.GT.Y) 100, 200

Answer: 200

If your answer is correct, skip to III.D.24.

III.D.23 In the two-branch logical IF, as in the arithmetic IF, the action the computer takes depends on the result of the control expression. In the above case X is compared to Y. If X>Y, the state of the expression will be true and the first option, statement number 100, will be chosen. But we see that for X = 3.0 and Y = 4.0, X is not greater than Y; therefore, the expression is false and the second option, statement number 200, is chosen.

III.D.24 The One-Branch Logical IF

III.D.25 The second type of logical IF, the one-branch logical IF statement, works differently from the other IF statements. Instead of giving a choice of statement numbers for branching, one optional statement itself is written on the same line, immediately following the control expression. This statement following the control expression may be any executable FORTRAN expression, which the computer will or will not execute, depending on the control expression. (However, this executable expression should not be another logical IF statement.)

Examples of one-branch logical IF statements:

```
IF (X.GT.Y) AREA = Q * R ** 2

IF (RADIUS.GT.4.5.OR.ARC.EQ.1.7) GO TO 300

IF (A.OR.B) C = .TRUE.
```

Which of these examples is not a valid one-branch logical IF statement? _____

a.) IF (A.EQ.B) GO TO 200

b.) IF (A = B) GO TO 200

c.) IF (A.NE.B) IF (A.GT.B) GO TO 50

Answer: b and c

III.D.26 When the computer encounters a one-branch logical IF statement, it evaluates the control expression first. Next, if the control expression is true, the optional statement is executed. After execution of the optional statement, if that statement does not cause a branch to another part of the program, the next statement after the logical IF is executed. However, if the control statement is false, the entire logical IF statement is treated as though it is not there. It is disregarded and the next sequential statement after the logical IF is executed.

III.D.27 For example, in the following coding, suppose the computer evaluates the control expression.

```
IF (X.EQ.0.0) Y = 100.0

GO TO 300
```

If X has the value of 0.0, the expression (X.EQ.0.0) is true; therefore, the computer would next perform two operations:

1.) Y would be set equal to 100.

2.) Then the computer would go to the next sequential statement, which would cause the program to branch to statement 300.

However, if X had some non-zero value (not equal to zero), making the control expression false, the computer would next perform two operations:

1.) The computer would ignore the instructions to set Y equal to 100.0.

2.) As above, the computer should then go to the next sequential statement, which would cause a branch to statement 300.

III.D.28 In these examples of the one-branch logical IF statement, suppose you "play computer" and evaluate the control expression; then indicate which statement you will execute next (either the statement in the right end of the IF statement, or the statement on the next line).

a.) Let X have value of 2.0

```
IF (X.GE.1.0) Y = 5.0
GO TO 300
```

_____

Answer: Y = 5.0

b.) Let X have value of 0.0

```
IF (X.LT.0.0) A = B-X
GO TO 300
```

_____

Answer: GO TO 300

c.) Let X have value of -1.0; give branching options for each IF statement:

```
 30 IF (X.EQ.0.0) GO TO 200
    GO TO 100
100 IF (X.GT.0.0) GO TO 300
    GO TO 400
400 XNEG = X
    GO TO 50
200 OMEGA = X
    GO TO 50
300 XPOS = X
 50 (program continues)
```

_____

_____

Answers:
For statement 30: GO TO 100

For statement 100: GO TO 400

III.D.29 The one-branch logical IF statement allows the programmer to "slip in" an extra statement when conditions call for it.

Write the FORTRAN statements, using the one-branch logical IF for the following:

When the logical variables A and B are both true, add one to the variable X, then calculate $Y = X^3$. Otherwise, calculate $Y = X^3$ only.

Answer:
```
IF (A.AND.B)X=X+1.0
Y = X ** 3
```

III.D.30 Work exercise III.D in your workbook at this time.

III. E The DO Statement

III. E. 1 It was mentioned earlier that the computer becomes a more efficient device when it works in a repetitive mode. Because of this, programmers take great pains to use repetitive loops as often as possible (within reason) in their programs. You have used IF statements to execute repetitive loops, and will now see a more powerful tool for this, the DO statement. All repetitive loops have certain properties in common:

1. There is an initial value for the variable that governs the number of repetitions.

2. The variable is incremented by some amount after each repetition.

3. The program provides a means of testing the variable to determine whether to repeat the loop or to exit from the loop.

III. E. 2 It is not surprising then to learn that a FORTRAN control statement exists which automatically controls the above three properties of repetitive loops. This statement is the DO statement. With one DO statement, the programmer defines a variable, gives it an initial value, sets the upper limit on the value of the variable, defines the value of the increment and determines the set of statements to be repeated.

A section of code may be repeated several times by use of a DO statement. True or false? _____ Answer: True

III. E. 3 The DO statement provides a simple way to make calculations that are quite complicated when done with individual instructions.

The DO statement does several things simply and automatically.
True or false? _____ Answer: True

III. E. 4 The format of the DO statement is the word "DO" followed by the terminus statement number and then an integer variable, called the index variable, whose value will be initialized, incremented and tested with each repetition. The integer variable is followed by an equal sign and then three integers called the loop parameters which represent the initial value, final value, and the increment to be given to the index variable.

The three values following the integer variable are the loop _____. Answer: parameters

III. E. 5 For example: 500 DO 100 I = 1, 21, 2

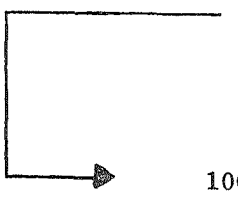
The above DO statement will be interpreted by the computer to execute the statements from 500 through 100, with the variable I initially equal to 1, repeating this series of statements as many times as necessary until I > 21, with I being increased by 2 each time statement 100 is reached.

How many values will I assume during execution of the above loop controlled by DO statement 500? _____ Answer: 11

III.E.6 In the above example, I is initially set equal to 1. When statement 100 is reached, the increment 2 is added to I. If I is greater than 21, the next statement in sequence is executed. If not, the program returns to the statement following 500. When statement 100 is again reached, the process is repeated until I is greater than 21. Thus, I will assume the values 1, 3, 5 . . . 19, 21 during execution of the loop. When I reaches 23, the statement following 100 is executed.

III.E.7 The section of the program which is going to be repeated under control of the DO statement is called the DO loop. The last statement in the loop is assigned a statement number. The DO loop contains the DO statement, the ending statement, and all the statements in between.

The following example of a simple DO loop has an arrow showing the loop structure:



```
      SUM=0.0
      DO100 I=1, 21, 2
      A(I)=B(I)-C(I)+D(I)
      E(I)=F(I)*12.0
      G(I)=A(I)+E(I)
      SUM = SUM + G(I)
  100 CONTINUE
      (program continues)
```

The CONTINUE statement will be discussed later.

In the following example, an arrow similar to the one above would point to statement _____.

Answer: 20

```
      DO 20 ICOUNT = 2, 16, 1
      A = B**2*D
      ARRAY (ICOUNT) = A
   20 CONTINUE
```

III. E. 8 The index variable is restricted to a simple integer variable. The initial value, maximum value, and increment to be given to the index variable may be either integer constants or simple integer variables. If they are constants, they must be positive and non-zero. If variables are used, they may assume positive, negative, or zero values. However, zero or negative index variables can generate many errors, and should be avoided. On computed variables, tests should be made to prohibit zero or negative variables from being used.

Detect the errors in the following DO statements:

a. DO 206 X = 1, 5, 2 _____

b. DO 100 I = 0, 40, 3 _____

c. DO 200 L = 1, B, 3 _____

d. DO 300 J = I, K, (M+1) _____

Answer:

a. Index variable must be an integer variable.
b. A zero constant is not allowed as one of the loop parameters.
c. The loop parameters must not be real variables.
d. Although integer variables are allowed, integer expressions are not.

III. E. 9 If the incrementing value (third parameter) is 1, this parameter may be omitted and 1 will be assumed. The following statements are equivalent.

DO 100 J = 1, 40, 1 DO 100 J = 1, 40

Are these statements equivalent? _____

DO 100 J = 1, 40, 2 DO 100 J = 1, 40

Answer: No. If the third parameter is omitted, it is assumed to be 1.

III. E. 10 The DO loop terminates when the incremented value of the index variable is greater than the value given by the maximum value loop parameter. If the initial value of an index variable is greater than the maximum value, the loop will be performed once and then terminated.

The following DO statements represent the beginning of three different loops. How many passes will occur through each loop?

a. DO 100 INDEX = 1, 16, 2 _____

b. DO 200 LINK = 5, 4 _____

c. DO 300 KAT = 2, 20, 18 _____

Answer: a. 8

b. 1

c. 2

III. E. 11 Because the loop parameters may be integer variables, the programmer can construct loops whose number of repetitions can be changed outside the loop by computations. You will see a very effective use of this characteristic when you study nested DO loops later in this section. (The loop parameters may also be changed during the execution of the loop, but it is not good practice to do so and avoidance of the procedure is encouraged.)

Can the loop parameters in this example be changed by outside computations? _____

DO 20 I = 1, 20

Answer: No, because to be changed, they must be variables, not constants as these are.

III. E. 12 Let's look again at the reference in the DO statement to the ending statement, such as statement 95 in the following:

```
      SUM = 0
   50 DO 95 I=1, 20
      INDEX(I) = I
   95 SUM = SUM + INDEX(I)
  100 (program continues)
```

Notice that statement 95, the ending statement, is an executable statement, an arithmetic computation. This is a valid terminal statement for a DO loop.

However, there are some restrictions on what kind of statement may be used to end a DO loop. In general, it must be an executable statement, so this rules out all the non-executable statements you have already studied, plus COMMON and FORMAT that you will see later.

The end statement in a DO statement must not be a FORMAT statement, DIMENSION statement, COMMON statement, or type declaration. True or false? _____

Answer: True

The ending statement in a DO statement must always have its own statement number. True or false? _____

Answer: True

III. E. 13 Further, the ending statement referenced in a DO statement must not be certain other statements you will learn about a little later, including the PAUSE, STOP, END, and RETURN statements.

The ending statement must not be any of these three executable statements:

1.) Arithmetic IF
2.) DO
3.) GO TO

Which of these statements would be forbidden as the ending statement for a DO loop? _____

1.) DIMENSION . . .
2.) IF (X+Y) 10, 20, 30
3.) GO TO 250
4.) X=Y+Z
5.) LOGICAL . . .
6.) PAUSE
7.) DO 24 I = 1, 10

Answer: All but 4.) are forbidden.

III. E. 14 Now that we have listed what the DO loop's last statement cannot be, let's see what is left that it can be:

1.) Executable statements are permitted, such as arithmetic expressions (but excluding those we just listed above such as GO TO, another DO, and the arithmetic IF statements).

2.) The all-purpose CONTINUE statement, which will be explained in the next paragraph, is the most useful.

III. E. 14 (Cont.)

3.) The last statement of a DO loop may be a logical IF statement. However, the beginning programmer should be careful in using the logical IF as the ending statement. The one-branch logical IF may change the desired flow of the program, as will be evident in later examples of DO loops. This can have a disastrous effect if nested DO loops end with the same terminal statement. To be "safe rather than sorry ", end each DO loop with a CONTINUE statement, which may be preceded by either an arithmetic IF or a logical IF statement.

Which of these statements would be allowed as a valid last statement in a DO loop? _____

1.) X = X*(-1.0)
2.) CONTINUE
3.) GO TO 100
4.) ISUM = ISUM + J

Answer: 1, 2, 4

III. E. 15

To avoid ending a DO loop with a forbidden statement, FORTRAN provides the CONTINUE statement. This statement is a do-nothing instruction, which causes no operation, but is always a valid ending statement for a DO loop. It is also useful in that it makes the ends of DO loops easy to find on program listings.

The CONTINUE statement is a _____ - _____ instruction.

Answer: do-nothing

III. E. 16 Since the end statement of a DO loop must be indicated in the DO statement, every CONTINUE statement must be assigned a statement number.

A CONTINUE statement at the end of a DO loop does not require a statement number. True or false? _____ Answer: False

III. E. 17 CONTINUE statements may appear at any place in the program. When not ending DO loops, they must still be assigned statement numbers, but they merely pass control to the next sequential instruction.

CONTINUE statements provide statement numbers for branching, but cause no operations to be performed. True or false? _____ Answer: True

III. E. 18 A transfer to a CONTINUE which ends a DO loop is always permissible from statements preceding the DO loop. A transfer to such a CONTINUE from statements following the loop is permissible only if a reference was made to the same CONTINUE in a statement preceding the DO loop. Transfer to the CONTINUE may occur whenever required from within the DO loop.

Suppose statement 200 is a CONTINUE statement which ends a DO loop. Which of the following are valid? _____

1.) Transfer to statement 200 from an IF or GO TO within the DO loop. Answer: 1.) and 2.)

2.) Transfer to statement 200 from a statement preceding the DO loop.

3.) Transfer to statement 200 from a statement which follows statement 200, when no other reference to statement 200 has been made outside the DO loop.

III. E. 19 To illustrate use of the CONTINUE statement, suppose we have a series of 100 values of some function which have been stored randomly in an array called X. It is necessary to find the smallest value of X and make it the first value of the array. (That is, the smallest X must be X(1).) A program to search through the X array would be:

```
 50  DO 100 K = 2, 100
     IF (X(1) - X(K)) 100, 100, 200
200  TEMP = X(1)
     X(1) = X(K)
     X(K) = TEMP
100  CONTINUE
300  (program continues)
```

The DO loop illustrated here starts with statement _____ and ends with statement _____.

Answer: 50
100

III. E. 20

The first time through the DO loop with K = 2, the IF statement compares the first X in the array with the second X. If the first X is smaller or equal to the second X, the IF statement causes a branch to statement number 100, which is a CONTINUE instruction. Because it is the last statement in the DO loop, the computer would interpret the CONTINUE statement as an instruction to increment the index variable and continue processing the DO loop. If the first X is greater than the second, the IF statement branches to statement number 200, where the X values are interchanged, thus putting the smaller X in X(1). This method assures placement of the smallest X in X(1) when all the values in the X array have been compared.

Did you notice how another variable called TEMP was used to help interchange numbers (without losing any of them) by putting them into new locations in the same array?

Suppose that in the preceding coding, the original value of X(1) at the start was 4.0, and at the same time the original value of X(K) was 2.0, thus causing the program to go to statement 200.

"Play computer" by indicating the value of each variable before and after the execution of these statements:

	before	after
200 TEMP = X(1)		
X(1) = X(K)		
X(K) = TEMP		

Answer:

TEMP		4.0
X(1)	4.0	2.0
X(K)	2.0	4.0

III. E. 21 The CONTINUE statement allows the programmer to perform multiple branches within the DO loop. In the above example, there would be no way to repeat the DO loop if the CONTINUE statement was not used. The coding following is the same as above, but with the CONTINUE statement at the end deleted:

FORTRAN CODING FORM

C for Comment

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
		DO 100 K = 2, 100	
		IF (X(1) - X(K)) ?, ?, 200	
200		TEMP = X(1)	
		X(1) = X(K)	
100		X(K) = TEMP	

Can you replace the question marks in the IF statement with statement numbers which would accomplish the recycling of the loop if $X(1) - X(K) \leq 0$? _____

Answer: No

Branching back to the DO statement would cause the loop parameters to be reset to their _____ values. Clearly, a "dummy" statement is required which would allow both branch paths to meet at the end of the DO loop.

Answer: initial

III. E. 22 The reason for the existence of the CONTINUE statement is to provide a common finishing point for branches within the DO loop. There is no limit to the number of branch paths which may occur within a DO loop, but all paths must eventually lead to the statement marking the end of the loop.

In a DO loop, all paths eventually lead to the statement marking the _____ of the loop.

Answer: end

III. E. 23 Look at the following examples, and answer the questions.

Example 1

```
 4  A(I) = A(I) + B(I, J)
 2  GO TO 1
10  DO 1 I = 1, 5
    GO TO 4
 1  CONTINUE
30  (program continues)
```

Example 2

```
    K = 0
    IF (K) 1, 10, 1
 4  A(I) = A(I) + B(I, J)
 2  GO TO 1
10  DO 1 I = 1, 5
 6  GO TO 4
 1  CONTINUE
30  (program continues)
```

1.) In Example 1, after the computer first executes statement 4, control will next be transferred to statement_____, then to statement _____, and then to statement_____.

Answer: 2
1, 30

Will statement 10, the DO statement, ever be executed? _____

No

2.) In Example 2, after the computer executes the first two statements, what statement will it go to next? _____

Answer: 10

After this statement is executed, the next statement to be executed will be_____.

6

After this statement is executed, the next statements in turn to be executed will be _____, _____, then_____.

4, 2, 1

Will statement 10, the DO statement, be executed in this example?

Yes

_____

III. E. 24 An example of the efficiency of the DO loop can be seen by reworking a problem shown earlier, where we discussed how and when to initialize and test the control variable in a loop using the arithmetic IF.

(a) IF statement version:

```
 5  I = 0
10  I = I + 1
20  (any arithmetic computation)
25  IF (I-100) 10, 40, 40
40  (program continues)
```

(b) DO loop version:

```
 8  DO 30 I = 1, 100
20  (any arithmetic computation)
30  CONTINUE
40  (program continues)
```

Notice that, in example (b) above, the statement DO 30 I = 1, 100 does all of the following, which required separate statements in example (a):

1.) Defines the counter variable I.

2.) Defines the amount of the increment for the variable I.

3.) Initializes the counter variable I.

4.) Defines the maximum value for the counter variable I.

Notice that in (a) above the sequence of execution was to initialize the variable, increment it, process the computation, and then test the variable. In (b) above, this was done by the ____loop.

Answer: DO

In both examples, whenever the program leaves the loop, it goes next to statement 40. In example (a) the exit from the loop is when the variable is ____(<, =, >) 100. In example (b) the exit is when the variable is____(<, =, >) 100.

Answer: =
>

III. E. 24 (Cont.)

When the programmer uses a DO loop, he (does, does not) have to decide in what sequence he should initialize, increment, and test the control variable. _____

Answer: does not

The one advantage the arithmetic IF loop has over the DO loop is that the control variable may be other than a simple integer variable. True or false? _____

Answer: True

III. E. 25

There are two ways for the program to leave a DO loop. A control statement may cause a special exit (a branch to another part of the program outside the DO loop), or the loop may be performed a sufficient number of times so that the loop parameters will be satisfied and control will pass to the statement immediately following the last statement in the DO loop. If a DO loop is allowed to terminate naturally, that is, the loop parameters have been satisfied, the value of the index variable is not saved, and consequently is not available for later computations. If the loop is left by a branch before the loop parameters have been satisfied, the value of the index variable is available for subsequent calculations.

If the programmer wishes to save the value of the DO-loop control variable upon special exit, he has the choice of either using the control variable as it is, or of setting some other variable equal to it. True or false? _____

Answer: True

III. E. 26 In this program, each value in an array is tested. If it is zero or positive, some processing is done, and the DO loop keeps cycling; but, if any negative value is found, its location in the array (which is also the value in the control variable) is saved, the loop is halted, and a "special exit" is made immediately.

FORTRAN CODING FORM

C for Comment

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
10		DO 3 I =1, MAX	
		IF (ARRAY(I) .LT. 0.0) 5, 6	
6		ARRAY(I) = ARRAY(I) + TABLE(I)	
3		CONTINUE	
		GO TO 40	
5		ISAVE =I	
20		(program for negative values)	
		.	
		.	
40		(program for no negative values)	

The DO loop is made up of statements _____ and _____ with all the statements between these.

Answer: 10, 3

III. E. 26 (Cont.)

Suppose that during the first iteration of the DO loop the value in ARRAY(1) is tested by the IF statement and found to be positive. Then the next two statements to be executed will be_____and _____.

Answer: 6, 3

Suppose no negative values are found and the DO loop has completed all iterations up to and including the value in MAX. Control is then passed to the next sequential instruction following the CONTINUE statement. This statement then transfers control to statement____.

Answer: 40

On the other hand, suppose that, during any loop, the IF statement found a negative value in ARRAY. The next statements to be executed would be _____ and_____.

Answer: 5, 20

Are these two statements inside of the DO loop?_____

Answer: No

In this case, a "special exit" has been made from the loop without satisfying the requirements of the DO-statement parameters. True or false? _____

Answer: True

In this case, the variable ISAVE holds the value of I, which is the number of the loop last begun, also the index of the negative value found in ARRAY. True or false? _____

Answer: True

The main advantage of saving the value of I in ISAVE is that now the variable I may be used elsewhere without destroying its contents. True or false?_____

Answer: True

III. E. 27 In the following examples of coding, how many times will the DO loop be executed?

a.)

```
      DO 20 I=1,10
```

b.)

```
      JIG=1
      .
      .
      GO TO (50, 20) JIG
50    MIKE=JIG+6
      DO 30 I=JIG,MIKE
      .
      .
30    CONTINUE
20    (rest of program)
```

c.)

```
      DO 20 I=6,2
```

d.)

```
      .
      .
4     A=B**X**2
2     GO TO 1
10    DO 1 I=1,5
      GO TO 4
1     CONTINUE
30    (program continues)
```

e.)

```
      DO 95 COUNTER=1.0,10.0
```

Answers:

a.) 10

b.) 7

c.) 1

d.) None (since you go directly from statement 2 to statement 1, the CONTINUE statement is not treated as part of a DO loop).

e.) None (real variables and real constants are not allowed).

III. E. 28 A DO loop may contain within it one or more other DO loops. Many DO loops may be "nested" within a DO loop. How many DO loops are in each example here?

(A)

```
20  DO 95  I = . . .
    .
    .
    .
30  DO 95 J = . . .
    .
    .
    .
95  CONTINUE
```

(B)

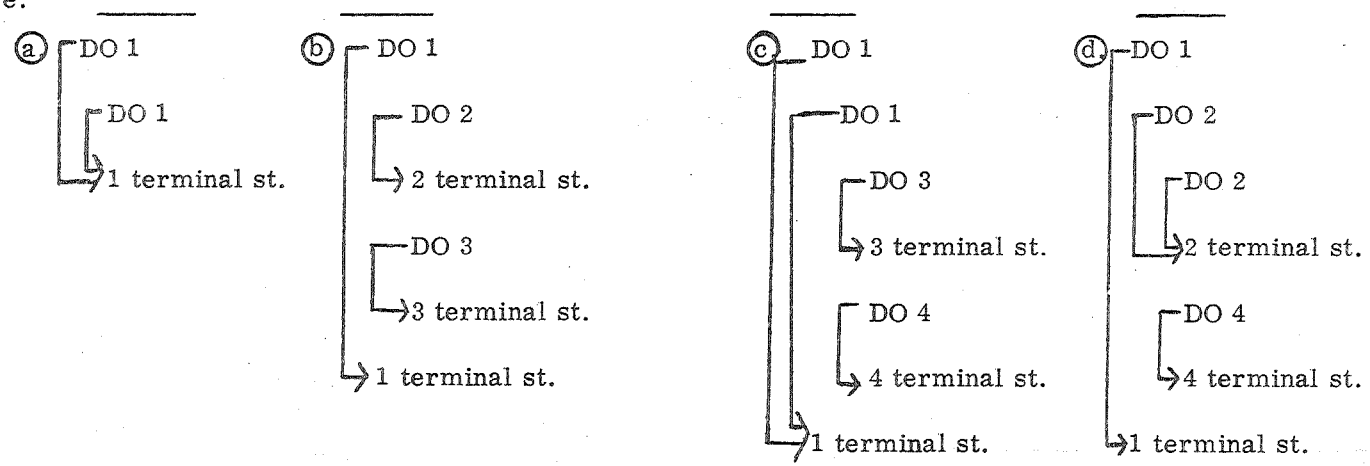
```
20  DO 95 I = . . .
    .
    .
40  DO 95 J = . . .
    .
    .
    .
65  DO 200 K = . . .
    .
    .
75  DO 200 L = . . .
    .
    .
200 CONTINUE
    .
    .
60  DO 300 M = . . .
    .
    .
250 DO 400 N = . . .
    .
    .
400 CONTINUE
300 CONTINUE
    .
    .
    .
70  DO 150 IAC = . . .
    .
    .
150 CONTINUE
    .
    .
    .
45  DO 50 IBC = . . .
    .
    .
50  CONTINUE
    .
    .
95  CONTINUE
```

Answer: (A) 2 loops

(B) 8 loops

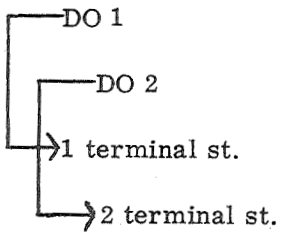
III. E. 29 Nested (inner) DO loops must be entirely contained by the outer DO loop; therefore, the terminal statement of the inner DO loop must be performed either before the terminal statement of the outer DO loop, or they must share a common terminal statement.

For each of the following examples of nested DO loops, decide if the terminal statements meet one of the two requirements described above.



Answer: (a) Yes
(b) Yes
(c) Yes
(d) Yes

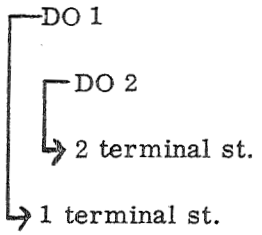
III. E. 30 A nested DO loop can be considered as a single entity by itself with the DO statement being the only valid entrance. Therefore, DO loops cannot overlap, unless one lies entirely within the other. If DO loops numbers 1 and 2 in the example here had an overlapping configuration, when DO loop number 2 was satisfied, the program would have to branch back into DO loop number 2 to execute the last statements of DO loop number 1. This is clearly not allowed.



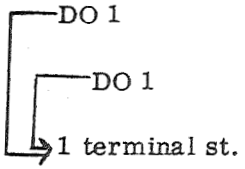
Redraw this diagram to two possible correct configurations for this example.

Answer:

a.)

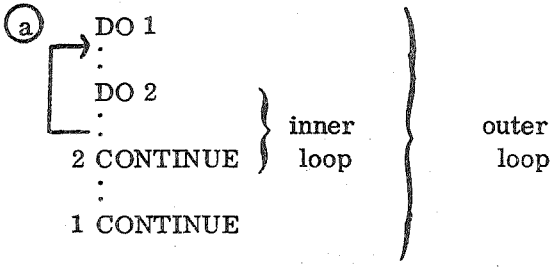


b.)

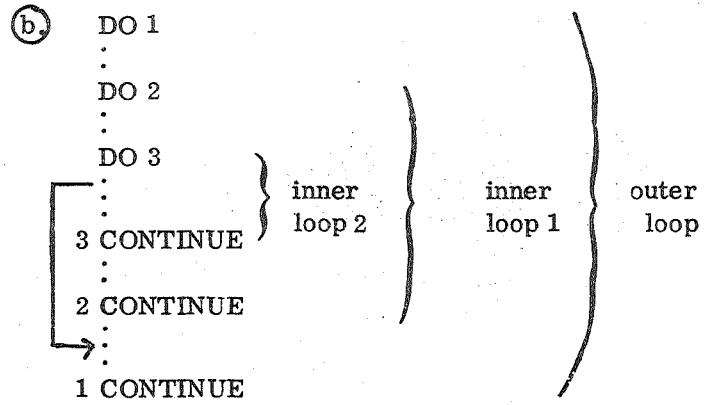


III. E. 31 Generally, transfer within nested DO loops may occur from inner loops to outer loops freely with, however, one exception. That is the transfer to the terminal statement of an outer loop. This is subject to the restriction discussed in section III. E. 18 of this manual.

In the converse situation, transfer from an outer to an inner loop can occur only down to one level of nesting, and then only through the DO statement of the inner loop. In other words, in the transfer from an outer to an inner loop, the DO statement of any nested DO loop cannot be skipped over.



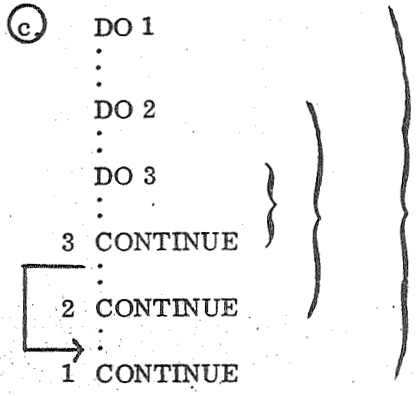
ⓐ permissible - a transfer is permitted from inner to outer loops.



ⓑ permissible - a transfer is permitted from inner to outer loops.

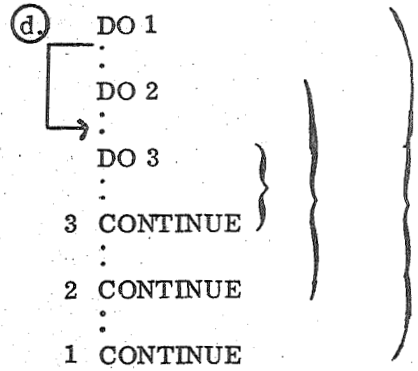
III. E. 31
(Cont.)

(c.)



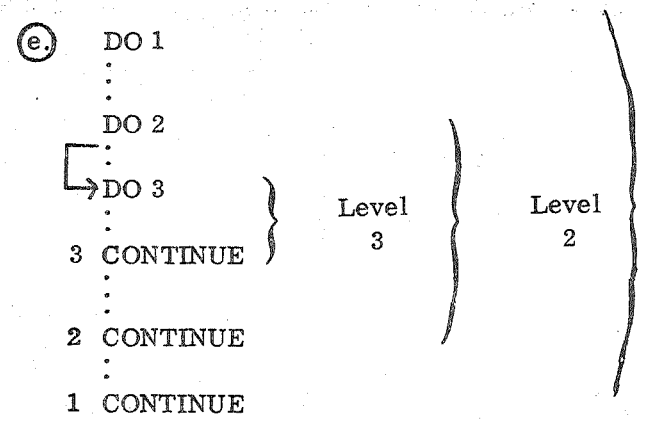
(c.) permissible - a transfer is permitted from inner to outer loops.

(d.)

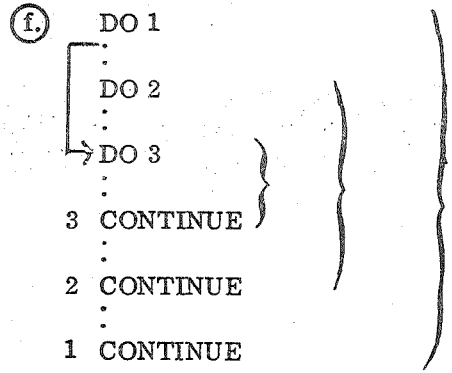


(d.) not permissible - a transfer from an outer loop to an inner loop, one level down, must be to the inner loop's DO statement. The "DO 2" statement, since it initiates a nested DO loop, cannot be skipped over in a transfer from an outer to an inner loop.

III. E. 31
(Cont.)



e. permissible - a transfer is permitted, one level down, from an outer to an inner loop through the DO statement of the inner loop.



f. not permissible - a transfer from an outer to an inner loop is not permitted more than one level down. The "DO 2" statement, since it initiates a nested DO loop, cannot be skipped over in a transfer from an outer to an inner loop.

III. E. 32 The power of the nested DO loop is well exemplified by the following sort program. This program will sort an array containing random numbers into one which is in numerically ascending order. This is accomplished in only eight FORTRAN statements no matter what the size of the array.

For illustrative purposes let's assume we wish to sort an array of five values. For clarity, and also because programs should be written this way, comments will be included in this sample sort program.

C for Comment — FORTRAN CODING FORM

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
		DIMENSION X(5)	
C		THIS OUTER DO LOOP DEFINES THE INDICES	
C		OF THE VALUES TO BE COMPARED	
		DO 100 N=1, 4	
		J=N+1	
C		THIS INNER DO LOOP COMPARES X(N)	
C		WITH X(K) AND INTERCHANGES	
C		VALUE WHEN X(K) IS SMALLER	
C		THAN X(N).	
		DO 100 K=J, 5	
C		THIS STATEMENT COMPARES	
		IF(X(N)-X(K)) 100, 100, 200	
C		THESE STATEMENTS INTERCHANGE	
200		TEMP=X(N)	
		X(N)=X(K)	
		X(K)=TEMP	
100		CONTINUE	

III. E. 32 (Cont.)

Let's follow this program using actual numbers. Assume we wish to sort the following array.

X(1) = 22.0
X(2) = 47.0
X(3) = 3.0
X(4) = 40.0
X(5) = 17.0

In this sort program the outer loop defines which values in the array are to be compared and interchanged by the inner loop. At each completion of the outer loop one value has been sorted out as the lowest of the values compared and properly positioned in the array.

After completion of the first pass through the outer loop, the smallest value of the array, in our sample case, X(3), will now be in position X(1) of the array, and the value formerly in X(1) will be in X(3).

Up to this point, this program has operated exactly like that in III. E. 19, which placed a value only in the first position of the array. But now we go a step farther, because after the second pass the next smallest value, X(5), will be in X(2) and so on to completion.

In the table on page 193, each line across represents the array after one pass through the inner loop. The logic of the program is followed in that the indices for both the outer and inner loops are indicated, the compare is shown, the interchange of values, when necessary, is indicated, and the current status of the array is given. The values in the array which are compared are indicated by being underlined. When the outer loop is completed, one more value has been sorted and positioned. The value is circled when this occurs.

III. E. 32 (Cont.)

Before you answer the questions about the table which follows, notice that the problem here is much like that in III. E. 19 except that here you do not stop after finding the smallest value, but next go on to find the second-smallest value, then the third-smallest value, and so on. Refer to the preceding coding and the table that follows, and answer these questions:

a.) In the present problem, the smallest value considered in any comparison always ends up in X(____), and the value compared to it is in X(____).

b.) During the first iteration of the outer loop, the value of N is _____; therefore, the smallest value found will be stored in X(____).

c.) During the second iteration of the outer loop, N = _____, and J = _____(which is also the initial value of K).

d.) This means that, during this second iteration of the outer loop, the first pair of values to be compared--X(N) and X(K)--will be X(____) and X(____).

e.) Therefore, during the second iteration of the outer loop, the smallest value will be stored in X(N); N = _____.

f.) Because the outer loop on each iteration increments the values of N and J, the low value put in X(N) during any iteration is left untouched during subsequent iterations and is not used in any future comparisons. True or false? _____

Answers:

a.) N, K

b.) 1, 1

c.) 2, 3

d.) 2, 3

e.) 2

f.) True

III. E. 32
(Cont.)

g.) After the first three iterations of the outer loop have sorted the three lowest values and stored them in X(1), X(2), and X(3), the fourth iteration compares the values in X(4) and X(5). Therefore, four iterations of the outer loop are sufficient to sort and store five values, and you can write: N=1, 4 as the outer loop's variable. True or false? _____

g.) True

h.) If you write the outer loop's variable as N=1, 4 to compare five values, how should the same variable be written to compare 50 values? To compare 100 values? _____ _____

h.) N=1, 49
N=1, 99

i.) In this program, the outer loop with its DO statement selects the values to be compared and identifies the storage location for the low value found; whereas the inner loop does the comparing and, if necessary, interchanging the values. True or false? _____

i.) True

j.) The coding at the beginning of this problem would work equally well if you were to delete from the outer loop the statement J=N+1 and, instead of having the inner loop set K equal to J, write DO 100 K=N+1, 5 as the inner DO statement. True or false?

_____

j.) False. A DO parameter must be a simple variable or constant, not an expression.

k.) Notice that the control variable for the outer loop has a maximum value of 4, whereas the inner loop's variable is always one integer higher (both initially where it is equal to N+1, and in its maximum value of 5.)

This permits the inner loop's control variable to select, for comparison with the array value selected by the outer loop's variable, the next value beyond the first, and then the next value beyond that one, and so on. In this way, the first selected value is never compared to itself. True or false? _____

k.) True

III. E. 32
(Cont.)

Outer Loop		Inner Loop	Compare	Interchange	Current Array Order				
N	J	K	X(N)-X(K)	If Positive	X(1)	X(2)	X(3)	X(4)	X(5)
					22.0	47.0	3.0	40.0	17.0
1	2	2	X(1)-X(2)	None	22.0	47.0	3.0	40.0	17.0
		3	X(1)-X(3)	22.0 & 3.0	3.0	47.0	22.0	40.0	17.0
		4	X(1)-X(4)	None	3.0	47.0	22.0	40.0	17.0
		5	X(1)-X(5)	None	3.0	47.0	22.0	40.0	17.0
					3.0	47.0	22.0	40.0	17.0
2	3	3	X(2)-X(3)	47.0 & 22.0	3.0	22.0	47.0	40.0	17.0
		4	X(2)-X(4)	None	3.0	22.0	47.0	40.0	17.0
		5	X(2)-X(5)	22.0 & 17.0	3.0	17.0	47.0	40.0	22.0
					3.0	17.0	47.0	40.0	22.0
3	4	4	X(3)-X(4)	47.0 & 40.0	3.0	17.0	40.0	47.0	22.0
		5	X(3)-X(5)	40.0 & 22.0	3.0	17.0	22.0	47.0	40.0
					3.0	17.0	22.0	47.0	40.0
4	5	5	X(4)-X(5)	47.0 & 40.0	3.0	17.0	22.0	40.0	47.0
				Final Array	3.0	17.0	22.0	40.0	47.0

III.E.33 Rewrite the program in III.E.32 to sort an array containing 100 values.

Answer:

```
      DIMENSION X(100)
      .
      .
      DO 100 N = 1, 99
      J = N+1
      DO 100 K=J, 100
      IF (X(N)-X(K)) 100, 100, 200
  200 TEMP = X(N)
      X(N) = X(K)
      X(K) = TEMP
  100 CONTINUE
```

III.E.34 Work exercise III.E in your workbook before starting section III.F.

III. F Other Control Statements

III. F. 1 There are a few other control statements, optional or required, that do not affect the program logic, but that the programmer may or must use for special purposes. These are the PAUSE, STOP, and END statements.

III. F. 2 The programmer may wish to have the execution of the program stop at a given place for any number of reasons, perhaps to see if an error may have been detected in the data, or he may wish to have the operator perform some operation on the computer console, etc. The PAUSE statement stops all program execution. When this occurs, the operator is given the option to either proceed or terminate the program. If the operator elects to continue the program, he enters this decision through the console and the program proceeds with the statement immediately following the PAUSE statement.

(NOTE: At some installations, a PAUSE statement is not allowed due to the fact that it stops computer operation. Check with your installation to see if the PAUSE is legitimate.)

III. F. 3 The STOP statement terminates program execution and causes control to be passed to the monitor system.

The STOP statement is effective only during execution of the program. True or false? _____

Answer: True

III. F. 4 Because there may be more than one STOP statement in the program, the STOP statements may be numbered so the programmer will know which STOP occurred. The STOP statement is written STOP N where N may be as many as five numbers which are restricted to the digits zero through seven. If no number is written after the word STOP, the number is assumed to be zero.

Which STOP statements are valid? Which are not, and why not?

a. STOP 1003 _____

b. STOP 1181 _____

c. STOP 2A _____

d. STOP, 101 _____

e. STOP 19 _____

f. STOP 007 _____

g. STOP _____

Answer:

a. valid

b. 8 is not allowed

c. letters are not allowed

d. comma not allowed

e. 9 is not allowed

f. valid

g. valid

III. F. 5 The END statement is mandatory, and it does exactly what it says, it defines the end of a program or of a subprogram. The final statement of a program or subprogram must be an END statement. When a source program consists of a main program plus one or more subprogram, then each of these parts must have an END statement as the last statement. The opening statement of another subprogram will follow immediately after this. For example:

```
PROGRAM
.
.
.
STOP

END
SUBROUTINE
.
.
.
END
SUBROUTINE
.
.
.
END
```

The word END may be followed on the same line by the name of your program, if you wish it for your own convenience, but the compiler will ignore the name.

The STOP statement terminates execution of a program.
The END statement terminates compilation of a program.
THIS IS IMPORTANT!!!

PART IV

BASIC INPUT/OUTPUT

IV.A Introduction

IV.A.1 All the programming skill and hardware technological advances available would be completely useless if there were no way for the computer to communicate its "answer" to the user. Furthermore, supplying data to the computer would be extremely tedious if the programmer needed to code each piece of information into the program. Fortunately, there are convenient ways for the user and the computer to exchange information. Such exchanges are commonly known as "input/output", or "I/O". The user "inputs" data and instructions, the computer "outputs" results and commentary.

IV.A.2 Cards into which data has been keypunched is a primary means of data input. There are 80 columns (spaces) per card and all 80 columns may be used for data. FORTRAN coding forms are used when writing data and, since all 80 card columns may be used, the 80 columns of the FORTRAN coding form may be used also.

Output to the printed page is limited to 132 characters per line.

IV.A.3 Several input and output devices may be utilized by a FORTRAN program, the most common being card reader, card punch, printer and magnetic tape. The computer uses the card reader to obtain information from punched cards; the card punch to output information onto cards; the printer to output information in printed form; the magnetic tape both for reading and for writing information.

Of the above four devices, _____ and _____ are input devices, while _____, _____, and _____ are output devices.

Answer: Card reader, magnetic tape - Card punch, printer, magnetic tape

IV.B Basic I/O Statements

IV.B.1 The general form of the most simple I/O statements is

```
READ n, L
PRINT n, L
```

where n is the statement number of the FORMAT statement.
and L is the list of variable names of the values to be input or output.

IV.B.2 A FORMAT statement is a FORTRAN statement which describes the format of the data to be input or output. (To be discussed in IV.D.)

Section IV.N will give full discussion of I/O statements. The definitions above should be sufficient for now.

IV.C I/O List

IV.C.1 This list tells which data items are to be transmitted, and in what order, i.e., from left to right.

The part of the I/O control statement that specifies which data items are to be transmitted is the_____.

Answer: list

IV.C.2 The length of the I/O list is restricted only by the limit of 19 continuation cards. Each item except the last is followed by a comma, and may be a simple or subscripted variable or an array name.

IV.C.2 (Cont.) Of the following five items, which are acceptable in an I/O list?

A. XYZ
B. IFLAG
C. 24CPS
D. ARRAY (1, 4)
E. INDEX (I)

Answer: A, B, D, E

IV.C.3 In the preceding question, item C is not acceptable in an I/O list because it is not acceptable as a FORTRAN variable name.

IV.C.4 If an item in an I/O list is to be subscripted, the subscript must be of the form (c * I ± d), where c and d are positive integer constants or zero, and I is a simple integer variable.

Which of the following subscripts are unacceptable in an I/O list?

Answer: B, E

A. (2 *KOUNT - 1)
B. (INDEX (3))
C. (I)
D. (I + 1)
E. (X)

IV.C.5 If you missed this one, remember that a variable used as a subscript may not be subscripted itself, and must be integer rather than real.

IV.C.6 Arrays to be input or output may be handled in several ways. If the array name appears in the I/O list without any subscripts, the entire array will be transmitted, according to the size specified in the DIMENSION statement, and having the first subscript varying first. For example:

```
      DIMENSION MATRIX (4, 4)
      READ 10, MATRIX
   10 FORMAT (4E16.8)
```

would read from cards 16 quantities according to the FORMAT statement which will be described in the next section. The quantities are stored in the array MATRIX, the first going into MATRIX (1, 1), the second into MATRIX (2, 1), the third into MATRIX (3, 1), the fourth into MATRIX (4, 1), the fifth into MATRIX (1, 2), the sixth into MATRIX (2, 2), etc., until finally the sixteenth would go into MATRIX (4, 4).

If ARRAY has been dimensioned (5, 5), how many quantities would be written by the statement PRINT 14, ARRAY? _____ Answer: 25

Which would be the first quantity written? _____ Answer: ARRAY (1, 1)

Which would be the last quantity written? _____ Answer: ARRAY (5, 5)

Which would be the twelfth quantity written? _____ Answer: ARRAY (2, 3)

IV.C.7 Remember, the first subscript varies most rapidly in this kind of array transmission.

IV.C.8 Arrays may also be specified in whole or in part in an I/O list by an implied DO loop, such as (A (I), I = 1, 10). If this implied DO loop appeared in an input list, ten items would be read into array A, beginning at A(1) and ending with A(10).

Nested implied DO loops such as

```
READ 30, ((HR (L, M), L = 1, 50), M = 1, 4)
```

would cause data to be input in the same order as is illustrated in the following nested DO loop. This is the DO loop that is implied.

```
      DO 100 M = 1, 4
      DO 100 L = 1, 50
      READ 30, HR (L, M)
   30 FORMAT (E 16.8)
  100 CONTINUE
```

How many quantities would be transmitted if the following implied DO loops appeared in an I/O list?

A. (X (KK), KK = 1, 6)_____

B. ((Y (I, J), J = 1, 4, 2), I = 1, 3) _____

Answer: A. 6
B. 6

IV. C. 8 (Cont.) Write the DO statements which are implied by the above implied DO loops.

A.

```
      READ 30, X(KK)
   30 FORMAT (E16.8)
  100 CONTINUE
```

Answer: DO 100 KK = 1, 6

B.

```
      READ 30, Y(I, J)
   30 FORMAT (E16.8)
  100 CONTINUE
```

Answer: DO 100 I = 1, 3
DO 100 J = 1, 4, 2

IV.C.9 Note that when an implied DO loop is used in an I/O list, a set of parentheses must enclose the array name and the DO-loop parameter specifications.

Each subscript on the variable requires DO-loop parameter specifications and a set of parentheses.

Suppose array CAT has been dimensioned (3, 3, 3). To read in all the elements of the array, varying the first subscript first, the second subscript second, and the third subscript last, what would the implied DO loop look like?

Answer:

(((CAT(I, J, K), I=1,3), J=1, 3), K=1, 3)

IV.C.10 Of course, in the answer to the above question, any acceptable variable names could be used in place of I, J and K. But it is important to note that for three subscripts there are parentheses enclosing three sets of DO-loop parameter specifications, and each right parenthesis except the last is followed by a comma.

IV.C.11 Example:

```
      DIMENSION RAT (6, 8, 7)
      READ 10, RAT (1, 3, 5), RAT (2, 3, 5), RAT (1, 4, 5), RAT(2, 4, 5), RAT (1, 3, 6), RAT(2, 3, 6),
     1 RAT (1, 4, 6), RAT (2, 4, 6)
   10 FORMAT (4 E16.8)
```

Rewrite the above input statement using an implied DO loop to accomplish the same result.

Answer:

READ 10, (((RAT(L, M, N), L=1, 2), M= 3, 4), N= 5, 6)

IV.C.12 Work Exercise IV.C in your workbook at this time.

IV.D Data Formatting

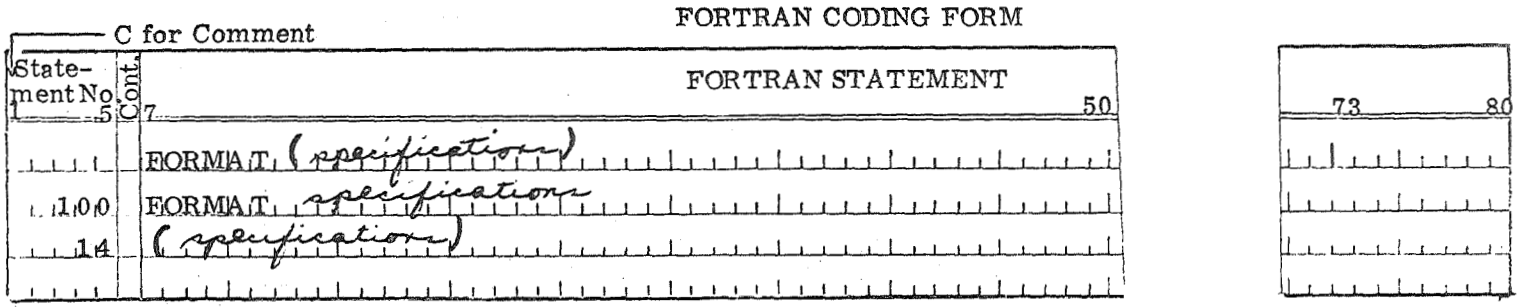
IV.D.1 Although most computers are binary, dealing with, say, ones and zeros only, most computer users prefer to think in terms of alphabetic and numeric characters. A term used to mean a mixture of alphabetic and numeric characters is alphanumeric. To satisfy both computer and user, there are several types of data conversions available in FORTRAN. If the programmer wants any of these conversions applied to the data transmitted by his list, he provides a FORMAT statement, specifying the precise type of conversion for each item in the list.

IV.D.2 The FORMAT declaration is a non-executable statement, may appear anywhere within the program, must have a statement number in columns 1-5, and be of the form

n FORMAT (specifications)

where "n" stands for the statement number, and "specifications" will be explained shortly.

What fault do you find with each of the following FORMAT forms?



A. _____
B. _____
C. _____

Answer: A. No statement number in 1-5
B. No parentheses around specifications
C. "FORMAT" missing

IV.D.3 The conversion types available are

Ew.d	Single precision real number with exponent
Fw.d	Single precision real number without exponent
Iw	Decimal integer
Dw.d	Double precision number with exponent (See VI.A)
Gw.d	Single precision real number with or without exponent (See VI.B)
Ow	Octal integer (See VI.I)
Aw	Alphanumeric (See VI.G)
Rw	Alphanumeric (See VI.H)
Lw	Logical (See VI.C)
nP	Scaling factor (See VI.D)

Complex data items require a pair of consecutive Ew.d or Fw.d conversion specifications for each complex value. In addition to the conversion specifications, there are four editing specifications:

wX	Spacing of card columns or printer position
wH	Heading and labelling
or	
...	Heading and annotating
/	Begin new record (one card or one printed line constitutes a record

In the foregoing, w and d are unsigned integer constants; w specifies the field width in number of character positions (card columns or printer spaces) and d specifies the number of digits following the decimal point within the field.

IV.D.4 Before considering the individual conversion specifications you may wish to see a pair of typical I/O statements together with their associated format specifications:

```
      READ 100, A, B, C
100   FORMAT (F5.2, E10.4, F7.5)
      PRINT 101, A, B, C
101   FORMAT (3E15.5)
```

Should this sequence occur in a program, the instructions would cause three quantities to be read by the card reader according to F5.2, E10.4, and F7.5 specifications in that order, and stored in A, B, and C, respectively. Then the printer would print on a new line of the output paper the three quantities A, B, and C, each in E15.5 format. Note that in each I/O statement a comma follows the format number.

IV.D.5 At time of output, the least significant digit of an output value is rounded up if the first digit to be dropped is 5 or greater than 5.

IV.D.6 Work Exercise IV.D in your workbook at this time.

IV.E Ew.d Conversion, Output

IV.E.1 For output, a number converted by the Ew.d specification will have the form

$$\overbrace{bbX.\underbrace{X\ldots X}_{d}E\pm ee}^{w} \qquad 0 \leq ee \leq 99$$

or

$$\overbrace{bbX.\underbrace{X\ldots X}_{d}\pm eee}^{w} \qquad 100 \leq eee \leq 322$$

where b indicates blank character(s), X the most significant digits of the integer and fractional parts, and ± eee or ±ee the power of ten by which the number is to be multiplied. The second blank will be replaced by - for negative values.

By the specification E12.5 output of the value +500.500 would appear on the printed page as

$$\overbrace{\underbrace{b}_{a}\underbrace{5}_{b}\underbrace{.}_{c}\underbrace{00500}_{d}\underbrace{E+02}_{e}}^{w}$$

w the field width (in this ex. w = 12)

a the space reserved for the sign of the number

b the most significant digit of the number

c the decimal point

d the number of decimal places in the specification designates d digits second in significance only to b.

e the exponent, whose field width is always 4

The minimum value w is d + 7. Therefore,

$$w \geq d + 7$$

A. For the E-type number -4.395E+02 how long is the d-field?

_____

B. By what number should -4.395 be multiplied to represent the actual quantity?

_____

Answer: A. 3

B. 10^2 or 100

IV.E.2 If d is zero or blank, the decimal points and all digits right of it are suppressed. For example, if A contains + 439.5, an E7.0 conversion specification would cause

bb4E+02

where each b is a single blank, to be printed or punched. The plus sign is suppressed.

If X contains +500.500, what will the output look like for each of the following conversion specifications?

A. E12.5_____
B. E14.5_____
C. E10.0_____

Answer: A. b5.00500E+02
B. bbb5.00500E+02
C. bbbbb5E+02

IV.E.3 The full field width, w, must be large enough to accommodate the output quantity, including the sign, significant digits, decimal point, sign of the exponent and exponent. Remember the rule is to figure $w \geq d + 7$.

IV.E.3 (Cont.)

What is the minimum value for w which would allow three significant figures of the quantity -57900.4 to be output by Ew.d format?____
Give the "E" representation of the number._____

Answer: 9
-5.79E+04

What is the minimum value for w which would allow six significant figures of the quantity 98765.4321 to be output by Ew.d format?____
Give the "E" representation of the number._____

Answer: 12
b9.87654E+04

IV.E.4

If you said eleven in answer to the last question, remember that space for the sign must always be provided even though the plus sign is suppressed when positive quantities are output.

IV.E.5

Suppose the w field is not large enough to accommodate the output value. When the computer encounters this situation during execution of a program, the output statement will print as many of the least significant digits as possible and precede these by an asterisk (*). Example:

```
      QBAR = 4790.2358
      PRINT 10, QBAR
10    FORMAT (1X, E10.5)
```

Output would be b*79024E+03.

IV.E.6

Up to 14 significant digits may be output by Ew.d type specifications.

IV.E.7

Work exercise IV.E in your workbook at this time.

IV. F Ew.d Conversion, Input

IV. F. 1 The total number of characters to be input is specified by w. The field may contain up to 15 significant digits, blanks are considered to be zeros. The input w-field may consist of one or more subfields, namely integer, fraction and exponent. A sign for the quantity is not required, but if absent it will be assumed to be positive. Examples of valid subfield combinations for Ew.d input are

A. -47.926E-10	Integer, fraction, exponent
B. -47.92	Integer, fraction
C. 326+3	Integer, exponent
D. +12345	Integer only
E. .12345	Fraction only
F. E+4	Exponent only (interpreted as zero)

All three subfields are required to be present in all Ew.d type input conversion specifications. True or false? _____

Answer: False

IV. F. 2 Note that the integer subfield may start with a sign, + or -, or a digit; the fraction subfield begins with a decimal point; the exponent subfield may begin with E, + or - . If the exponent subfield should begin with E, the + sign is optional between the letter and the string of digits following the letter. The value of the exponent subfield must be less than 322.

Which subfields are present in the following Ew.d type numbers?

A. -6228.4E5 _____

B. 25E-15 _____

C. 3.1415926 _____

Answer: A. Integer, fraction, exponent
B. Integer, exponent
C. Integer, fraction

IV.F.3 The number of digits in the field following the decimal point is specified by d. If d does not appear in the Ew.d specification, as in E9, it is assumed to be zero. Input values for Ew.d conversion may be written with or without a decimal point.

IV.F.4 If a decimal point does not appear in the input field, d in the format specification is used, and the quantity will be multiplied by 10^{-d} to position the decimal point. Thus a number is interpreted in this manner:

$$(\text{integer subfield}) \times 10^{(\text{exponent subfield})} \times 10^{-d}$$

Consider the examples

A. 326+3
B. +12345

where no decimal points are present. Example A, if converted by an E5.2 specification, would be $(326 \times 10^{3} \times 10^{-2})$ or 3260.0. Example B would be $(12345 \times 10^{0} \times 10^{-2})$ or 123.45 for a specification of E6.2.

A decimal point in the input field of w characters always takes precedence over d. If the input field contained 3260. or 326.+1 instead of 326+3 (Example A above), the result would still be 3260.0. (d = 2) in the E5.2 format is ignored.

With an E7.1 specification, what will be the values of the following input quantities?

A. 987.000 _____
B. 1.4E+22 _____
C. -20E+03 _____
D. .726+14 _____
E. 726+14 _____
F. bbbbE-6 _____

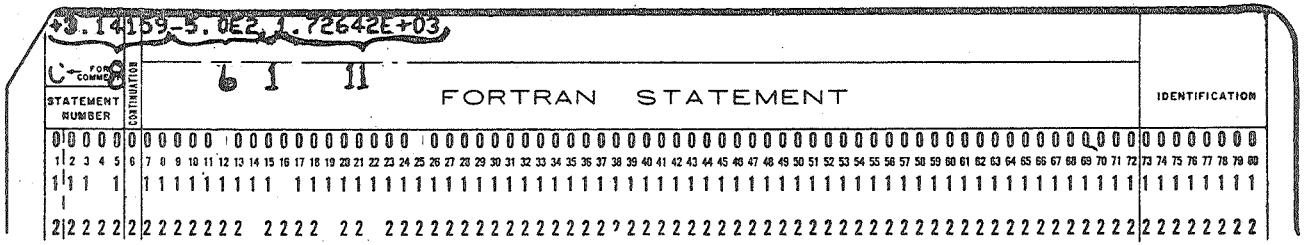
Answer: A. 987.000_{22}
B. 1.4×10^{22}
C. $(-20) \times 10^{-1} \times 10^{3} = -2000.0$
D. $.726 \times 10^{14}$
E. $726. \times 10^{13}$
F. 0.0

IV. F. 5 It is important that the w in an Ew.d input specification should be exactly the same length as the field containing the input number. If it is not, incorrect quantities will be input. Consider the following example:

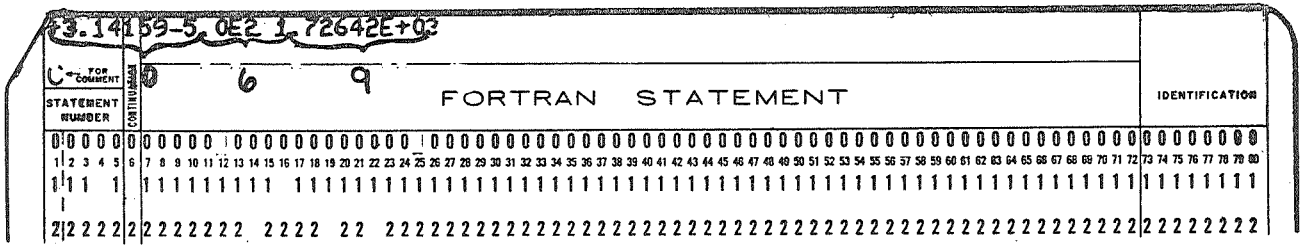
It is desired to input three quantities --

3.14159, -500.00 and 1726.42

They are punched in the first twenty-six columns of a card thus --



Input conversion specifications of E10.5, E6.1, E9.5 would interpret the three quantities



as .0000314159, 0.0, and .72642 .

IV.F.5 (Cont.) On input, some value read by an Ew.d specification will always be stored in the computer, and it is very important that the programmer insures that it is the correct value.

With an E7.3 specification, what value would be stored for each of the following input numbers:

A. bb246.3 _____
B. bb246E3 _____
C. 246.3bb _____
D. b246.E3 _____

Answer: A. 246.3
B. 246.0
C. 246.300
D. 246000.0

IV.F.6 Work Exercise IV.F in your workbook at this time.

IV.G Fw.d Output

IV.G.1 Now that you are familiar with the most complicated type of conversion, let us consider others. F-type output is similar to E-type: w sets the total field width and d specifies the number of digits to appear right of the decimal point. The difference between E- and F-type conversions is that in F-type the exponent subfield is not output, and is assumed to be zero.

IV.G.2 Restrictions pertaining to E-type conversion also apply to F-type:

IV.G.3 If d is zero or blank, the decimal point and numbers right of it are suppressed.

IV.G.4 If the field is too short to contain the output value, an * followed by as many of the least significant digits as the format can take care of will be output.

The output of values 267.26 and -13.54 by an F5.2 specification would be *7.26 and *3.54.

IV.G.5 The sign is output only when the number is negative, but a space must be allowed even though the sign of the number is positive.

IV.G.6 If the field is longer than the quantity to be output, the number will be right-adjusted in the field and extra spaces filled with blanks.

What would the output look like if each of the following quantities were output by its accompanying Fw.d specification?

A. 3.1415926, F5.2 _____
B. 500000.0, F7.0 _____
C. 2468.1357, F7.1 _____
D. -999.111, F8.3 _____
E. 12345.6789, F10.6 _____

Answer: A. b3.14
B. b500000
C. b2468.1
D. -999.111
E. *45.678900

IV.G.7 At this time work Exercises IV.G. in your workbook.

IV.H Fw.d Input

IV.H.1 Fw.d input specifications are similar to Ew.d input specifications. Again the difference is that the exponent subfield is omitted, and is assumed to be zero. All Ew.d restrictions apply to Fw.d.

IV.H.2 At this time you should work exercises IV.H in your workbook.

IV.I Iw, Input and Output

IV.L.1 I-type conversion is used to input or output integer quantities.

Which of the following variables should use I-type conversion if no type declaration has appeared in the program?_____

A. PI
B. HOURS
C. MINUTE
D. ALPHA
E. INDEX
F. N
G. RAD
H. KOUNT

Answer: C, E, F, H

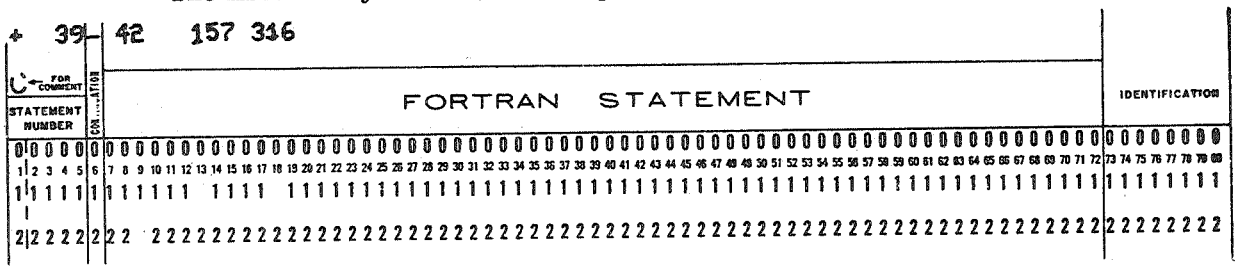
IV.I.2 In Iw specifications, w determines the width of the I/O field. On input if blanks appear within the field, they will be interpreted as zeros.

What value would be stored for 5bb65, read by an I5 specification?

Answer: 50065

IV.I.3 If a sign is included on an input quantity to be read by an I format, it must precede the first digit.

The first twenty columns of an input card contain



What values will be stored if the card is read by I5, I6, I4, I5 specifications? Answer: 39, -4200, 157, 3160

IV.I.4 On output if w is larger than necessary to accommodate the value to be transmitted, the number will appear right adjusted in the field, and preceded with blanks.

The quantity 987654 is to be output by an I8 format. How will it look? Answer: bb987654

IV.I.5 If the w field specified for output is too short to accommodate the value, the (w-1) least significant digits of the number will be output with an * in the left most position of the field. Space must be allowed for the sign of the number.

The quantity 987654 is to be output by an I4 specification. How will it look? Answer: *654

IV.I.6 Work Exercises IV.I.

IV.J Editing Specifications

IV.J.1 Many times computer programs make provision for punching out cards containing several items of data to be used as input for another program. Should the second program not need all the data items on a card, or not need each sequential card, it would be extremely tiresome to repunch or reassemble the input data deck. For this reason, and for spacing output data, editing specifications are available which permit skipping columns, advancing to the beginning of a card or printed line, and inserting commentary, titles, labels, etc.

IV. J. 2 wX specification indicates that the next w columns of card or print positions of printed line are to be skipped over. For example, if two integers, 5 and 10, are to be printed, specifications I5, I5 might be used and the output would be

bbbb5bbb10

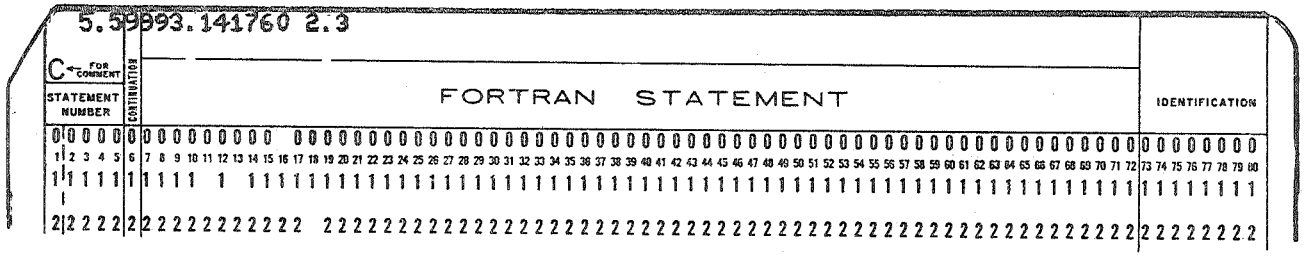
I5 I5

The same output format could be achieved with specifications 4X, I1, 3X, I2:

bbbb5bbb10

4X I1 3X I2

If the input list is A, B, C, what values would be read in from this card for A, B, and C:



for format specifications F5.1, 3X, F4.2, 4X, F4.1? A = _____

B = _____

C = _____

Answer: A = 5.5
B = 3.14
C = 2.3

IV.J.3 In a FORMAT statement, it is entirely optional whether the customary comma follows a wX specification or not. For example, the specifications

n FORMAT (I2, 6X, E10.2) (n represents the FORMAT statement number)

and

n FORMAT (I2, 6XE10.2)

are equivalent.

IV.J.4 Work Exercises IV.J in the workbook.

IV.K New Record

IV.K.1 A "record" in FORTRAN input/output operations consists of one punched card, one printed line, or one magnetic tape record. When it is desired to begin a new record in the course of data transmission, a slash (/) in the list of format specifications will terminate operations on the current record, and subsequent operations will be on a new record. For example, to read in three quantities by input specifications E10.2, F10.6/I5 it would be necessary to have one card with two real quantities followed by one card with one integer quantity. To print out the three quantities by output specifications 10X, E15.5, 10X, E15.5/20X15 would mean printing two lines, the first one containing the two real quantities and the second the integer quantity.

How many cards would be required to read six items by specifications

n FORMAT (A10/I5, E15.5, I5, E15.5/10X F10.2) Answer: 3

IV.K.2 The slash in a list of format specifications has slightly different effects for input and output. In input transmission, the slash causes reading of current card to cease, and reading on a new card to begin. Consider the specifications

A. n FORMAT (I4/I5)
B. n FORMAT (I4, I5/)

In order to read in two integer quantities by either of these sets of specifications, two cards are required. In case A , one value would be read from the first card, the slash causes reading of the first card to stop, and the second value would be read from the second card, In case B, two values would be read from the first card, the slash causes reading of that card to stop, and reading of a second card to begin, even though no more quantities are to be read. Therefore, set B will cause one card to be skipped after reading the two data items.

To input five items, how many cards would be required for specifications

A. n FORMAT (I5/I10/F10.5/E15.6/E15.6) _____ Answer: A. 5
B. n FORMAT (I5, I10/F10.5, E15.6, E15.6) _____ B. 2
C. n FORMAT (I5, I10/F10.5, E15.6, E15.6/) _____ C. 3
D. n FORMAT (I5, I10//F10.5, E15.6, E15.6) _____ D. 3

IV.K.3 In output specifications the slash indicates "terminate and begin." Consider again the specifications

A. n FORMAT (I4/I5)

B. n FORMAT (I4, I5/)

In order to print two integer quantities by set A, two lines are required. The first line will contain the first quantity; the slash terminates printing of the first line; the I5 specification causes printing of the second quantity on the next line. In case B two lines will be printed: the two quantities will be printed on one line; the slash terminates printing on that line, and causes a line of blanks to be printed on the second line.

How many lines are required to print five quantities by specifications

A. n FORMAT (4X, I5, I10/4X, E15.5, F10.5, E15.5) _____ Answer: A. 2

B. n FORMAT (I5, I10, E15.5, F10.5, E15.5) _____ B. 1

C. n FORMAT (I5, 4X, I10, E15.5, F10.5, E15.5/) _____ C. 2

D. n FORMAT (/I5, I10/E15.5, F10.5, E15.5/) _____ D. 4

IV.K.4 In part D., the first slash terminates printing of a blank line, the second slash terminates printing of a line with two integer quantities, the third slash terminates printing of a line with three real quantities and prints a blank line--a total of four lines.

IV.K.5 When consecutive slashes appear in a list of format specifications, they need not be separated by a comma. The effect of consecutive slashes is to skip one or more cards or lines. During input, N consecutive slashes embedded within format specifications will cause N-1 cards to be skipped; N consecutive slashes preceding or terminating specifications will cause N cards to be skipped.

If two quantities are to be input by the following format specifications, how many cards will be skipped in each case?

A. n FORMAT (I4///I4) _____
B. n FORMAT (///I4, I4) _____
C. n FORMAT (I4, I4///) _____

Answer: A. 2
B. 3
C. 3

IV.K.6 During output, N consecutive slashes preceding or terminating format specifications will cause N lines to be skipped; N consecutive slashes embedded within format specifications will cause N-1 lines to be skipped.

If two quantities are to be printed by the following specifications, how many lines will be skipped in each case?

A. n FORMAT (I4///I4) _____
B. n FORMAT (///I4) _____
C. n FORMAT (I4///) _____

Answer: A. 2
B. 3
C. 3

IV.K.7 Go to your workbook and do Exercises IV.K.

IV. L wH, Output and Input

IV. L. 1 The wH output specification is used to insert w Hollerith characters (6-bit alphanumeric characters) into the output record. The w characters to be inserted follow immediately after the H in the format specification. For example, if the specifications

n FORMAT (10X19HTHISbISbANbEXAMPLE.)

are used to output one line, the line would be

bbbbbbbbbbTHISbISbANbEXAMPLE.bbb - - -
10X 19H

In the wH output specification, w must be equal to or less than 132 since there are only 132 print positions on most printers. A comma following the w characters is optional.

What would the printed line look like if it were output according to

n FORMAT (4X,10HbVELOCITYb,3X,4HTIME)

Answer:

bbbbbVELOCITYbbbbTIMEbbb - -

IV. L. 2 wH, input specification causes w Hollerith characters to be read into the specified FORMAT statement. For example, if a card were read by

n FORMAT (15Hbbbbbbbbbbbbbbb)

the first 15 columns of the card would be read into the 15 blank spaces following the H of the specification. This same FORMAT statement, containing its new Hollerith characters in place of blanks, could then be used to output a title, label, or commentary.

If a card were read by specifications n FORMAT (17Hbbbbbbbbbbbbbbbbb), and the card contained NOVEMBERb24,b1968 in the first seventeen columns, could the same format later be used to print a date on output?

Answer: Yes

IV.L.3 An option to wH output is simply to enclose the desired Hollerith type data with (*------*). If we apply the (*------*) option to the first example in IV.L.1 the FORMAT statement would be

n FORMAT (10X*THISbISbANbEXAMPLE.*)

IV.M. Repeated Specifications

IV.M.1 Any conversion specification mentioned so far may be repeated in a FORMAT statement by a convenient notation: immediately precede the specification with the integer constant which is the number of times the specification should appear in the FORMAT statement. For example, to read in four integers from one card, the specification might be

n FORMAT (4I5)

which is equivalent to

n FORMAT (I5, I5, I5, I5)

What is the equivalent form of

n FORMAT (E10.3, E10.3)

Answer: n FORMAT (2E10.3)

IV.N Input/Output Statements

IV.N.1 Now that you know how to specify which data items are to be input or output from a list, and in what form they are to be transmitted by means of FORMAT conversion specifications, there remains one further process to be mastered: namely, writing the input/output statements. As you know, the input operation is reading -- punched cards or magnetic tape; the output operations are writing -- magnetic tape, printing -- on paper, and punching -- cards. Each operation makes reference to a piece of equipment -- a card reader or punch, a magnetic tape unit or the printer. So input/output statements must include two pieces of information in addition to list and format reference number. These two pieces of information are the kind of operation desired, and the piece of equipment to be used. A typical I/O statement has the form

OP n, L

or

OP (i, n) L

where OP specifies the desired operation, i is an integer constant or variable designating the I/O equipment, n is a FORMAT statement number, and L represents the list.

How many elements make up the second form of a typical input/output statement as shown above? _____

Answer: 4

These elements are _____, I/O unit number, _____, and list.

Answer: operation

FORMAT number

IV.N.2 The read-card operation is stated

READ n, L

This statement will cause one or more cards to be read from the card reader according to format, n, and list, L.

Which of the four I/O statement elements is missing from the preceding READ statement? _____

Answer: I/O unit number

IV.N.3 To read cards it is unnecessary to specify a unit number, since only one unit -- the card reader -- can read the cards. As an example,

```
      READ 100, A, B, C
100   FORMAT (E15.6)
```

will cause three cards to be read by E15.6 format. The value from the first card will be stored in A, the value from the second card will be stored in B, and the value from the third card will be stored in C.

How many cards will be read by

```
      READ 99. X, Y, Z
99    FORMAT (3E15.6)
```

_____ Answer: 1

IV.N.4 By way of review, remember that reading (or writing) will continue as long as the list specifies. If the format statement "runs out" before the list, it will repeat according to the rules mentioned under repeated specifications.

Write the statement to read from punched cards according to

```
10  FORMAT (20I4)
```

all the elements of array KTABLE, which is a one-dimensional array, dimensioned 25. _____

Answer: READ 10, KTABLE or
READ 10, (KTABLE(I), I=1, 25)

IV.N.5 If formatted data is to be read from a magnetic tape unit, the READ statement must specify which unit. Therefore, the READ statement will be

READ (i, n) L

Notice that when the unit number and format number are enclosed in parentheses, they are separated from each other by a comma; but the right parenthesis is not separated from the list by a comma.

Write the statement to read three real values for X(5), Y(10) and ALPHA, in that order, from magnetic tape unit 4, according to

500 FORMAT (3X2E20.10, 5XF5.1)

Answer:
READ(4, 500) X(5), Y(10), ALPHA

IV.N.6 An equivalent way of writing

READ (i,n) L

is to write

READ INPUT TAPE (i, n) L

What is the equivalent form of

READ INPUT TAPE (5, 50) MATRIX ?

Answer: READ (5,50) MATRIX

IV.N.7 As you might have suspected, the output statements to print, punch or write formatted data onto magnetic tape are

PRINT n, L

PUNCH n, L

WRITE (i, n) L

WRITE OUTPUT TAPE (i, n) L

} equivalent forms

Write a statement and accompanying format to print four positive integer quantities, each with a maximum value of 999 on one line, separated from each other by at least five spaces. The quantities are named I, J, K, and L.

Answer:

PRINT 100, I, J, K, L

100 FORMAT (5XI4, 5XI4, 5XI4, 5XI4)

or 100 FORMAT (4I9)

IV.N.8 In the preceding answer, of course, you could have used any number for the format statement, so long as it was the same number used in the PRINT statement.

IV.N.9

You may have wondered why the printed line was begun with five spaces instead of with the integer I. Actually, it was unnecessary to leave five spaces at the beginning of the line. However, the very first space of the line has a special function in controlling the spacing of lines on the output paper. The first character on the output line is never actually printed, and acts as a carriage control, with these results:

For first character	0,	result: double-space before printing
	1,	go-to new page before printing
	+,	spacing is suppressed after printing (next print record continues on same line)
any other character or blank,		single-space before printing

To print ten elements each from two one-dimensional arrays, KOUNT and ANGLE, with one element from KOUNT and the corresponding element from ANGLE on the same line, separated by three blanks, and double-spacing between lines, the coding might be

```
      PRINT 976, (KOUNT (I), ANGLE (I), I=1,10)
  976 FORMAT (1H0I25, 3X, E20.10)
```

If KOUNT and ANGLE are dimensioned 10 each, an equivalent way of writing the above print statment is

```
      PRINT 976, KOUNT, ANGLE
```

True or false? _____

Answer: False

IV.N.10 Remember, when a dimensioned array is used in an I/O list without a subscript, the entire array is transmitted before the next item of the list is transmitted. Therefore, the result of

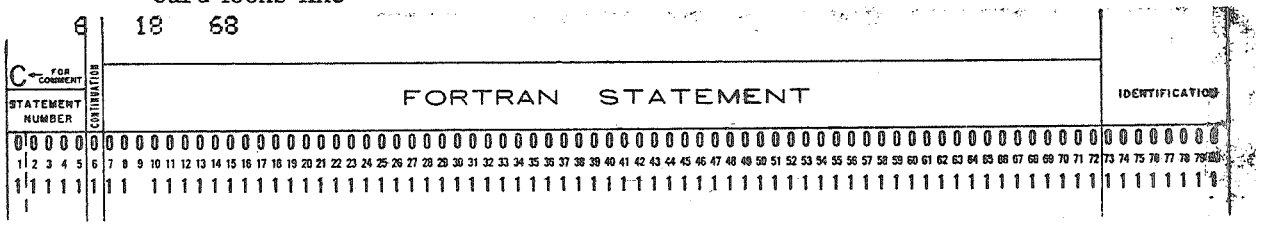
```
      PRINT 976, KOUNT, ANGLE
  976 FORMAT (1H0 I25, 3X, E20.10)
```

would be

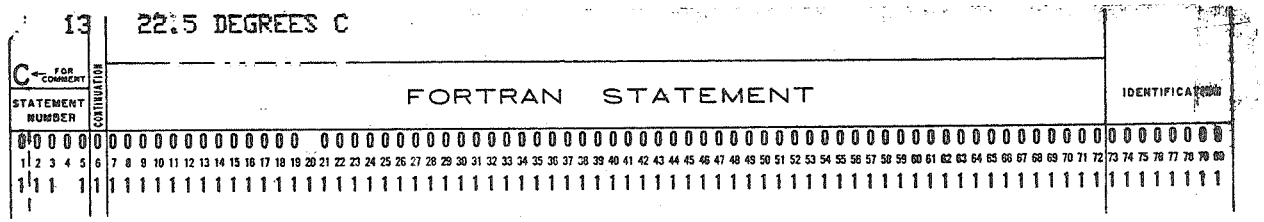
line 1	KOUNT (1)	KOUNT (2)
line 2	KOUNT (3)	KOUNT (4)
line 3	KOUNT (5)	KOUNT (6)
line 4	KOUNT (7)	KOUNT (8)
line 5	KOUNT (9)	KOUNT (10)
line 6	ANGLE (1)	ANGLE (2)
line 7	ANGLE (3)	ANGLE (4)
line 8	ANGLE (5)	ANGLE (6)
line 9	ANGLE (7)	ANGLE (8)
line 10	ANGLE (9)	ANGLE (10)

This is bad enough, but there is worse trouble: every item from KOUNT which appears in the right-hand column would be converted as though it were a real number, which it isn't, and every item from ANGLE in the left-hand column would be converted as though it were an integer. Recalling the computer forms of real and integer numbers, with and without characteristics, you might imagine the "real" KOUNT items would all be printed as zero, and the "integer" ANGLE items would all be very large integers indeed!

IV.N.11 As a final example, let us look at a little coding to do the following: read from a punched card a date, then from twenty-four more cards each an hour and a temperature. The first card looks like



The remaining cards are similar to this one



(The rightmost digit of the hour will always be in column five, the temperature will always have the decimal punched in column 11, and one fractional digit.) Then, starting on a new page of output, let us print the date as a page heading, followed by the hours and corresponding temperatures in chronological order.

```
      DIMENSION TEMPS(24), IDATE(3)                              (1)
      READ 100, IDATE                                            (2)
100   FORMAT (3I5)                                               (3)
      DO 1 I = 1, 24                                             (4)
  1   READ 101, K, TEMPS(K)                                      (5)
101   FORMAT (I5, F7.1)                                          (6)
      PRINT 102, IDATE                                           (7)
102   FORMAT (1H1 I2, 1H/ I2, 1H/ I2)                            (8)
      PRINT 103                                                  (9)
103   FORMAT (/1X 4H HOUR 2X 22HTEMPERATURE IN DEGREES)          (10)
      PRINT 104, (I, TEMPS(I), I = 1, 24)                        (11)
104   FORMAT (1H0, 1X, I2, 7X, F7.1)                             (12)
```

IV.N.12 Several things to note in the above twelve lines of coding are:

(1) IDATE must be dimensioned at least 3, since it will require 3 computer words to contain enough display code (or alphanumeric) characters to write the date in full.

(2) Since IDATE is not subscripted in the read statement, the entire array -- 3 words in the case -- will be read in.

(3) Although FORMAT statements may appear any place in a program, it is convenient for debugging and referencing purposes to keep them near the I/O statement using them.

(5) An item may be input in one statement and then used in the same statement for reading other items.

(8) In the format, 1H1 advances the paper to a new page before printing the date.

(9) Note there is no list with this statement -- this implies that the format will consist of editing and/or labelling specifications only.

(10) The initial slash skips a line before printing, 1X insures that the carriage control will indicate single spacing; an alternate way of achieving this spacing would be to replace "/1X" with "1Hb".

(8)-(10) Formats 102 and 103 could have been combined:

```
102 FORMAT (1H1 I2, 1H/ I2, 1H/ I2 //  1X4HHour 22HTEMPERATURE IN DEGREES)
```

IV.N.12 (Cont.)

In this case lines (9) and (10) would be unnecessary. Since an output line can have up to 132 print columns, the above output could have been centered on the output page by following the output format carriage control characters with 50X.

(10) An alternate way by which this format statement could have been written is as follows:

```
103  FORMAT (/* HOUR  TEMPERATURE IN DEGREES*)
```

Note that it is not necessary to count characters when using this method.

IV.O DATA Statements

Variables may be assigned constant values at load time by use of the DATA statement.

IV.O.1

One form of the data statement is

$$\text{DATA } d_1, \ldots, d_n/a_1, \ldots, a_n/, d_1, \ldots, d_n/a_1, \ldots, a_n/, \ldots$$

where d_i are identifiers representing simple variables, array names, or variables with integer constant subscripts or integer variable subscripts (implied DO-loop notation).

a_i are signed or unsigned constants; i.e., integers, real numbers, complex numbers, Hollerith information.

The variable appearing in the d_i field must not be blank common or numbered common variables. COMMON will be discussed in Part V.

IV.O.2 The simplest example of a DATA statement is the statement

```
DATA  PI/3.14159265/
```

This is equivalent to the arithmetic statement

```
PI = 3.14159265
```

except that in the DATA statement the value is assigned to the variable at the time the program is compiled. The latter case is a simple form of an arithmetic statement which is performed during execution of the program.

Little benefit is obtained by using a DATA statement in the above example. The most advantage occurs when assigning values to arrays. When a DATA statement refers to an array, the array must be properly dimensioned and the DIMENSION statement must appear prior to the DATA statement.

An array must be dimensioned prior to assigning it values by use of a DATA statement. True or false? _____

Answer: True

IV.O.3 The type of the constant stored is determined by the structure of the constant rather than by the variable type in the statement. In DATA A/2/, an integer 2 replaces A, not a real 2 as might be expected from the form of the name A.

Constants in data statements are not converted to match the type of the variable to which they are assigned. True or false? _____

Answer: True

IV.O.4 A single-subscript, implied DO loop may be used for storing constant values in arrays.

Example:

```
DIMENSION TABLE(5)
DATA (TABLE(I), I=1,5)/7., 6., 5., 4., 3./
will assign 7.0 to TABLE(1), 6.0 to TABLE(2) etc.
```

The value _____ is assigned to TABLE(5). Answer: 3.0

IV.O.5 When the number of list elements exceeds the range of the implied DO, the excess list elements are not stored, and a diagnostic is issued.

In the use of the statements

```
DIMENSION TABLE(3)
DATA (TABLE(I), I=1, 3)/7., 6., 5., 4., 3./
```

the values _____ and _____ are discarded. Answer: 4.0, 3.0

IV.O.6 An alternate form for storing constants into an array eliminates the implied DO.

For example:

```
DIMENSION B(5)
DATA B/4., 3.2, O000077, 0, 5.9/
```

will assign 4.0 to B(1), 3.2 to B(2), etc. This is similar to input or output of values into or from dimensioned arrays by entering the name of the array without subscripts in the I/O list.

IV.O.7 Another form of the DATA statement is

DATA $d_1, \ldots, d_n/a_1, k*a_2, \ldots, a_n/$

where k is an integer constant repetition factor that causes the constant following the asterisk to be repeated k times.

For example, the statement

```
DATA FAT,CAT,SAT,RAT/4.0,2*5.2,2.1/
```

is equivalent to the statement

```
DATA FAT,CAT,SAT,RAT/4.0,5.2,5.2,2.1/
```

The statement

```
DIMENSION AMAT(10)
DATA AMAT/10*3.2/
```

will assign the value _____ to each of the ten elements of the array. Answer: 3.2

IV.O.8 Variable Hollerith or alphanumeric information is frequently needed in a program. This may be defined by a DATA statement with a maximum of 10 characters to a word.

Example:

```
DIMENSION MESAGE (2)
DATA MESAGE/10HTHISbISbAN, 8HbEXAMPLE/
```

Array MESAGE: THIS IS AN EXAMPLE

IV.O.9 An illustration of the use of variable Hollerith data may be realized by considering that if the answer to an equation, IANS, is <0 we wish to print NO; if the answer is >0, we wish to print YES; and if the answer = 0, we want to print MAYBE. The Aw I/O specification used to print the message will be discussed in Part VI.

The following coding will illustrate this:

FORTRAN CODING FORM

C for Comment

State-ment No. 1–5	Cont.	FORTRAN STATEMENT 7–50	73–80
		DIMENSION MESAGE(3)	
		DATA(MESAGE(I),I=1,3)/5HbbNOb,5HMAYBE,	
	1	5HbYESb/	
		.	
		.	
		.	
		IF(IANS)10,20,30	
10		K=1	
		GO TO 100	
20		K=2	
		GO TO 100	
30		K=3	
100		PRINT 1000, MESAGE(K)	
1000		FORMAT (1X,A5)	
		.	
		.	
		.	

IV.O.9 (Cont.) What would be stored in core for the following DATA statements?

A. DIMENSION TABLE(3)
DATA (TABLE(I), I=1,3)/1.0,2.0,3.0,4.0,5.0/

Answer: 1.0, 2.0, 3.0

B. DIMENSION TABLE(5)
DATA (TABLE(I), I=1,5)/5*4.0/

Answer: 4.0, 4.0, 4.0, 4.0, 4.

C. DATA MESAGE/11HTHISbISbBAD/

Answer: THISbISbBA

IV.O.10 If you missed the last problem, remember that only ten characters may be stored in one computer word.

IV.O.11 Another form of the DATA statement may be used. This form is

$$\text{DATA } (d_1 = \text{list}_1)\ ,\ (d_2 = \text{list}_2),\ \ldots\ ,\ (d_n = \text{list}_n)$$

where d_1, d_2, etc. may be a simple variable or a subscripted variable with or without subscripts. The list is a set of constants as before except that repetition is indicated by a set of parentheses instead of an asterisk.

For example,

FORTRAN CODING FORM

C for Comment

State-ment No.	Cont.	FORTRAN STATEMENT (7–50)	73–80
		DIMENSION GIB(10),Y(20),Z(4,4)	
		DATA (GIB=1.,2.0,3.0,7(4.3))	
		DATA (X=3.14159)	
		DATA ((Y(I),I=1,6)=3.0,4.0,1.5,2(0.0),1.0)	
		DATA (Z(4,4)=10.0)	

are all acceptable. Notice that as in the first form, an array name without subscripts implies the entire array.

IV.O.12 Now you are ready to work the final Section IV exercises in your workbook.

V.A Statement Functions, Function and Subroutine Subprograms, COMMON

V.A.1 In previous sections you have learned to write fairly complicated programs which included the input of certain data, the required calculations, and the output of the results. In theory, these tools previously presented will solve almost any numerical scientific problem. In practice, however, the series expansion of a few hundred trigonometric and exponential functions in some problem would probably cause the programmer to decide on a different career. To prevent such a mass exodus, FORTRAN provides the programmer with methods for handling often-repeated functions, namely, statement functions, function and subroutine subprograms.

V.A.2 Statement Function

V.A.3 There are two ways to write a function. The first method is used when the function can be expressed with one FORTRAN statement. This is called a statement function. A statement function is defined in the program where it is to be used.

The form of a statement function is:

Name $(p_1, p_2, \ldots, p_n) = E$

p_i are formal parameters (or arguments) and must be simple variables.

There must be at least one and not more than 60 parameters.

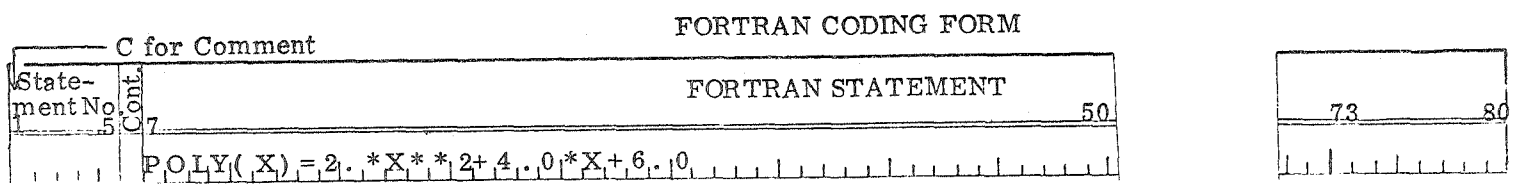
E is an expression, arithmetic or logical, which is a function of the arguments $p_1, p_2, \ldots, p_n$.

V.A.3 (Cont.) A statement function must always be defined in the program in which it is used. True or false? _____

Answer: True

V.A.4 Statement functions are named according to the rules which apply in the naming of variables.

The statement function below may appear in a program.



The name of this statement function is _____.

Answer: POLY

It has one formal parameter, _____.

Answer: X

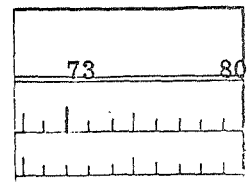
V.A.5 A statement function must preceed all executable statements in the program. It must follow all DIMENSION, type, COMMON, and EQUIVALENCE statements which pertain to variables used in the function definition. COMMON and EQUIVALENCE will be discussed later in the chapter.

V.A.5 (Cont.) What is wrong with the following sequence?

FORTRAN CODING FORM

C for Comment

State-ment No. 1–5	Cont.	FORTRAN STATEMENT 7–50
		AFUN(J,K) = L(3) * J - K
		DIMENSION L(10)



Answer: L(3) appears in statement function before it was dimensioned.

V.A.6 A statement function is referenced by placing the statement function name in an arithmetic or logical expression. For example,

```
Y = POLY(X1)*Z + 1.5
```

references the statement function POLY (defined in V.A.4). This statement causes the expression, 2.0*X**2+4.0*X+6.0, to be evaluated using the value of X1 in place of X. The result is then multiplied by the value of Z, added to 1.5 and assigned to Y.

The result of POLY(X1) is _____ since the type of a function is covered by the same rules as the type of a variable.

Answer: real

V.A.7 In the statement

$$Y = POLY(X1) * Z + 1.5$$

the value X1 is called the actual parameter of the statement function POLY. This means that X1 is the actual value which is used in the evaluation.

The parameters used in defining a statement function are called _____ parameters.

Answer: formal

The parameters used when referencing a statement function are called _____ parameters.

Answer: actual

V.A.8 When a statement function is referenced, it often looks exactly like a subscripted variable. In order to distinguish between the two, it is essential that all subscripted variables be dimensioned and that a function name never be dimensioned.

What is the value of Y after statements 10, 20 and 30 have been executed? _____

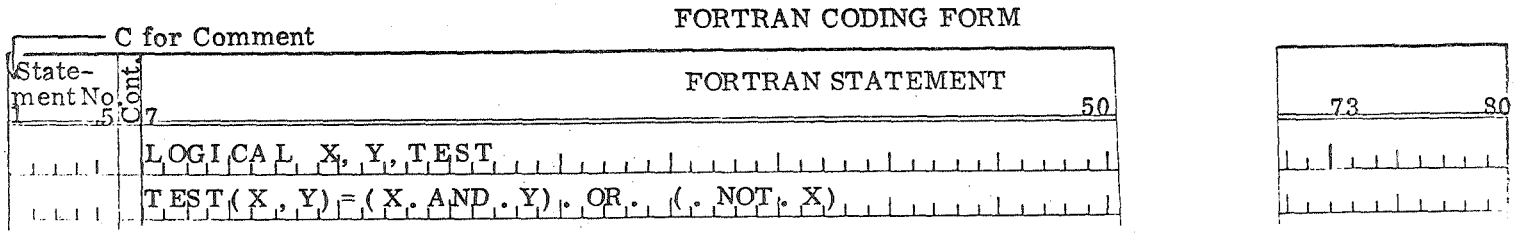
FORTRAN CODING FORM

C for Comment

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)
		POLY(X)=2.0*X**2+4.0*X+6.0
10		Z=1.5
20		R=2.0
30		Y=POLY(R)*Z+1.5

73 80

V.A.9 A logical expression may also be defined by a function statement. An example follows:



The logical variables X, and Y are the _____ parameters of the statement function.

Answer: formal

A reference to the function will give a .TRUE. or .FALSE. result depending on the values of the _____ parameters.

Answer: actual

V.A.10 A statement function name must not appear in a DIMENSION, EQUIVALENCE, COMMON, or EXTERNAL statement. EXTERNAL will be discussed later in the chapter.

If a statement function name were allowed to appear in a DIMENSION statement, it would look exactly like a _____ variable.

Answer: subscripted

V.A.11 A statement function may reference library functions, FORTRAN functions, other statement functions, function subprograms, but not itself. Library functions, etc. will be discussed later in this chapter.

V.A.11 (Cont.) For example:

FORTRAN CODING FORM

C for Comment

Statement No. 1–5	Cont. 6	FORTRAN STATEMENT 7–50	73–80
		COMPLEX Z	
		Z(X,Y) = (1.0,0.)*EXP(X)*COS(Y)+(0.0,1.0)	
		FIRST(X) = A**2+B	
		XSQ(X) = X**2	
		SECONDF(X) = FIRST(X) / XSQ(X)	

The library functions EXP and COS are used in defining the statement function Z. The statement function SECONDF uses other _____ _____ in its definition.

Answer: statement functions

V.A.12 Work exercise V.A in your workbook at this time.

V.B Subprograms

V.B.1 A subprogram is a section of preprogrammed coding which is designed to accomplish a very specific objective, such as to evaluate a sine, perform an interpolation, etc.

FORTRAN provides a group of commonly used subprograms, notably trigonometric and exponential functions, but also allows the programmer to develop his own special subprograms for his own special needs.

V.B.2 In this discussion, the program that references or "calls" a subprogram is referred to as the main program. It may be a subprogram which references another subprogram.

One subprogram may reference another subprogram. When this situation occurs, the subprogram which references the other is referred to as the _____ program.

Answer: main

V.B.3 A subprogram is written as an entity which is completely separate from the main program. Once a subprogram is written, it can be used with any program which has need of that specific computation.

V.B.4 In using a subprogram, the programmer must do the following:

1.) Indicate at what particular points in the program a subprogram is to be used. This is done by a reference to the subprogram.

2.) Provide the necessary arguments to the subprogram.

V.B.4 (Cont.)

3.) Make sure that the arguments are of the type required by the subprogram.

4.) Indicate to the main program the type(s) of the result(s).

V.B.5

See exercise V.B for an illustration of the use of a subprogram and work Exercises V.B in your workbook at this time.

V.C Function Subprograms

V.C.1 The preceding section discussed subprograms in general. The two different types of subprograms are function subprograms and subroutine subprograms. The first of these to be discussed is the function subprogram.

V.C.2 Just like statement functions, the function subprogram must be given a name. The word FUNCTION, the name of the subprogram and the formal parameters form the first statement. The type must precede FUNCTION if the first letter of the name does not properly indicate the type.

The function subprogram, like the statement function, must have at least _____ argument and not more than _____ arguments. Answer: 1, 60

V.C.3 The following function subprogram will sum the first N values from the array A.

FORTRAN CODING FORM

C for Comment

Statement No. (1–5)	Cont. (7)	FORTRAN STATEMENT (7–50)	(73–80)
		FUNCTION SUM(A,N)	
		DIMENSION A(1000)	
5		SUM=0	
		DO10 K=1,N	
		SUM=SUM+A(K)	
10		CONTINUE	
		RETURN	
		END	

V.C.3 (Cont.)

In the example above, the name of the function is _____.

Answer: SUM

The type of the function is _____. This means that the resulting value will be real.

Answer: real

V.C.4

In the example of section V.C.3, you probably noticed that the name of the function is used as a variable within the function. This must always be done in order to indicate the result to be returned to the main program. Thus, the value of SUM at the time control is returned to the main program is the resulting value of the function.

The variable SUM has the same name as the function subprogram. The value which is returned to the main program is the value of the _____ which has the same name as the subprogram.

Answer: variable

V.C.5

In the example of a subprogram, V.C.3, attention is called to the RETURN statement. A RETURN statement is an executable statement which terminates the logical flow of the subprogram and returns the logical program flow to the main program. A return from a function subprogram would be to the statement where the subprogram was referenced.

Example:

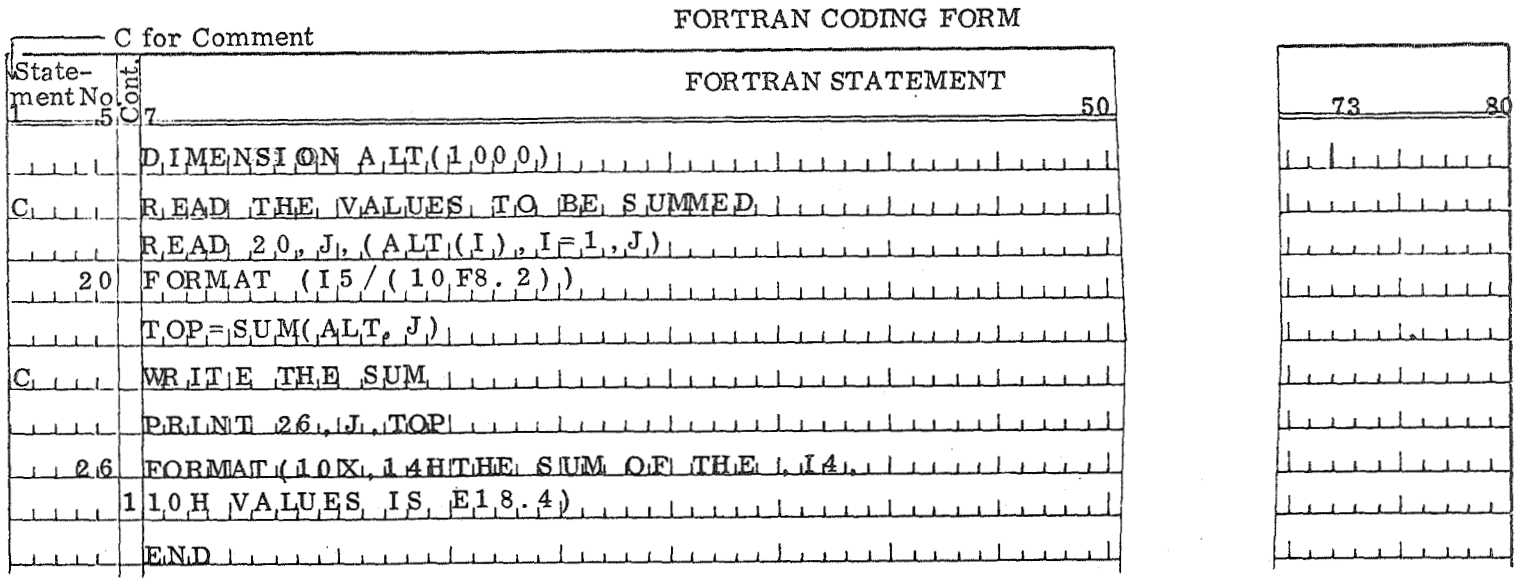
```
Main Program
    .
    .
    .
10 FL = SUM(ALPHA, N)/FN
    .
    .
    .
```

V.C.5 (Cont.) SUM (ALPHA, N) references FUNCTION SUM (A, N). When RETURN is encountered, the logical program flow is returned to statement 10 of the main program. The result of SUM is divided by FN and the quotient is stored in FL.

If the RETURN statement had followed statement 5 in section V.C.3, the resulting value of the function would always be _____.

Answer: zero

V.C.6 A function subprogram is referenced in exactly the same manner as a statement function. The following program references the function subprogram which was written in V.C.3.



In the above example, the first statement sets a maximum for the number of values which can be assigned to the array ALT. This maximum is _____.

Answer: 1000

V.C.7 In the example of V.C.6, the read statement reads J values into the array ALT. If more than 1000 values are read, no diagnostic message occurs, but the resulting sum may be incorrect. One of the reasons for this is that all values read after the 1000th value would be stored in locations assigned to be used by other variables, constants or parts of the program. Even if TOP were to give the correct value many other errors could occur. This means that program restrictions must be well known to the person using the program. The programmer, therefore, should make a note of restrictions of this type as he writes the program.

Overflowing data arrays at execution time may result in _____ answers without any indication that the program is being used improperly.

Answer: incorrect

V.C.8 Suppose the function CPLX(A, B) is to produce a complex result. The first card would read

```
COMPLEX FUNCTION CPLX(A, B)
```

In this case, do not forget to declare CPLX complex in the main program.

V.C.9 The function subprogram is generally used when the subprogram is to calculate a single result. The trigonometric, exponential, and logarithmic functions are examples of this type of subprogram. These are kept in a FORTRAN library and are automatically added to programs which reference them. See your reference manual for a complete list of available functions and the appropriate arguments.

V.C.10 Work exercise V.C in your workbook at this time.

V.D Subroutine Subprograms

V.D.1 The Subroutine Subprogram is used when several values are to be returned to the main program.

V.D.2 Since the subroutine name is not associated with a single variable, the subroutine name is not associated with the type of any of the variables involved. Any valid variable name is valid as a subroutine subprogram name.

The subroutine name (is, is not) directly connected with the type of the values being calculated? _____ Answer: is not

V.D.3 A subroutine, like a function, requires a name and can have up to 60 parameters. It differs from the function, however, in the fact that subroutines can be written with no parameters.

A function must have at least _____ parameter(s). Answer: 1

Subroutines may be written with a minimum of _____ parameters. Answer: 0

V.D.4 The first card of a subroutine is written with the word SUBROUTINE followed by the subroutine name and parameter list (if any). For example,

V.D.4
(Cont.)

C for Comment — FORTRAN CODING FORM

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
		SUBROUTINE EXCHNG(A, B)	
		TEMP=B	
		B=A	
		A=TEMP	
		RETURN	
		END	

will exchange the values A and B.

Parameters in the example above were used to pass information to the subroutine. They were also used to pass information back to the _____.

Answer: main program

V.D.5 A reference to a subroutine is always made in a separate statement. For example,

C for Comment — FORTRAN CODING FORM

Statement No. (1–5)	Cont.	FORTRAN STATEMENT (7–50)	(73–80)
		X=28.6	
		Y=30.7	
		CALL EXCHNG(X, Y)	

will set Y = 30.7 and Y = 28.6

V.D.5 (Cont.) A reference to a subroutine can not be made within an expression. In fact, a subroutine reference must always be made in a separate _____.

Answer: statement

V.D.6 The reference to a subroutine always starts with the word CALL followed by the subroutine name and the parameter list (if any). The CALL statement may have a statement number and may use continuation cards if needed.

V.D.7 Review the definition of RETURN in V.C.5. Look at SUBROUTINE EXCHNG, V.D.4. In the case of a subroutine, the logical flow of the subprogram is terminated and control is returned to the next sequential statement of the main program.

Example:

```
      Main Program
      .
      .
      .
      X = 28.6
      Y =30.7
      .
      .
      .
      CALL EXCHNG(X, Y)
   10 Z = C*EXP(X)
      .
      .
      .
```

CALL EXCHNG(X, Y) references SUBROUTINE EXCHNG(A, B). The logical flow of the program is sent to SUBROUTINE EXCHNG. RETURN terminates the logical flow in the subprogram and returns it to statement 10 in the main program.

V.D.8 Work exercise V.D in your workbook at this time.

V.E Available Functions

There are many functions which have been written by other programmers which are available for your use. These functions are divided into two categories. One set is called in-line functions and the other set is called library functions. All functions in these two sets are referenced in exactly the same way, but there are some differences which may be of interest.

The two categories of functions which are available for your use are _____ _____ and _____ _____.

Answer: in-line functions
library functions

V.E.1 The in-line functions are not actually subprograms, but small sections of computer instructions which the FORTRAN compiler inserts into your program at the place where the function is referenced.

The computer code for a specific in-line function is placed in the program every time a reference is made to that function.

Reference to an in-line function does not cause control to be passed to a subprogram, but just adds computer instructions to the main program. True or false? _____

Answer: True

V.E.2 Library functions are actually subprograms which are available for your use. A reference to any of these functions will cause the referenced function to be taken from the library and placed in computer memory whenever your program is being executed.

Since the available functions are all referenced in the same manner, it is usually not necessary to remember which functions belong to what category. True or false? _____.

Answer: True

V.E.3 To use the available functions, it is necessary that the type of the actual parameters are correct. It is also important to know the type of the result you will obtain. The other important item is to be sure that you understand what the function does.

In using a function which someone else has written, you need to know the definition of what the function does, the type(s) and meaning(s) of the different parameters, and the _____ of the result.

Answer: type

V.E.4 We have discussed in-line and library function in general. We will now discuss several of the most popular of these functions.

a.) There are two in-line functions which take the absolute value of a number. They are ABS(X) and IABS(J). They will return the value of the argument whenever the argument is positive. They will return the negative of the argument whenever the argument is negative.

Form	Actual Parameter Type	Type of Result	Definition
ABS(X)	real	real	|x|
IABS(J)	integer	integer	|j|

Write a statement which will add the absolute value of x to the absolute value of y and assign the result to z.

Answer: Z = ABS(X)+ABS(Y)

V.E.4
(Cont.)

b.) There are ten in-line functions for picking a minimum or maximum value from the parameter list. These functions may have from two to sixty parameters in a particular reference. For example, the statement

B = AMAX1(A, P, X, Y)

will assign B the maximum value represented by the set of values A, P, X, and Y.

The different functions for minimum and maximum are the variations on parameter type and result type.

Form	Actual Parameter Type	Type of Result	Definition
AMAX1(X_1, X_2, ..., X_n)	real	real	maximum value
AMIN1(X_1, X_2, ..., X_n)	real	real	minimum value
AMAX0(I_1, I_2, ..., I_n)	integer	real	maximum value
AMIN0(I_1, I_2, ..., I_n)	integer	real	minimum value
MAX1(X_1, X_2, ..., X_n)	real	integer	maximum value
MIN1(X_1, X_2, ..., X_n)	real	integer	minimum value
MAX0(I_1, I_2, ..., I_n)	integer	integer	maximum value
MIN0(I_1, I_2, ..., I_n)	integer	integer	minimum value
DMAX1(D_1, D_2, ..., D_n)	double precision	double precision	maximum value
DMIN1 (D_1, D_2, ..., D_n)	double precision	double precision	minimum value

Write a statement which will assign the minimum value of the variables I, J, and N to the real variable Y.

Answer: Y=MIN0 (I, J, N)

or

Y=AMIN0 (I, J, N)

V.E.4
(Cont.)

c.) There are two in-line functions which will perform modulo arithmetic. These functions require two parameters and the result is the remainder obtained by dividing the first actual parameter by the second actual parameter.

Form	Actual Parameter Type	Result Type	Definition
AMOD(X_1, X_2)	real	real	X_1 modulo X_2
MOD(I_1, I_2)	integer	integer	I_1 modulo I_2

Write a statement which will set J equal to the number of years since the last leap year. Assume that the year is stored in variable IYEAR and the year is between 1950 and 1990.

Answer: J = MOD(IYEAR, 4)

---

d.) Four other in-line functions are useful when working with complex arithmetic. These are used to combine real variables or constants to form a complex variable or to form real variables from the different parts of a complex variable.

Form	Actual Parameter Type	Type of Result	Definition
AIMAG(C)	complex	real	obtain the imaginary part of a complex argument
CONJG(C)	complex	complex	change sign of imaginary part of complex argument (Conjugate of C)
CMPLX(X_1, X_2)	real	complex	form complex number from pair of real arguments (X_1 + iX_2)
REAL(C)	complex	real	obtain real part of complex argument.

V.E.4 (Cont.) Consider the following code:

FORTRAN CODING FORM

C for Comment

Statement No. 1–5	Cont.	FORTRAN STATEMENT 7–50	73–80
		COMPLEX D, E, F	
		F=(20.4,15.6)	
		E=(2.5,6.3)	
		X=AIMAG(F)	
		Y=REAL(E)	
		D=CMPLX(X,Y)	

The value of D after the above instructions is _____ + i _____.

Answer: 15.6, 2.5

e.) The library functions are mainly the trigonometric, exponential, and square root computations. For the trigonometric functions, all angular arguments and results are in radians.

Form	Actual Parameter Type	Result Type	Definition
SIN(X)	real	real	sine of X radians
COS(X)	real	real	cosine of X radians
TAN(X)	real	real	tangent of X radians
TANH(X)	real	real	Hyperbolic tangent of X radians

V.E.4 (Cont.)

Form	Actual Parameter Type	Result Type	Definition
ALOG(X)	real	real	natural logarithm of X
ALOG10(X)	real	real	logarithm to base 10 of X
EXP(X)	real	real	e to the X power
SQRT(X)	real	real	square root of X
ASIN(X)	real	real	arcsine of X. The result is in the range $[-\pi/2, \pi/2]$
ACOS(X)	real	real	arccosine of X. The result is the range $[0, \pi]$
ATAN(X)	real	real	arctangent of X. The result is the range $[-\pi/2, \pi/2]$
ATAN2(X_1, X_2)	real	real	arctangent of X_1/X_2. The result is in the range $[0, 2\pi]$

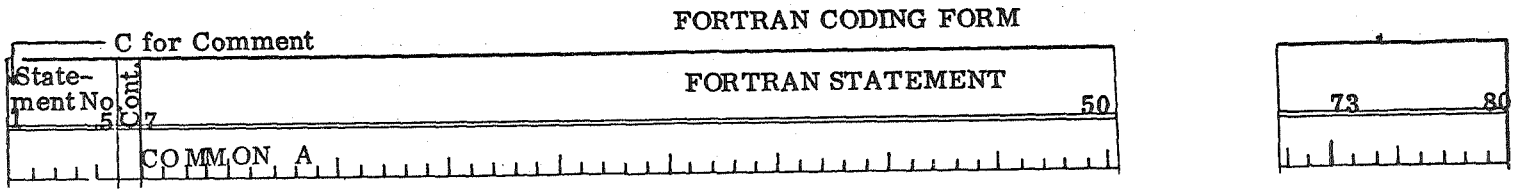
Most of these same functions are also available for double precision and complex computations. See your reference manual for a complete list.

V.E.5 Work exercise V.E. in your workbook at this time.

V.F COMMON

V.F.1 I am sure that some of you are curious as to how information is passed to and from subroutines which have no parameters. This is accomplished through the use of COMMON. Functions may also gain access to main program variables through COMMON assignments.

V.F.2 Suppose the statement



appears in the main program. This tells the main program that this is a variable which will probably be used by one or more subprograms. It is then assigned the first location in the computer memory which is reserved for common variables.

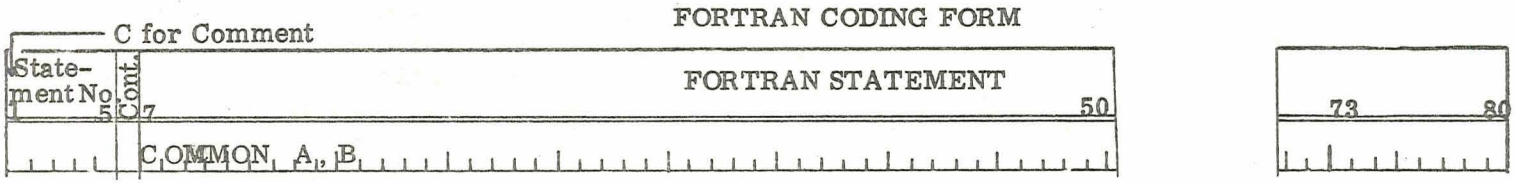
Common variables are assigned computer memory locations in the order the variables appear in the COMMON statements. The statement COMMON A, B will assign A to the first location in COMMON and B to the _____ location in COMMON.

Answer: second

```
COMMON  A
COMMON  B
```

will also assign A to the first location in COMMON and B to the second location in COMMON.

V.F.3 If the main program has the statement



and a subprogram has the same COMMCN statement, then references to the variables A and B in the subprogram will use the same values as references to A and B in the main program. This follows from the fact that the subprogram will assign the two variables to the same computer memory locations.

Please notice that the order of assignment to the common area of memory is the important factor and not the names of the variables. If a COMMON statement in the subprogram had contained only the variable B, would a reference to B in the subprogram use the value of B in the main program? _____

Answer: No

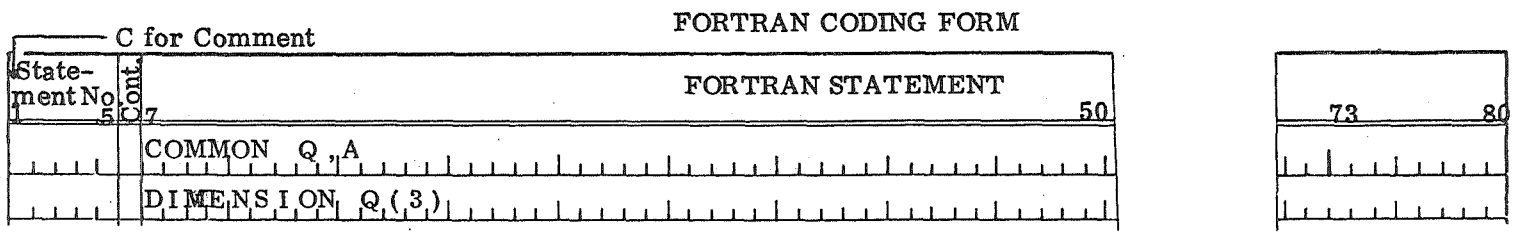
V.F.4 Actually, the value of A from the main program would be used. An illustration follows:

main program		common area		subprogram
COMMON A	→	value 1	←	COMMON B
COMMON B	→	value 2		

V.F.4 (Cont.) In making variables available to subprograms through the use of COMMON, it is the _____ of the variable in the COMMON definition which is important and not the variable name.

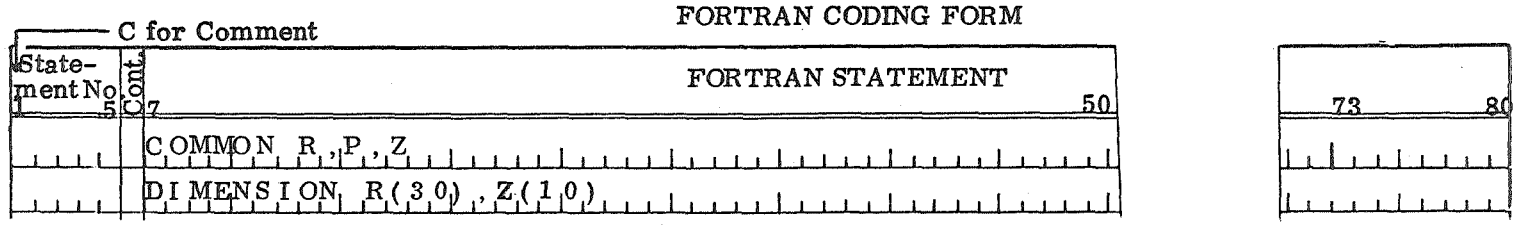
Answer: position

V.F.5 When a subscripted variable is put in COMMON, the entire array is assigned to COMMON. For example, the two statements



will place A in the fourth location of COMMON.

The statements



will place P in the _____ location of COMMON

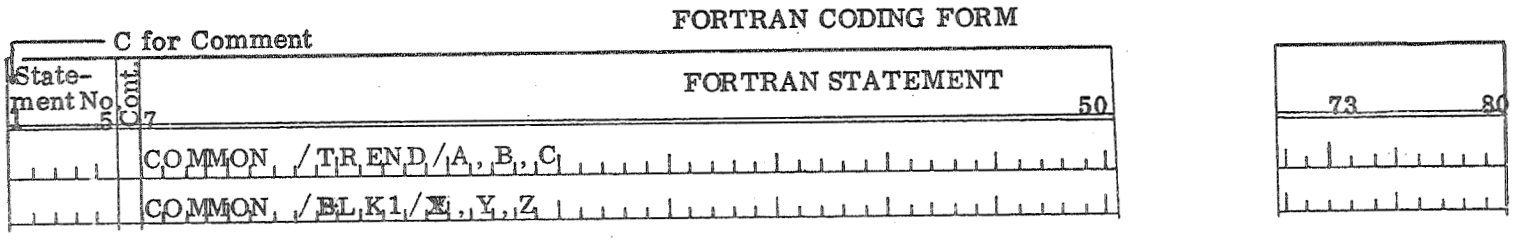
Answer: 31st

V.F.6 Often a main program has many variables, with different variables being needed by different subprograms. To avoid including all common variables in all the subprograms, it is possible to define several common areas. This is done by assigning names to the different common areas.

V. F. 6 (Cont.) The variables needed by a specific subprogram can be grouped into a separate common area. The different common areas are identified by a _____.

Answer: name

V. F. 7 The common area with no name is called blank common. The common areas with names are called labeled common and the name identifies the different blocks. The following statements show how labeled common is defined.



The block name appears between the two slashes. The variables are ordered within the common blocks in the same way they are ordered in blank common.

If X, Y, and Z are simple variables, Z will occupy the _____ location of block BLK1.

Answer: third

V. F. 8 Names of common blocks may be 1-7 characters. If the first character is a number, all must be numbers within that name. A labeled common name may be the same as a variable name used within the program but cannot be the same as a subprogram name.

V. F. 8 (Cont.) If the first character of a block common is a number, then all characters in the name must be_____.

Answer: numbers

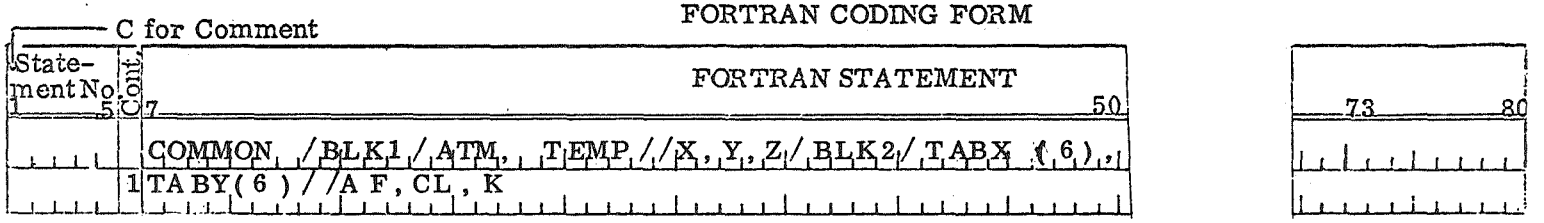
V. F. 9 Blank common and labeled common may be used in the same program. A particular variable must only appear once in a COMMON statement.

Is it permissible to define the variable A in the common block TREND and also in the common block THINK?_____

Answer: No

Blank common and labeled common declarations can be made in the same COMMON statement. The // indicates blank common.

In the COMMON statement

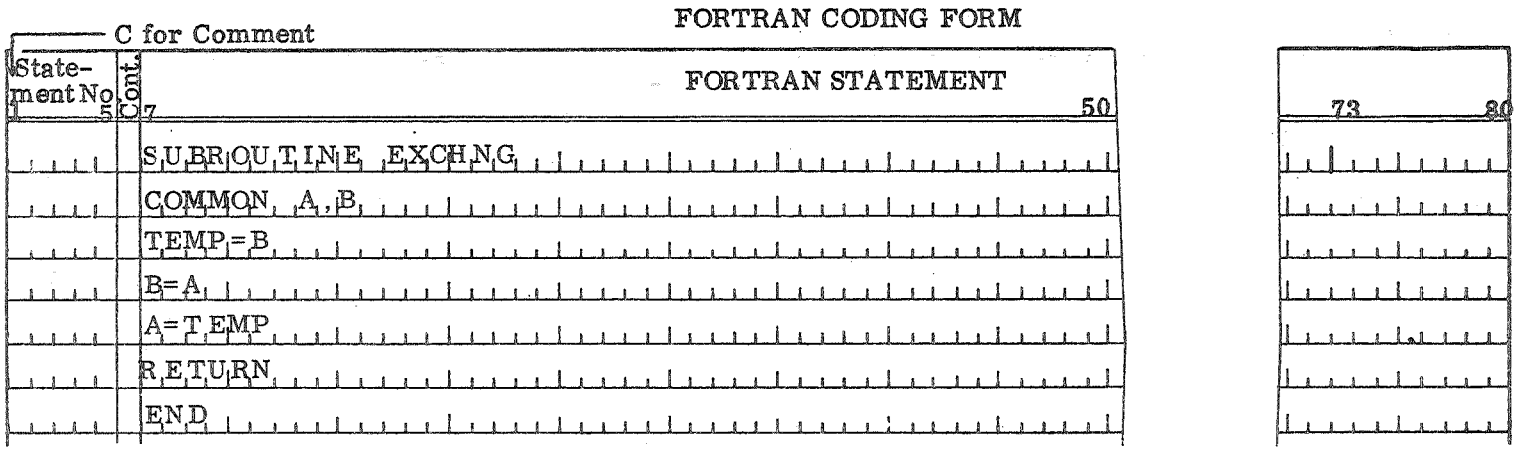


what variables are located (1) in the labled common block, BLK1; (2) in the labled common block, BLK2; (3) in blank common?

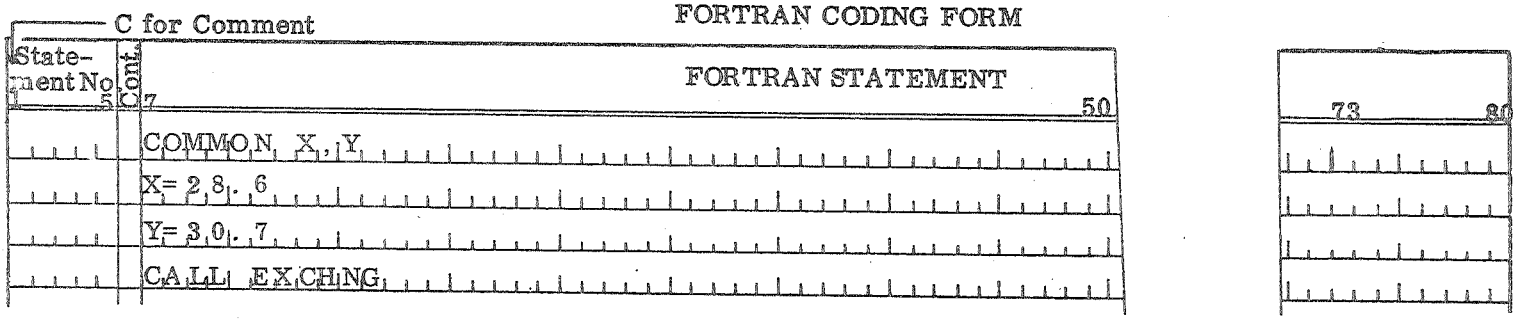
(1)_____
(2)_____
(3)_____

Answer:
(1) ATM, TEMP
(2) TABX, TABY
(3) X, Y, Z, AF, CL, K

V.F.10 The subroutine EXCHNG is now written using COMMON.



The main program from section V.D.5 is now written using COMMON.



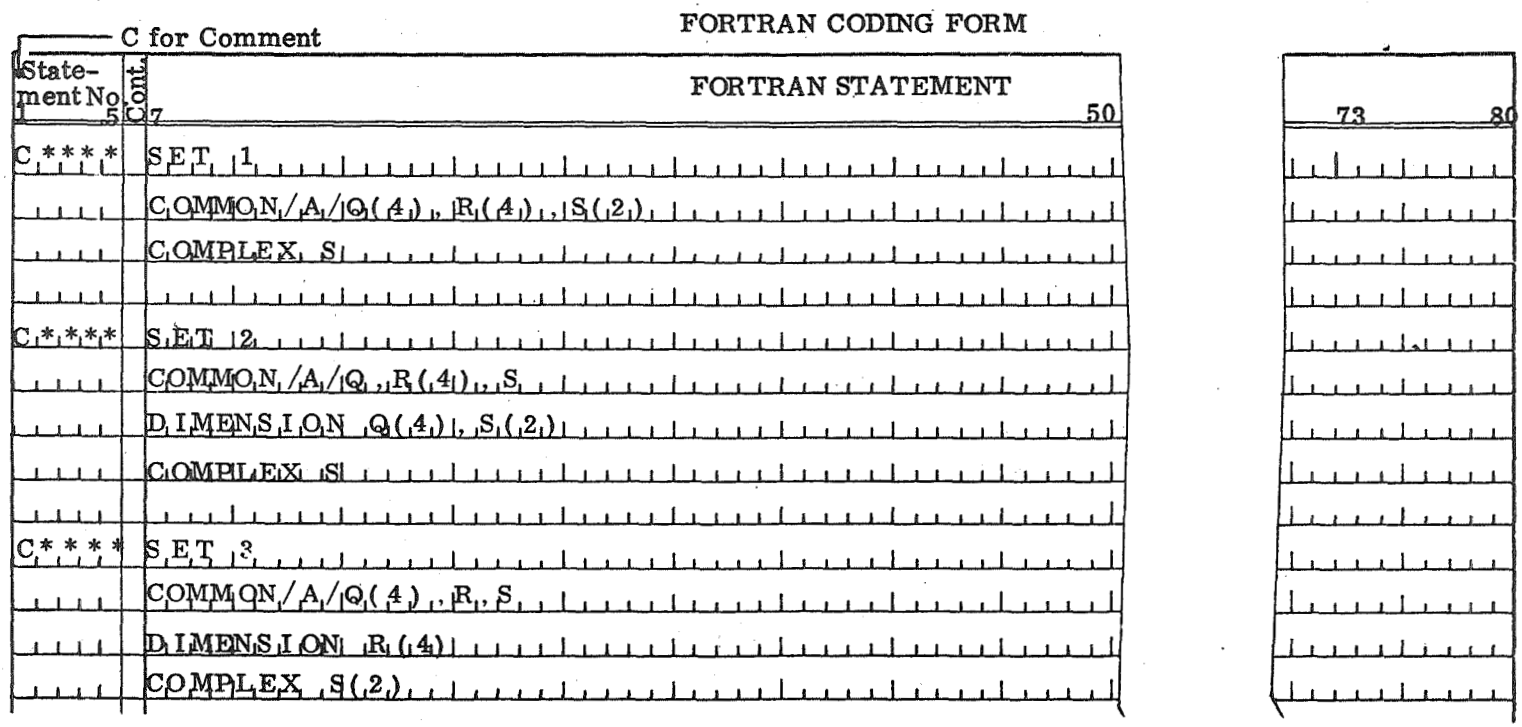
The value of X after the reference to EXCHNG is _____.

Answer: 30.7

Warning: If a variable appears in COMMON in a subprogram, it must not be a formal parameter of the same subprogram.

V.F.11 For convenience, it is permissible to declare the dimensions of a subscripted variable in the COMMON or type statements. This means that if a variable must appear in a COMMON or type statement there is no need to enter it again in a DIMENSION statement. The dimension must be declared in only one of the three statements, COMMON, type and DIMENSION.

This is illustrated in the following statement sets.



In the first set, S is dimensioned in the _____ statement. Answer: COMMON

In the second set, S is dimensioned in the _____ statement. Answer: DIMENSION

In the third set, S is dimensioned in the _____ statement. Answer: type

V. F. 12 If a subprogram is to be used with many different main programs, it is permissible to use integer variables in DIMENSION statements within the subprogram. These are called variable dimensions or adjustable dimensions. The following matrix multiply subroutine illustrates this technique.

FORTRAN CODING FORM

C for Comment

Statement No. (1–5)	Cont. (6)	FORTRAN STATEMENT (7–50)	(73–80)
		SUBROUTINE MATMUL(A, B, C, N, M, K, NMAX, MMAX, KMAX)	
		DIMENSION A(NMAX, MMAX), B(MMAX, KMAX),	
	1	C(NMAX, KMAX)	
		DO 30 I = 1, N	
		DO 30 J = 1, K	
		C(I, J) = 0.0	
		DO 30 L = 1, M	
		C(I, J) = C(I, J) + A(I, L) * B(L, J)	
30		CONTINUE	
		RETURN	
		END	

The values NMAX, MMAX, and KMAX represent the maximum permissible values of N, M, and K respectively. These maximum values must be the same as the constants used in the corresponding DIMENSION statements in the main program. The values of N, M, and K may vary depending on the input data for a particular run. They must always be the same or less than the maximum values used in dimensioning the arrays. Specifically, when NMAX is 50, the value of N must never be larger than _____.

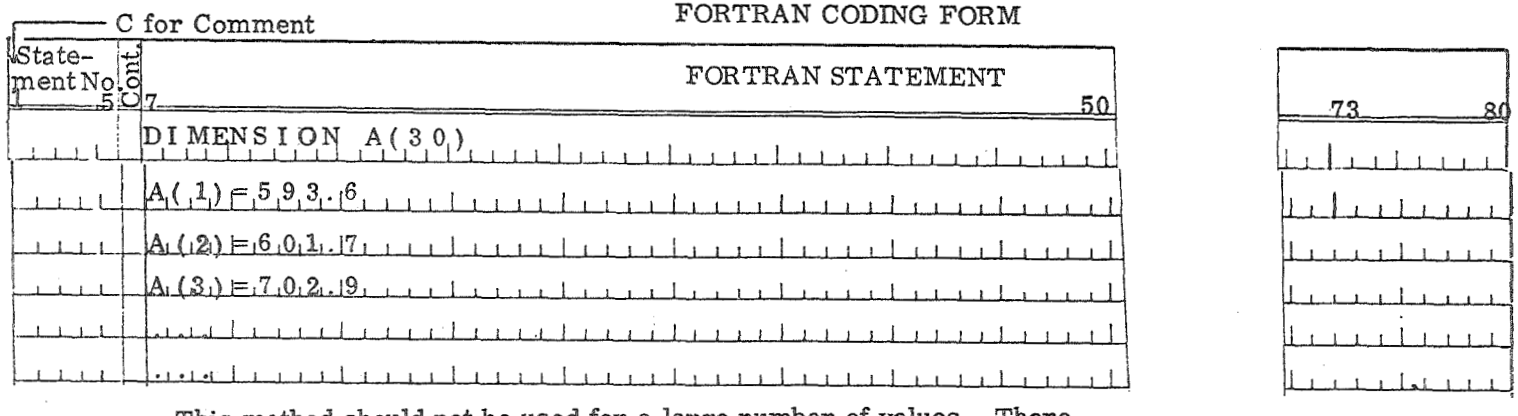
Answer: 50

V.F.13 The formal parameters of a subprogram cannot appear in COMMON.

V.F.14 Work exercise V.F in your workbook at this time.

V.G Block Data

V.G.1 Often it is necessary to set up tables of constants which are used by the program. There are several ways to handle data tables but some are rather inefficient. One way is to read in the values as input data. This means that a deck of cards containing the table values must be submitted with each computer run. A second method is to write the constants into the program. For example,



This method should not be used for a large number of values. There are two reasons for the previous statement. The first disadvantage is that the amount of computer memory required is twice that of other methods. This arises from the fact that the constants must be stored and then moved to the array when execution of the program occurs. The second disadvantage is the computer time required to move the constants to the array.

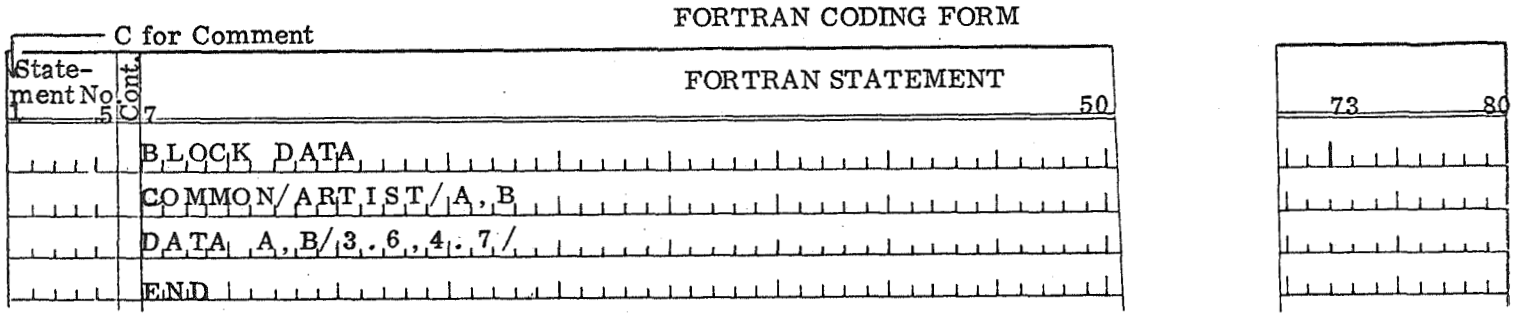
V.G.1 (Cont.) The block data subprogram places the constants in the appropriate array locations at the time the subprogram is compiled. The only disadvantage of this method is that the variables involved must be in labeled common.

V.G.2 No computations are permitted in a block data subprogram. In fact, it is not assigned a name and no reference may be made to it.

All variables which are assigned constants values in a block data subprogram must appear in a _____ statement.

Answer: labeled common

V.G.3 Type and DIMENSION statements must also be included when appropriate. The actual assignment of constants to the different variables is accomplished by the use of DATA statements. The simplest form of the DATA statement is the word DATA followed by the variables to be assigned values, a slash, the values, then an ending slash. For example,



will set A to the value 3.6 and B to the value 4.7.

V.G.3 (Cont.) The variables A and B are assigned to the common block named _____.

Answer: ARTIST

V.G.4 A more useful form of the DATA statement is shown in the following example.

FORTRAN CODING FORM

C for Comment

Statement No (1–5)	Cont.	FORTRAN STATEMENT (7–50)	73–80
		BLOCK DATA	
		COMMON/TREND/A(200)	
		DATA(A(I),I=1,200)/4.06,20*12.0,175*0.0,	
	1	16.32*10.0/	
		END	

This form permits the assignment of constants to data arrays. The (A(I), I=1,200) is called an implied DO loop and works in a manner similar to the DO loop. In the example above, the value of I takes on all integer values from 1 to 200. Like a DO loop, an increment different from one may be specified. The 20*12.0 means to use the real value 12.0 twenty times.

The 175*0.0 means to use the real value _____ one hundred and seventy-five times.

Answer: 0.0

V.G.5 Different constant types may be mixed in the same DATA statement. The constant type should match the corresponding variable type, however.

If the type of the variable is integer, the corresponding constant should be _____.

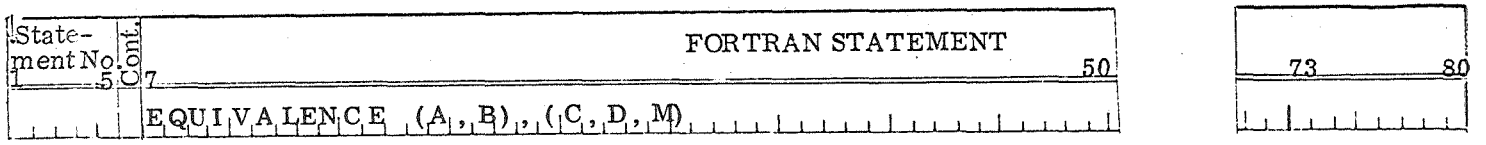
Answer: integer

V.G.6 Work exercise V. G. in your workbook at this time.

V.H EQUIVALENCE Statements

It is often desirable to have alternate names for the same storage location. In FORTRAN, the vehicle which allows alternate names of storage locations is the EQUIVALENCE declaration.

V.H.1 The EQUIVALENCE declaration is coded as follows:



Each set of parentheses encloses the variables that share a storage location.

In the above example, suppose A, B, C, D, M are all simple variables. Which sets of variables share storage locations?

Answer: A and B share one. C, D, M share another.

V.H.2 The order of variables within the parentheses is immaterial. Thus the same result may be obtained from: EQUIVALENCE (M, C, D), (B, A).

Notice that the variable set C, D, M involves both real and integer variables. More about that later.

The EQUIVALENCE declaration is a non-executable statement that can appear anywhere in a program or subprogram.

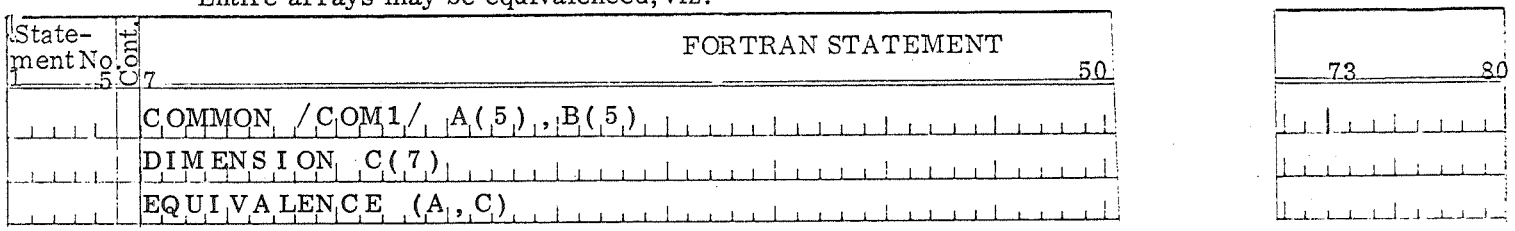
The EQUIVALENCE declaration is used more often with arrays than with simple variables. Double precision and complex variables may also be involved.

In earlier sections, you learned of three non-executable FORTRAN statements that allow dimensioning of arrays. Name them.

Answer: COMMON, DIMENSION, type

V.H.3 Arrays to be used in the EQUIVALENCE declaration must be dimensioned in either a COMMON, DIMENSION, or type statement in the same program or subprogram.

Entire arrays may be equivalenced, viz:



This series of statements specifies that the array C will begin at the same location as the array A. That is,

C(1) is the same as A(1)
C(2) is the same as A(2)

Complete the array through C(7) using " <=> " for "is the same as".

Answer: C(3)<=>A(3)
C(4)<=>A(4)
C(5)<=>A(5)
C(6)<=>B(1)
C(7)<=>B(2)

Remember that the array B immediately follows the array A in storage when defined in COMMON. The first two locations of B were equivalenced to the last two of C implicitly -- just because B follows A.

V.H.3 (Cont.) The same result could have been obtained by writing:

FORTRAN CODING FORM

C for Comment

State-ment No. 1 5	Cont.	FORTRAN STATEMENT 7 50
		COMMON /COM1/ A(5), B(5)
		DIMENSION C(7)
		EQUIVALENCE (C(6), B)

73 80

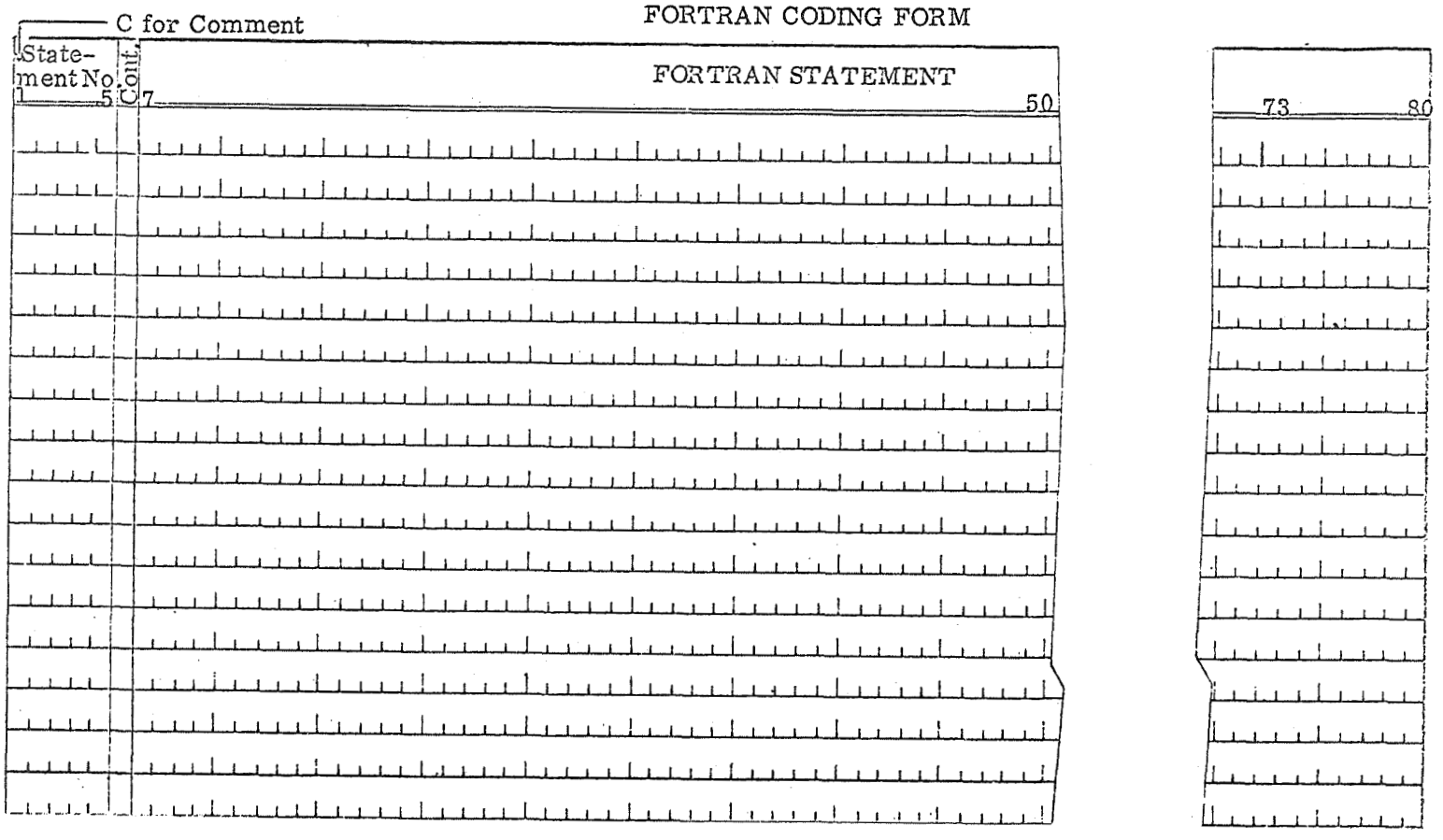
When written as shown, the first five locations of C correspond to the respective locations of A. Why?

Answer: A immediately precedes B in COMMON.

Notice that C(6) was written in the EQUIVALENCE statement. C(6) refers to the sixth location of C and not to the total dimension of the array. The array must have been dimensioned in a COMMON or DIMENSION statement.

V.H.3 (Cont.)

You have seen two different ways of equivalencing the same arrays using different elements of C. There are seven such ways using A, B, C as defined above. Write them:



V.H.3
(Cont.)

Answer:

C for Comment

FORTRAN CODING FORM

	Statement No.	Cont.	FORTRAN STATEMENT (7–50)	73–80
1			EQUIVALENCE (A,C) or EQUIVALENCE (A(1),C(1)) or EQUIVALENCE (C,A) or EQUIVALENCE (C(1),A(1))	
2			EQUIVALENCE (A(2),C(2))	
3			EQUIVALENCE (A(3),C(3))	
4			EQUIVALENCE (A(4),C(4))	
5			EQUIVALENCE (A(5),C(5))	
6			EQUIVALENCE (B,C(6))	
7			EQUIVALENCE (B(2),C(7))	

V.H.4 It is possible to equivalence arrays of different dimensions. Consider the following example:

C for Comment — FORTRAN CODING FORM

State-ment No. 1–5	Cont.	FORTRAN STATEMENT 7–50	73–
		DIMENSION D(5,2),E(5),F(5),G(12)	
		EQUIVALENCE (D(1,1),E,G),(D(1,2),F)	

Assuming D, E, F, G are not complex or double precision, complete the following tableau where " <=> " is "is the same as".

D(1, 1) <=> E(1) <=> G(1)

D(2, 1) <=> E(2) <=> G(2)

_____ G(3)

_____ G(4)

_____ G(5)

_____ G(6)

_____ G(7)

_____ G(8)

_____ G(9)

_____ G(10)

_____ G(11)

_____ G(12)

Answer:

D(1, 1) <=> E(1) <=> G(1)

D(2, 1) <=> E(2) <=> G(2)

D(3, 1) <=> E(3) <=> G(3)

D(4, 1) <=> E(4) <=> G(4)

D(5, 1) <=> E(5) <=> G(5)

D(1, 2) <=> F(1) <=> G(6)

D(2, 2) <=> F(2) <=> G(7)

D(3, 2) <=> F(3) <=> G(8)

D(4, 2) <=> F(4) <=> G(9)

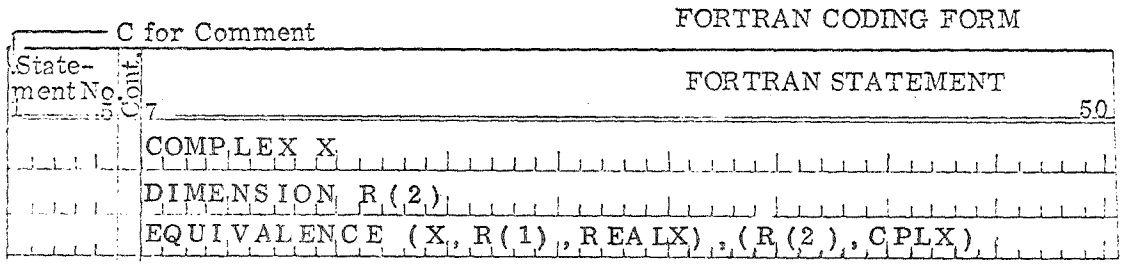
D(5, 2) <=> F(5) <=> G(10)

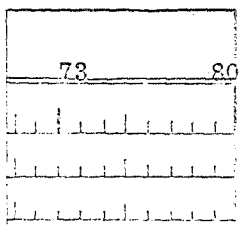
G(11)

G(12)

V.H.4 (Cont.)

Notice that we have forced an order on E and F--namely, F follows E. Also, there are no correspondents for G(11) and G(12). We pointedly ignored double precision and complex variables until now. Given the following sequence of statements:

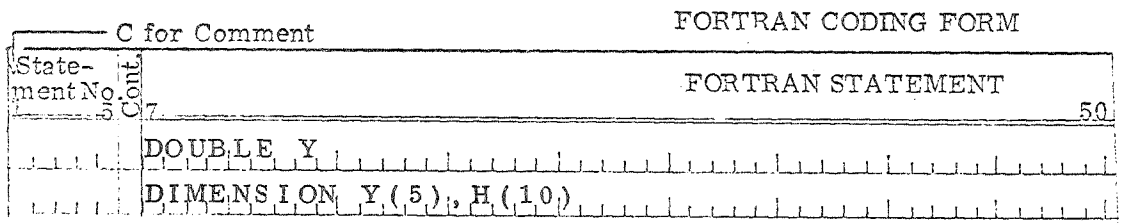


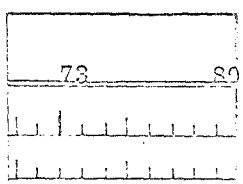


How would you think the real and imaginary portions of X would be addressed?

Answer: real part of X in R(1) or REALX
imaginary part of X in R(2) or CPLX

The EQUIVALENCE declaration used as above is one way to get directly at the imaginary part of a complex variable or the least significant part of a double precision variable.





Suppose you wanted to reference the least significant portion of Y(2) by the variable name DOWN. Write an EQUIVALENCE statement using Y, H and DOWN that does it.

Answer:

EQUIVALENCE(Y, H), (H(4), DOWN)

V.H.5 We mentioned the notion of forcing an order on variables within a program or subprogram. Certain restrictions have to be followed when variables are in COMMON.

(1) Variables or arrays within COMMON cannot be reordered by EQUIVALENCE. That is:

```
COMMON A, B
EQUIVALENCE (A, B)
```

is not allowed.

(2) COMMON should not be extended by EQUIVALENCE since disastrous destruction of subsequent storage areas may result. In other words, avoid things like:

```
COMMON /COM2/ C, D(5)
DIMENSION E(6)
EQUIVALENCE (D, E)
```

Setting E(6) would produce a store in the location following /COM2/ which might be another common area.

(3) Variables in different common areas may not be equivalenced.

In addition to the above restrictions regarding COMMON, one general restriction should be observed.

(4) Equivalenced arrays must be consistent. You can't have something like:

```
DIMENSION A(3), B(2), C(2)
EQUIVALENCE(A, B), (A(3), C), (B(2), C)
```

V.H.5
(Cont.)

Why not?

_____

Answer:

The first parentheses require
(a) A(1)<=>B(1)
(b) A(2)<=>B(2)
The second require
(c) A(3)<=>C(1)
The third require
(d) B(2)<=>C(1)

(b) and (d) states that A(2)<=>B(2)<=>C(1)
(c) states that A(3)<=>C(1)
C(1) cannot be both A(2) and A(3)

In the following set of statements, each of the equivalences (a), (b), (c), (d) violates one of the restrictions. Write the number of the restriction that has been ignored next to the equivalence:

FORTRAN CODING FORM

C for Comment

	Statement No.	Cont.	FORTRAN STATEMENT (7–50)	73–80
			COMMON /COM3/ A(3),B(5,2)	
			COMMON /COM4/ C(6,3),D(2)	
			DIMENSION E(19),F(13)	
a.			EQUIVALENCE (E(19),B(5,2))	
b.			EQUIVALENCE (F,A),(F(7),C)	
c.			EQUIVALENCE (E,C),(E(19),D(2))	
d.			EQUIVALENCE (C,D)	

V.H.5
(Cont.)

a. _____

b. _____

c. _____

d. _____

Answer:

(a)-(2): /COM3/ is only thirteen locations long, thus, there is no place in COM3 for for the first six locations of E. We've extended COM3 backward which could be dangerous.
(b)-(3): Both COMMONs have been equivalenced
(c)-(4): according to the parentheses (E, C), D(1) should be E(19). The second parentheses contradicts this
(d)-(1): can't reorder COMMON

V.H.6 As the final step let's see what happens when a real variable is equivalenced to an integer variable. Actually nothing very exciting occurs. You are allowed, however, to address the same variables as integer or real depending on the variable name used. The EQUIVALENCE statement does not cause any real-integer conversion, but the normal conversion rules apply in computations.

Suppose you wanted to read a ten word binary tape record into core and the first, fourth, sixth, and tenth words were integer. You could do the following:

```
DIMENSION PUTIN(10), INPUT(10)
EQUIVALENCE(PUTIN, INPUT)
```

and read the record into PUTIN (or INPUT)

Which values of INPUT would contain integer variables?

Answer: INPUT(1), INPUT(4), INPUT(6), INPUT(10)

Which values of PUTIN would contain integer variables?

Answer: PUTIN(1), PUTIN(4), PUTIN(6), PUTIN(10)

You would have to be careful in referring to the variables.

If you said X = INPUT(2) , a conversion of an already real variable into a real variable would be made-- erroneously. Careful use of mixed mode EQUIVALENCE can, however, facilitate programming, especially in applications similar to the example. Input/output records often have integer values representing checksums, identifiers and the like.

V.H.7 Work exercise V.H in your workbook at this time.

V.I EXTERNAL Statement

The name of a function or subroutine subprogram may be used as an actual parameter in a calling sequence. This is possible through the proper use of the EXTERNAL statement.

V.I.1 Consider the statement

CALL ADMAN (FLGS, X, Y, Z)

where FLGS is the name of a subroutine. Used in this manner, FLGS looks exactly the same as X, Y, and Z. Therefore, it will be considered a variable and not a subroutine. The use of the statement

EXTERNAL FLGS

prior to the statement

CALL ADMAN (FLGS, X, Y, Z)

will identify FLGS as an external name (i.e., a function or subroutine subprogram name).

A subprogram name may be placed in a calling sequence if the name has been included in an _____ statement.

Answer: EXTERNAL

V.I.2 The form of the EXTERNAL statement is the word EXTERNAL followed by the names of the subprograms which are to be used as actual arguments in some calling sequence. The subprogram names are separated by commas.

Write an EXTERNAL statement which declares SIN, COS, and TAN as subprograms. _____

Answer:
EXTERNAL SIN, COS, TAN

V.I.3 It would be easy at this point to confuse the use of a subprogram name in a calling sequence with the use of a reference to a particular function in the calling sequence. To clarify this point, examine the following statements which are contained in the main program.

FORTRAN CODING FORM

C for Comment

Statement No. 1–5	Cont.	FORTRAN STATEMENT 7–50	73–80
		EXTERNAL FLGS, CRAZ	
		DIMENSION X(100), B(300), C(50), P(100), Q(300),	
	1	R(50)	
		CALL ADMAN (FLGS, X, Y, Z)	
		CALL GCNT (FLGS(K), X, Y)	
		CALL ADMAN (CRAZ, P, Q, R)	

Here the first statement defines FLGS and CRAZ as external names. The third and fifth statements call ADMAN which expects a subprogram entry plus three variables as parameters. The fourth statement calls GCNT which expects three variables in the calling sequence. Thus, FLGS(K) is an arithmetic expression which is evaluated before the subroutine is called. The result of the function FLGS(K) then becomes one of the actual parameters.

V.I.3 (Cont.) Consider the following code:

FORTRAN CODING FORM

C for Comment

Statement No.	Cont.	FORTRAN STATEMENT (7–50)	73–80
		SUBROUTINE ADMAN(FNCT,A,B,C)	
		DIMENSION A(100),B(300),C(50)	
		DO 10 I=1,50	
		A(2*I)=FNCT(I)+B(6*I)-C(I)	
10		CONTINUE	
		RETURN	

Since FNCT is not dimensioned, it will be recognized as a function. Its appearance as a formal parameter, however, will mean that the function entry location is supplied by the main program. This means that the function which is actually used in performing the calculations is whatever function name appears as an actual parameter in the call to ADMAN.

Again consider the main program code which was seen earlier.

		EXTERNAL FLGS,CRAZ	
		DIMENSION X(100),B(300),C(50),P(100),Q(300),	
	1	R(50)	
		CALL ADMAN (FLGS,X,Y,Z)	
		CALL GCNT (FLGS(K),X,Y)	
		CALL ADMAN (CRAZ,P,Q,R)	

V.I.3 (Cont.) The third statement will call ADMAN and indicate that function _____ be used in the calculations to be performed. The fifth statement will use function _____ in the same manner.

Answer: FLGS, CRAZ

V.I.4 Work exercise V.I in your workbook at this time.

V.J Exercise V.J in your workbook is a review of Part V. Work exercise V.J at this time.

VI. INPUT/OUTPUT EXTENDED

VI.A Introduction

In part IV, you learned about the conversion types available for real, integer, complex, and alphanumeric quantities. These were Ew.d, Fw.d, Iw and wH. The individual format specifications are sufficient to read data, input headings and general information, and print results for the average FORTRAN program.

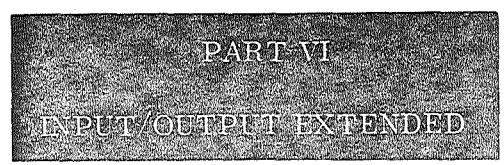
In this chapter, input/output is extended to include topics beyond the minimum FORTRAN capability. We will study conversion types for double precision and logical variables, scale factors, formats and techniques for character manipulation, statements for rearranging and changing information in storage, input/output with NAMELIST, and other features for more flexible programming.

VI.A.1 Dw.d, Input and Output

Before reading the following sections on double precision input and output, it is suggested that you review the discussion of the double precision mode in II.E, and Ew.d conversion in IV.E (Output), IV.F (Input).

VI.A.2

Dw.d conversion is used for input and output of double precision quantities. A double precision quantity is represented and used like a real quantity but has more digits. Thus Dw.d conversion is similar to Ew.d conversion, except that (1) D replaces the E in the format specification, (2) two memory locations are allocated for each double precision quantity, (3) the list variables must be double precision names. Both w and d have the same meaning.



VI.A.2 (Cont.) Example:

```
      DOUBLE EXTEND(50)
   10 FORMAT (4D18.7)
      READ 10, EXTEND
```

In the example above, EXTEND is an array name typed as double precision. The number of double precision elements in the array is 50, but the number of memory locations reserved is 100.

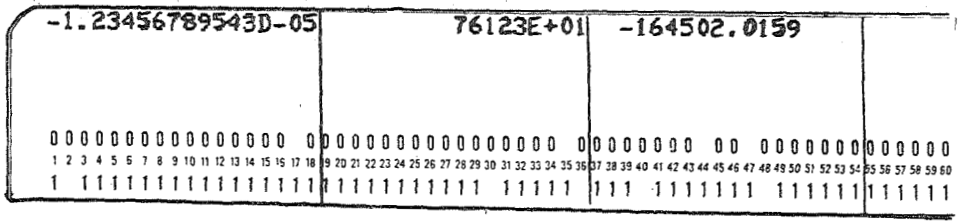
VI.A.3 Dw.d Input

The field occupies w positions (card columns) of the input record, and must be wide enough to contain the input quantity. Since the value on the card is converted to double precision, both D and E are acceptable as the beginning of the exponent subfield; e. g., the Dw.d type number (-6228.4D5) has an integer, fraction, and exponent subfield.

Examples:

(1)
```
      DOUBLE PRECISION A,B,C
      READ 6,A,B,C
    6 FORMAT (3D18.7)
```

Input Card



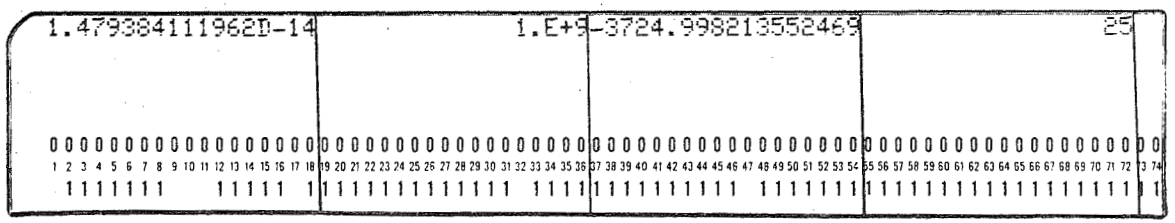
VI.A.3 (Cont.)

How many storage locations are allocated for B in the example above? _____.

Answer: 2

(2)
```
      DOUBLE W(4)
      READ 15, W
   15 FORMAT (4D18.7)
```

Input Card



Which of the quantities shown on the input card above are acceptable as double precision input? _____.

Answer: all

After input of the value for W(4), what value will be contained in storage? _____.

Answer: .0000025

Which subfields are present in the data for W(1), W(2), and W(3)?

W(1) _____

Answer: Integer, fraction, exponent

W(2) _____

Answer: Integer, exponent

W(3) _____

Answer: Integer, fraction

VI.A.4 Dw.d Output

The form of output for the D field specification is:

$$\pm a.aa....a \pm eee \qquad 100 \leq eee \leq 322$$

or

$$\pm a.aa....aD \pm ee \qquad 0 \leq ee \leq 99$$

where the "a's" are the most significant digits of the integer and fractional parts, and the "e's" are the digits in the exponent. If the sign is positive, the first position is blank. The field occupies w positions (print positions) of the output record. It must be wide enough to contain a sign, decimal point, and subfields of the output form.

Let's consider example (2) in section A.3 and assume that the input card has been read and values stored. The stored values printed by a D37.29 format specification would be:

```
bb1.47938411196200000000000000000D-14
bb1.00000000000000000000000000000D+09
b-3.72499821355246899999999999998D+03
bb2.49999999999999999999999999999D-06
```

Variables in the double precision mode are significant to approximately 29 digits. On output, part of this significance may be lost in the output conversion routine, and digits printed beyond the 28th place may not be reliable.

VI.A.4 (Cont.) In the example above, the last two answers are not exact. Consider the format specification D37.29. By this specification 30 digits were printed, and the limit of reliability was exceeded by two places. If .000025 had been printed by a specification D35.27, the last answer would have been

bb2.50000000000000000000000000D-06

In most cases you will not print the maximum number of digits, but restrict your output to a more suitable format such as the one in the example below.

The stored values printed by a D15.7 specification would be:

bb1.4793841D-14
bb1.0000000D+09
b-3.7249982D+03
bb2.5000000D-06

From the examples shown, note that Dw.d conversion and Ew.d conversion follow the same general rules.

VI.A.5 Work exercise VI.A in your workbook at this time.

VI.B Gw.d Input and Output

VI.B.1 Gw.d input specifications are the same as Fw.d specifications. (Reference IV.H)

VI.B.2 Gw.d output specifications are interpreted as F conversion or E conversion depending on the magnitude of the number to be printed, w, and d.

Gw.d is used for F conversion when $|n|$ (the number) $\leq 10^d$.

1. d indicates significant figures to be printed (not number of decimals)
2. w must be large enough to accommodate four blanks which are automatically inserted into the field right justified (these positions would contain the exponent subfield on an E-type conversion), the decimal point, d digits, and space for the sign. A field width of $w \geq d + 6$ is necessary for effective use by the F conversion.

VI.B.2 (Cont.)

Example:

In core storage A = 732.962, B = -24.75, and C = 1.7532. With an output specification of G9.4 the printed values appear as follows:

A = 733.0bbbb, B = *4.75bbbb, C = 1.753bbbb

w = 9, d = 4

The field width is less than d + 6. "B" cannot be output effectively because there is not space for the sign.

Gw.d uses the E conversion when $N > 10^d$.

You will recall the E conversion requires $w \geq d+7$. The same requirement is imposed upon the G specification.

In the example above if the output specification were G9.2 the output would be :

A = b7.33E+02, B = b-25.bbbb, C = bb1.8bbbb

VI.B.3 For the quantity 76543.21 which conversion type would be used for the following Gw.d output specifications? What would the output look like?

A. G9.0 _____

B. G12.6 _____

C. G12.2 _____

Answer: A. E bbbb8E+04

B. F b76543.2bbbb

C. E bbbb7.65E+04

VI.B.3 (Cont.)

The answers to the questions above are obtained as follows:

A. N = 76543.21 G9.0 w = 9 d = 0

w > d+7 but $76543.21 > 10^0$

Therefore, the E specification would be used. (E9.0)

$$\underbrace{\text{bbbb8E+04}}_{\text{w=9}}$$

B. N = 76543.21 G12.6 w = 12 d = 6

w > d+6 and $76543.21 < 10^6$

Therefore, the F specification is used. (with d significant figures)

$$\underbrace{\text{b76543.2bbbb}}_{\text{w=12}}$$

C. N = 76543.21 G12.2 w = 12 d = 2

w > d+7 but $76543.21 > 10^2$

Therefore, the E specification is used. (E12.2)

$$\underbrace{\text{bbbb7.65E+04}}_{\text{w=12}}$$

VI.B.4 Work exercise VI.B in your workbook at this time.

VI.C Lw, Input and Output

VI.C.1 The Lw specification is used for transmission of logical quantities. Again, w specifies the total width of the field. On input, the field is considered true or false if the first non-blank character in the field is T or F, respectively.

Three cards have punched on them in the first eight columns

A. bTRUEbbb
B. bFALSEbb
C. bbbbbbbT

What would be the value of the logical list item if these three cards are read by L8 specification?

A._____
B._____
C._____

Answer: A. True
B. False
C. True

VI.C.2 Should the input field be entirely blank, it is considered false.

Using an L6 specification, what value would be input for

A. bFbbbb _____
B. bbbbbb _____
C. bbbTbb _____

Answer: A. False
B. False
C. True

VI.C.3 When logical variables are to be output by an Lw specification, a T or F is output for true or false, respectively, right adjusted in a field filled with blanks.

If variable MAYBE is false, what is output for L2 specification?

Answer: bF

VI.C.4 Work exercise VI.C in your workbook at this time.

VI.D Scale Factors

Another format specification is the scale factor, nP. This specification permits scaling of values during I/O when used in conjunction with certain types of conversion.

VI.D.1 For input, scale factors have effect only on F-conversion. For output, scale factors may be used with the D, E, F, and G types of I/O conversion. The effects obtained from the use of the scale factor depend on the type of conversion and whether the operation is input or output.

The scale factor may be used with _____, _____, _____, or _____ conversions. Answer: D, E, F, G

The effect of the scale factor will depend not only on the type of conversion, but also on whether the operation is _____ or _____. Answer: input, output

A scale factor of zero is assumed if no value is given.

VI.D.2 Output

The format specification, nP, may be used as a separate entity in the FORMAT statement or it may appear as a prefix of any D, E, F, or G conversion. An important thing to remember is that once a scale factor is used, it will remain in effect for all D, E, F, and G conversions following the scale factor in the same FORMAT statement until a new scale factor is introduced.

To nullify the effect of a scale factor, a subsequent scale factor of zero in the same FORMAT statement must be specified by 0P.

A scale factor is effective for the output of only one value. True or false?
_____. Answer: False

VI.D.3 The scale factor may be entered in any of the following ways:

nP
nPFw.d
nPEw.d
nPGw.d
nPDw.d

where $-8 \leq n \leq 8$

In the FORMAT statement

20 FORMAT (2P, 3I2, F20.6)

a scale factor of 2 is established. Since scale factors do not apply to the _____ conversion, an equivalent statement would be

20 FORMAT (3I2, 2PF20.6) Answer: I

VI.D.4 Now consider the effect of the scale factor, n, on the values being converted. The following table illustrates this effect when used with the different conversion types.

F Conversion

Input The input value is multiplied by 10^{-n} during conversion. For example, a scale factor of one will cause a card value of 2.367 to be stored in the computer as .2367.

VI.D.4 (Cont.)

Output — The value in the computer is multiplied by 10^n before being transmitted to the I/O unit. For example, a scale factor of three will cause the value 2.367 to be printed as 2367.0.

E and D Conversion

Input — The scale factor is not effective in this case.

Output — The value is converted with n+1 digits to the left of the decimal point. The exponent is adjusted so that the value is not changed. For example, the value 23.68796 is to be printed using E conversion. The following table illustrates the effect of different scale factors.

Specification	Printed Value
E16.3	2.369 E+01
1PE16.3	23.688 E+00
2PE16.3	236.880 E-01
3PE16.3	2368.796 E-02
-1PE16.3	.237 E+02

G Conversion

Input — Same as F conversion.

Output — The effect of the scale factor is suspended when F-type conversion occurs. The scale factor is effective when the magnitude of the data requires that the E-type conversion be used. In this case, the effect is the same as for E conversion.

VI.D.4 (Cont.) Provide the correct values for the table below.

Specification	I/O	Computer Value	External Value	Answers:
2PF10.3	input	_____	326.475	3.26475
-1PF10.3	input	_____	326.475	3264.75
1PF10.3	output	3.26475	_____	32.648
-2PF10.3	output	3.26475	_____	.033
0PF10.3	output	3.26475	_____	3.265
2PE12.5	input	_____	b3.26475E+00	3.26475
-1PE10.4	output	3.26475	_____	b.3265E+01

VI.D.5 Example:

```
DATA K,A,B,C/27,-932.096, -.0075804, .55361/

      (No scale factor, E-conversion)                          (Printed Output)
      PRINT 102, K,A,B,C                  27   -9.3210E+02   -7.5804E-03   5.5361E-01
102   FORMAT (I3, 3E12.4)
      (Scale factor, 1P, E-conversion)
      PRINT 103, K,A,B,C                  27   -93.2096E+01  -75.8040E-04  55.3610E-02
103   FORMAT (I3, 1P3E12.4)
      (No scale factor, F-conversion)
      PRINT 104, K,A,B,C                  27   -932.096      -.008          .554
104   FORMAT (I3, 3F11.3)
      (Scale factor, 1P, F-conversion)
      PRINT 100, K,A,B,C                  27   -9320.960     -.076         5.536
100   FORMAT (I3, 1P3F11.3)
      (Scale factor, -1P, F - conversion)
      PRINT 101, K,A,B,C                  27   -93.210       -.001          .055
101   FORMAT (I3, -1P3F11.3)
```

VI.D.5 (Cont.) Example:

Array AA contains numerical values of a physical quantity that is usually expressed as a number x 10^5.

```
DIMENSION  AA(5)
DATA  AA/-932.096, -600.47, 1000.03, 24575., 738407./
DATA  K,B,C/27,-.0075804, .55361/
(Scale factor, -5P, used, cancelled, and reused.)

PRINT 200, K, AA(1), B,C, (AA(I), I=2, 5)

200 FORMAT(I3,-5PF10.3,6Hx10**5, 0PE12.4,E12.4/(-5PF13.3))
(Printed output)

27    -.009x10**5    -7.5804E-03    5.5361E-01
      -.006
       .010
       .246
      7.384
```

VI.D.6 Work exercises VI.D in your workbook at this time.

VI.E Group Specifications

VI.E.1 Groups of specifications may be repeated by enclosing the group in a set of parentheses, and preceding the set by the integer constant required. Should it be desired to print out an integer, then a real number, followed on the same line by another integer and real number, the specifications might be

```
n   FORMAT  (2 (I4, 2X, E15.7, 2X) )
```

which is equivalent to

```
n   FORMAT  (I4, 2X, E15.7, 2X, I4, 2X, E15.7, 2X)
```

What is the equivalent form of

```
n   FORMAT  (2XF5.1, 2XF5.1, 2XF5.1, 2X)
```

Answer:

```
n   FORMAT  (3 (2XF5.1), 2X)
```

VI.E.2 There is another way to cause specifications to be repeated: the innermost set of specifications in a FORMAT statement which is enclosed in parentheses without a repetition factor preceding it will be repeated as often as necessary to satisfy the corresponding I/O list. This set is known as an unlimited group, and any specifications to the right of an unlimited group will never be reached.

VI.E.2 (Cont.) Which are the unlimited groups in the following:

A. n FORMAT (I5, 2E10.4, (2I5)) _____

B. n FORMAT (2 (3XA10)/(1XE15.5)) _____

C. n FORMAT (2 (F7.2), 3(I4), (E10.6), I4) _____

Answer: A. (2I5)

B. (1XE15.5)

C. (E10.6)

VI.E.3 The right parenthesis of an unlimited group acts as a slash; that is, when the specification is repeated, it is on a new line or card. The specifications

n FORMAT (3I5, (F10.2))

if used to print out ten list items would cause four items to be printed on one line in 3I5, F10.2 format, then the next six to follow, each on a new line in F10.2 format. A total of seven lines would be printed.

How many lines would be required to print out a list of ten items by the following formats?

A. n FORMAT (I4, 2X, 4I3/(1XE15.7)) _____

B. n FORMAT (I4, 2X, 4I3, (1XE15.7)) _____

C. n FORMAT (I4, 2X, 4I3, 1X, E15.7) _____

Answer: A. 6

B. 5

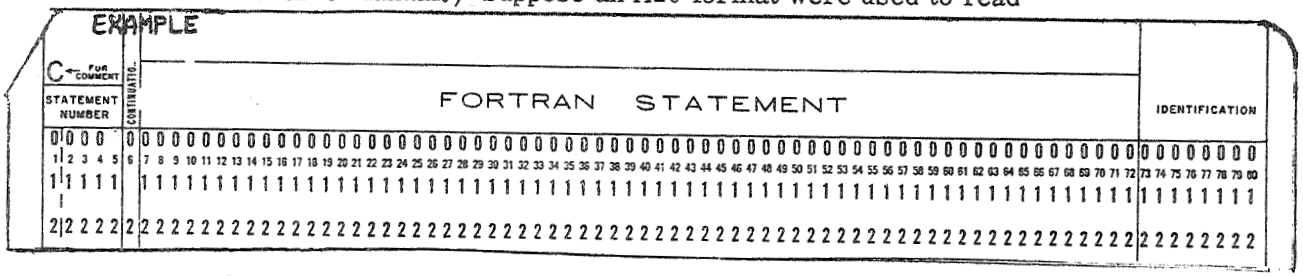
C. 2

VI.E.4 Note that in part C, the entire set of format specifications forms the unlimited group.

VI.E.5 Work exercise VI.E in your workbook at this time.

VI.F Aw, Input

VI.F.1 Aw conversion on input reads FORTRAN (alphanumeric) characters from a card and converts them to the 6000 Series console display code for storage in the computer. (The 6000 Series console display code is listed in Appendix A of the Control Data 6400/6500/6600 FORTRAN Reference Manual.) Suppose an A10 format were used to read



The quantity stored in the computer would be

5555550530011520 1405

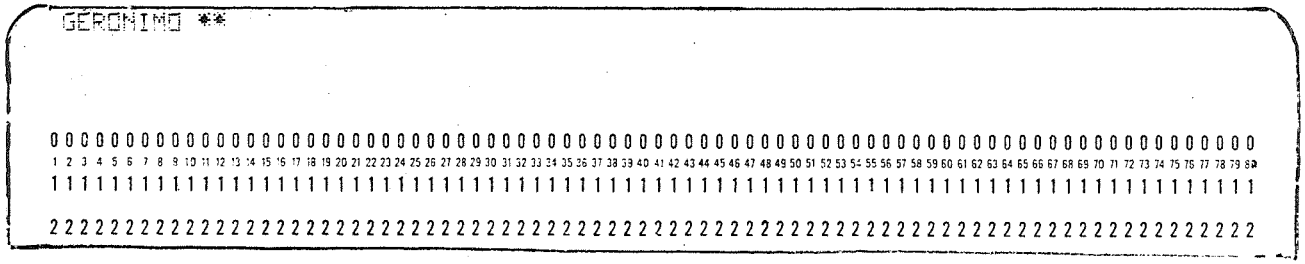
b b b E X A M P L E

Since the 6000 Series console display code requires two octal digits to represent a single FORTRAN character, how many such characters may be contained in a single CDC 6000 Series computer word? _____

Answer: 10

VI.F.2 If w in an Aw input conversion is greater than 10, the rightmost ten characters of the field will be input. If w is less than 10, the w characters will be stored left adjusted in the computer word and the right end of the word will be filled out with blanks.

What will be stored when



is read by

A. A10 _____
B. A12 _____
C. A3 _____

Answer: A. bGERONIMOb
B. ERONIMOb**
C. bGEbbbbbbb

VI.F.3 Work exercise VI.F in your workbook at this time.

VI.G Aw, Output

VI.G.1 Aw output converts internally-stored display code to FORTRAN characters for printing or punching. If w is greater than 10, the 10 characters will be output right adjusted and the field filled with blanks. If w is less than 10, the left-most w characters will be output.

Stored in the computer word is

55230520240515020522
b S E P T E MB E R

What would be output for specification

A. A12 _____
B. A5 _____
C. A10 _____

Answer: A. bbbSEPTEMBER
B. bSEPT
C. bSEPTEMBER

VI.G.2 Work exercise VI.G in your workbook at this time.

VI.H Rw, Input and Output

VI.H.1 Rw conversion on input reads FORTRAN characters from a card and converts them to console display code for storage in the computer.

For input, when w is less than 10, the quantity will be right adjusted and the remainder of the word filled with octal zeros.

If the punched card has CENTIMETER punched in columns 1-10, what will be stored in the computer if the card is read by an R4 specification? _____

Answer: 0000000000003051624
C E N T

VI.H.2 Rw output converts internally-stored display code to FORTRAN characters for printing or punching.

For output, when w is less than 10, the rightmost w characters will be output.

If the computer word contains

03051624111505240522
C E N T I M E T E R

what will the output be for specification R5? _____

Answer: METER

VI.H.3 Work exercise VI.H in your workbook at this time.

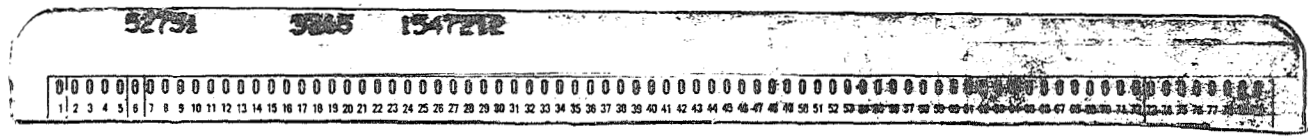
VI.I Ow, Input and Output

VI.I.1 Octal integers may be input by the O conversion. The field width is determined by w which must be ≤ 20.

Example:

```
      READ 7, PSI, THETA, OMEGA
    7 FORMAT (3O10)
```

The following card contains the data which is read by the above READ statement.



How will the values appear in storage?

A. PSI _____

B. THETA _____

C. OMEGA _____

Answer:

A. 0000000000000052751

B. 0000000000000003265

C. 0000000000001547212

VI.I.2 Octal integers are output by the O conversion with field width w. If $w \leq 20$, the rightmost w octal digits will appear; if $w > 20$, the number will be right adjusted and the field filled with blanks.

How will the quantity 72316400450511314277 appear output by

A. O20 _____

B. O25 _____

C. O15 _____

Answer:

A. 72316400450511314277

B. bbbbb72316400450511314277

C. 400450511314277

VI.I.3 Negative numbers will appear in their 1's complement form under O-type conversion. For example, if the quantity - 3 is punched on a card to be read under O-type conversion, it would appear in storage as 7777777777777777774

What would the internal number look like for -265 read by O4 specification? _____.

Answer: 7777777777777777512

VI.I.4 Work exercise VI.I in your workbook at this time.

VI.J Variable Formats

In section IV you learned to write FORMAT statements. As you remember, these statements are not executable. That is, they do not cause any computer operations to be generated. They are used only as a source of information for the I/O statements. In this section you will learn to read this information into the computer just as you would read values for different variables.

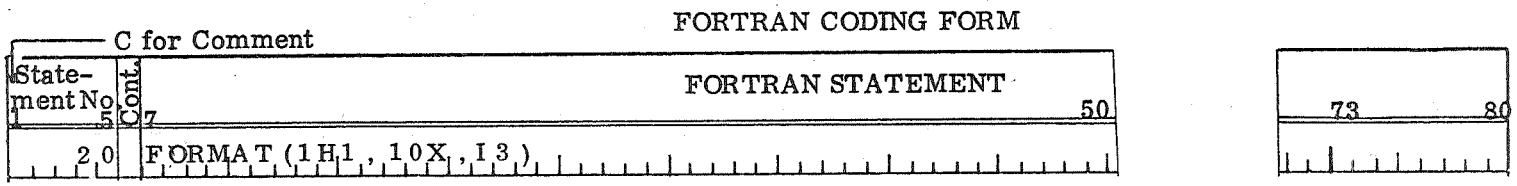
VI.J.1 The information in a FORMAT statement is stored and then interpreted at the time the I/O takes place.

A FORMAT statement is not _____ . It is used to supply information for _____ statements.

Answer: executable
Answer: I/O

VI.J.2 The information from the FORMAT statement which is stored begins with the first parenthesis after the word FORMAT and ends with the last parenthesis in the statement.

What information is stored from the following FORMAT statement ?



_____

Answer: (1H1, 10X, I3)

VI.J.3 This information is stored in the computer in console display code. Thus, each character including blanks requires six binary digits for its unique representation.

Since each computer word contains sixty bits and six are required for each character when represented in the console display code, each computer word can contain as many as _____characters. Answer: 10

VI.J.4 Suppose that the characters (1H1, 10X, I3) are punched in the first twelve columns of a computer card. These characters may be read into the computer in console display code by reading two variables according to the format specifications (A10, A2).

Information may be read into the computer in console display code by use of the_____format specification. Answer: A

VI.J.5 You now know two facts which are necessary to the reading of formats at execution time. These are:

1. The information contained in a FORMAT statement is stored in console display code and is not used until the I/O takes place.

2. Information read into the computer according to the A format specification is stored in console display code.

VI.J.6 When a format is required, an I/O statement usually references the FORMAT statement by its number. However, the I/O statement may use the name of an array instead of the FORMAT statement number. When this is done, the I/O statement will assume that the array contains the I/O format specifications in console display code.

EXAMPLE:

```
READ FMT, X, A, B, C
```

FMT is the name of the array which contains the format specifications.

In most cases, the I/O statement refers to a particular format by means of the _____ _____ associated with the format.

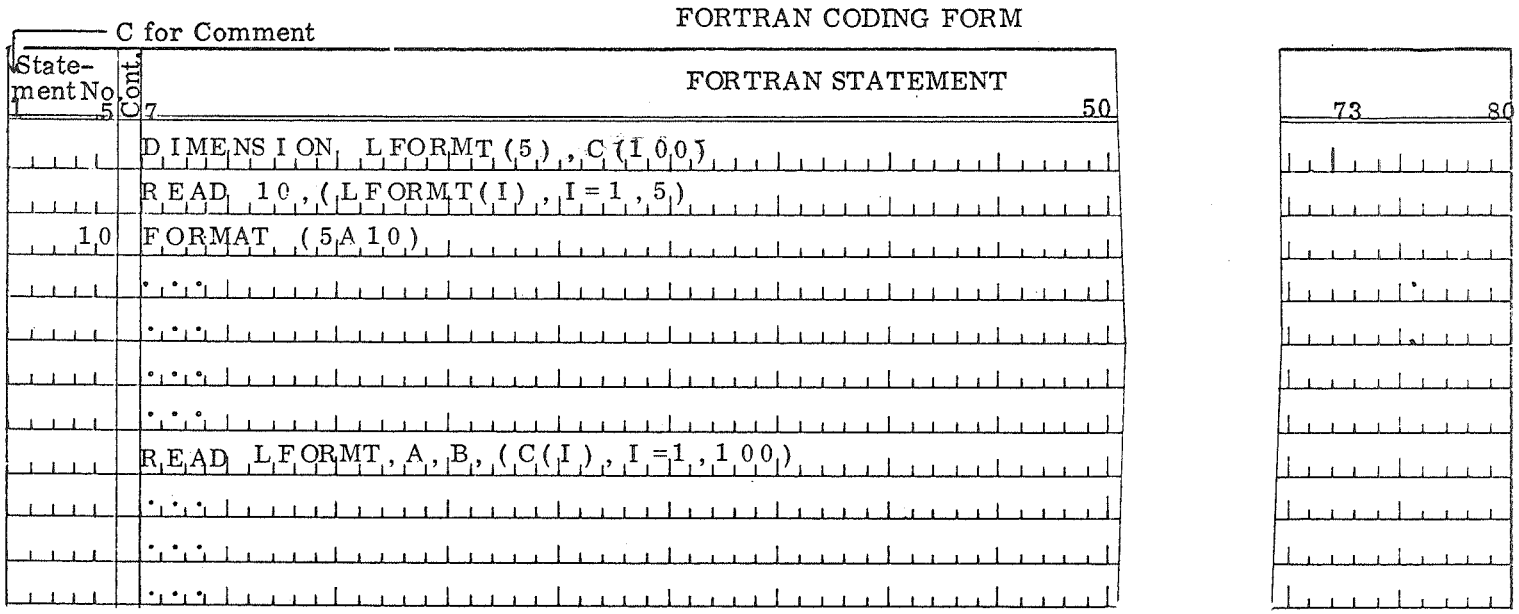
Answer: statement number

VI.J.7 When an array contains the format specifications, the reference to the array in the I/O statement is not required to be the first location in the array, but must reference the start of the format specifications.

The format reference in an I/O statement may be either a statement number or an _____ _____.

Answer: array name

VI. J. 8 Format specifications read in at execution time are referred to as variable formats. This means that the formats may be changed on any given run. The following will illustrate how variable formats are used.

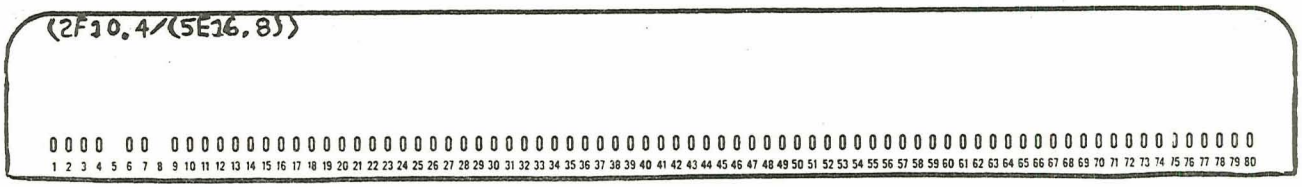


The first card read must contain format specification in columns 1 through _____. The specifications must begin with a _____ and end with a _____. These specifications are then used to read the values of the variables _____, _____, and _____.

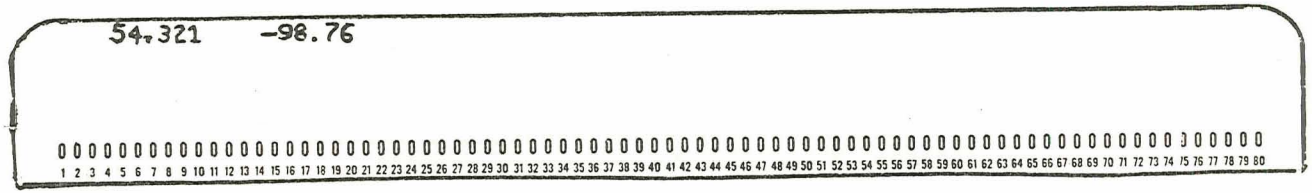
Answer: 50
parenthesis
parenthesis
A, B, C

VI.J.9 Following are examples of the input data required for the example in VI.J.8.

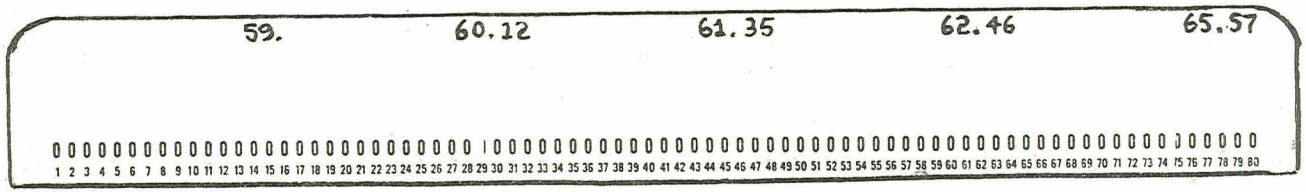
Card 1 is read by FORMAT(5A10) into LFORMT. Words 3, 4, and 5 of LFORMT would contain blanks.



Card 2 is read by 2F10.4 of LFORMT into A and B.



Cards 3 - 22 are read by the 5E16.8 of LFORMT into the 100 locations of the array C.



VI.J.10 Work Exercise VI.J in your workbook at this time

VI.K ENCODE/DECODE Statements

The use of formatted I/O statements provides conversion of data from binary to console display code on output and the reverse conversion on input. The use of ENCODE/DECODE statements permit the same conversions, but the results remain within the computer memory.

VI.K.1

The ENCODE statement has the form

```
ENCODE (N, I, ALPHA) LIST
```

where

N is the number of six-bit console display code characters in each resulting record. This value may be an integer constant or a simple variable.

I is a format statement number or an array name which contains the format.

ALPHA is the beginning location where the converted information will be stored.

LIST is an I/O list.

For example,

```
      DIMENSION AMAT(5), K(2)
      ENCODE (20, 5, AMAT)K
    5 FORMAT (2I9)
```

will convert the two integer values in the K array to 20 characters in console display code. This representation of the values is then stored in AMAT(1) and AMAT(2), ten characters in AMAT(1) and ten characters in AMAT(2). The last two characters in AMAT(2) will be blanks since the format only specifies 18 characters.

VI. K. 1 (Cont.) Consider the statements

FORTRAN CODING FORM

C for Comment

Statement No. 1–5	Cont.	7 FORTRAN STATEMENT 50	73–80
		DIMENSION KTAB(6),TABLE(12)	
		ENCODE (30,15,KTAB)(TABLE(J),J=1,6,2)	
15		FORMAT (F16.4/2F13.5)	

Each record of converted information will contain _____ characters. Answer: 30

There will be _____ records generated. Answer: 2

The remainder of the first record will be filled with _____ characters. Answer: blank

The second record will contain _____ values from the array TABLE. Answer: 2

VI.K.2 In the previous code, there are two records generated because of the slash in the FORMAT statement. Since the FORMAT statement requires only one value for the first record, the value used is TABLE(1). If the FORMAT statement requires fewer characters than the number called for in the ENCODE statement, blanks are added to meet the requirements of the ENCODE statement. Further, a new record is always started at the beginning of a computer word. Therefore, when the number of characters specified by the ENCODE statement is not a multiple of ten, enough blank characters are added at the end of each record to complete the computer word.

If the FORMAT statement requires twelve characters and the ENCODE statement specifies fifteen characters, _____ blank characters are required to complete the record. Since fifteen is not a multiple of ten, _____ blank characters are needed to complete the last computer word.

Answer: 3, 5

VI.K.3 To illustrate the use of the ENCODE statement, assume that:

```
A(1) = 3.14159265
A(2) = 666.33333
  Z  = 466.2743
ZB   = 425.999999
  K  = 125
```

These values will now be converted to console display code and stored into the array B according to a specified format.

VI.K.3
(Cont.)

Case 1.

```
      DIMENSION B(4)
      ENCODE (38, 10, B) A(1), A(2), Z, ZB, K
10    FORMAT (F7.4, E12.5, F8.3, F7.2, I4)
```

After execution of the ENCODE statement, the B array will contain the following console display code characters.

word 1	b	3	.	1	4	1	6	b	6	.
word 2	6	6	3	3	3	E	+	0	2	b
word 3	4	6	6	.	2	7	4	b	4	2
word 4	6	.	0	0	b	1	2	5	b	b

Case 2.

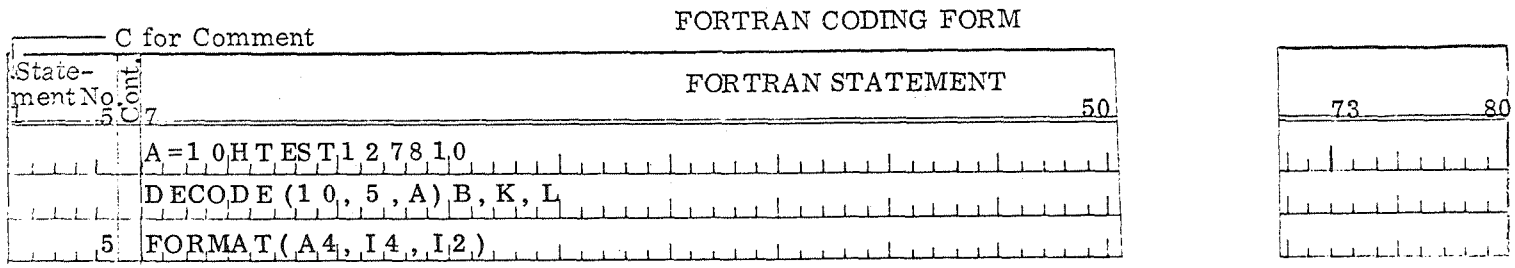
```
      ENCODE (26, 20, B) ZB, K
20    FORMAT (F16.4, I10)
```

After execution of the ENCODE statement, the B array will contain the following console display code characters.

word 1	b	b	b	b	b	b	b	b	4	2
word 2	6	.	0	0	0	0	b	b	b	b
word 3	b	b	b	1	2	5	b	b	b	b

In both cases, notice the correspondence between the characters stored in the B array and the characters which would be printed if the same variables were printed using the same FORMAT statement.

VI.K.4 The DECODE statement takes the console display code information which begins at a given location and assigns this information to the list variables according to a given format specification. For example,



will set B = TESTbbbbbb in console display code
K = 1278 in binary (integer)
and L = 10 in binary (integer).

In this example, a _____-character record is being decoded. This record starts at location _____. In fact, it is completely contained in location A.

Answer: 10, A

VI.K.5 Consider now the same two cases which were used in VI.K.3. This time we will start with the console display code characters in the B array and DECODE the information back to numerical values. Notice that the values obtained will not be the exact values we started with. This is due to the rounding and truncation which occurred during the ENCODE process.

Case 1. B Array

word 1	b	3	.	1	4	1	6	b	6	.
word 2	6	6	3	3	3	E	+	0	2	b
word 3	4	6	6	.	2	7	4	b	4	2
word 4	6	.	0	0	b	1	2	5	b	b

```
      DECODE (38, 30, B) A(1), A(2), Z, ZB, K
   30 FORMAT (F7.4, E12.5, F8.3, F7.2, I4)
```

This will convert the characters in the B array according to the format specified.

```
A(1) = 3.1416
A(2) = 6.66333E+02                    (666.333)
  Z  = 466.274
ZB   = 426.00
  K  = 125
```

Compare these values with the values in VI.K.3.

VI.K.5 (Cont.)

Case 2.

B Array

word 1	b	b	b	b	b	b	b	b	4	2
word 2	6	.	0	0	0	0	b	b	b	b
word 3	b	b	b	1	2	5	b	b	b	b

```
      DECODE (26, 20, B) ZB, K
   20 FORMAT (F16.4, I10)
```

In this case

```
ZB = 426.0000
 K = 125
```

VI.K.6 Work exercise VI.K in your workbook at this time.

VI.L. Unformatted I/O (binary)

When writing numeric information on the printer, the binary values in the computer must be converted to console display code according to certain format specifications. Numeric values entered on the card reader must also be converted according to format specifications. Unformatted I/O makes it possible to move values between the computer and certain peripheral units without any conversion.

VI.L.1 Suppose that one program is to compute values which will be used as input to some later program. If a large number of values are involved, they should be written on magnetic tape.

One way of saving values which have been computed is to write them on _____ _____.

Answer: magnetic tape

VI.L.2 As was mentioned in part IV, formatted information may be written onto magnetic tape. When it is formatted, the conversion must take place both on output and then again when the tape is used as input to another program.

The use of unformatted I/O saves conversion time on both _____ and _____.

Answer: input, output

VI.L.3 The use of unformatted I/O when applicable eliminates the loss of accuracy which often occurs during conversion.

Unformatted I/O preserves the complete sixty-bit word as well as reducing computer _____.

Answer: time

VI.L.4 Unformatted I/O can be used with any peripheral unit which is capable of storing the entire sixty-bit computer word. These units include magnetic tape, disks, drums, and other special storage devices.

Unformatted WRITE statements cannot be used to output information on the _____.

Answer: printer

VI.L.5 Statements for reading or writing unformatted information are similar to formatted reads and writes. In the unformatted case, the reference to the format is omitted. For example, where "i" is the unit number and "n" is the FORMAT statement number,

READ (i, n) LIST
becomes READ (i) LIST in the unformatted case.

Write the statement which will output the first twenty values from the array AMAT. Write these values on I/O unit 10 in unformatted form._____

Answer:
WRITE(10)(AMAT(I),I=1, 20)

VI.L.6 The type of variables being transferred is not important to the I/O statement since it moves the words without disturbing the binary configuration. If the values are used as input to another program, it is important to make sure that the values are considered to be of the same type as they were in the original program.

If the second word written is real and the third word written is integer, the second and third words should be treated as _____ and _____ respectively when this information is read into the computer.

Answer: real, integer

VI.L.7 Work exercise VI.L in your workbook at this time.

VI.M Data Files

Let us define a logical record as the information read or written by one READ or WRITE statement. A data file will mean a set of logical data records.

VI.M.1 A data file will normally contain data records which are related in some manner, but the collection of records which constitute a file is completely arbitrary.

One WRITE statement creates _____ data record on tape or other output device.

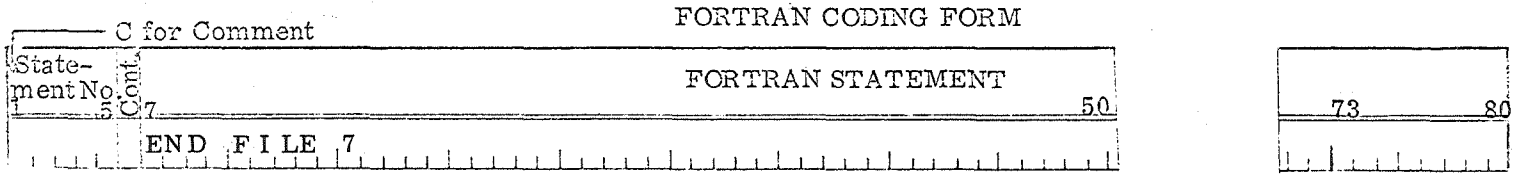
Answer: one

VI.M.2 Data files are created by writing an end-of-file indicator at the place you wish to end the file. This is accomplished by the statement

END FILE i

This causes an end-of-file to be written on unit i. Sections VI.M.6, 7, and 8 will explain the usefulness of the end-of-file indicator.

The statement



will cause an end-of-file indicator to be written on I/O unit _____.

Answer: 7

VI.M.3 Before writing your first data record onto a magnetic tape, make sure that the tape is positioned at the beginning of the tape (load point). This is accomplished by the statement

REWIND i

where i is the I/O unit.

The use of the REWIND instruction before reading a tape will insure that the _____ (first, last) record on the tape will be read by the first READ statement.

Answer: first

VI.M.4 The statement

BACKSPACE i

will cause the tape to move back one record from its present position.

If the read head of tape 8 is positioned immediately at the beginning of the fourth record, the execution of a BACKSPACE 8 instruction will position the tape such that the read head is immediately at the beginning of the _____ record.

Answer: third

VI.M.5 Both the REWIND and BACKSPACE statements are ignored if the tape is already positioned at load point (beginning of the tape).

The REWIND statement causes the tape to rewind until it reaches the _____ _____.

Answer: load point

VI.M.6 When reading information from tape, it is desirable to know when the end of a data file is reached. This is determined, of course, by the presence of the end-of-file indicator.

The end of a data file is signaled by the reading of an _____ indicator.

Answer: end-of-file

VI.M.7 After a READ statement, there are three different statements which may be used to determine if an end-of-file indicator was read. They are:

IF (ENDFILE i) N_1, N_2
IF (EOF, i) N_1, N_2
IF (IOCHECK, i) N_1, N_2

The i is an I/O unit and N_1 is the statement number to which control is transferred if unit i read an end-of-file indicator on its last read operation. Control goes to N_2 if no end-of-file was found.

Write a statement to read an unformatted record from tape unit 9 in array AMAT. Assume that AMAT has been dimensioned 100 and that the record length on the tape is 100 words.

Answer:
READ (9)(AMAT(I), I=1, 100)
or READ (9) AMAT

Write a statement to determine if the above READ statement actually read the data record or encountered an end-of-file indicator. Go to statement 10 if a data record was read. Go to statement 20 if an end-of-file was encountered.

Answer:
IF (EOF, 9) 20, 10
or IF (ENDFILE 9)20, 10
or IF (IOCHECK, 9)20, 10

VI.M.8 As a review, write the six new statements which have been introduced in this section.

_____

_____

_____

_____

_____

_____

Answer:
END FILE i
REWIND i
BACKSPACE i
IF (ENDFILE i) N_1, N_2
IF (EOF, i) N_1, N_2
IF (IOCHECK, i) N_1, N_2

VI.M.9 Work exercise VI.M in your workbook at this time.

VI.N NAMELIST Statement

A list of variables for input or output may be assigned a name, thus, the term NAMELIST. The assigned name may later be used as a shorthand method for referencing all variables in the list.

VI.N.1 The form of the NAMELIST statement is

$$\text{NAMELIST/NAME1/}a_1, a_2, \ldots, a_j\text{/NAME2/}b_1, b_2, b_3, \ldots, b_n$$

where the a_i and b_i are simple variables or array names.

For example,

FORTRAN CODING FORM

C for Comment

State-ment	C	FORTRAN STATEMENT (5 … 7 … 50)	73 … 80
		DIMENSION A(3),K(10),X(20)	
		NAMELIST/AMAT/A,K,X,Y	
		NAMELIST/SEC/R,P	

assigns the name AMAT to the list of variables A, K, X, and Y. The name SEC is assigned to the list of variables R and P.

A list of variables to be used for input or output may be given a name by use of the NAMELIST statement. True or false? _____

Answer: True

Assuming that the variables Z and ZZ have been dimensioned, write a NAMELIST statement which will associate these variables with the name ZLIST. _____

Answer:
NAMELIST/ZLIST/Z,ZZ

VI.N.2 The NAMELIST statement must conform to the following rules.

a.) The NAMELIST name must conform to the same rules as variable names.

b.) The NAMELIST name is enclosed by slashes and followed by a list of variable names separated by commas. This sequence may then be repeated with other NAMELIST names and lists if desired.

c.) The NAMELIST statement must appear prior to its use in an I/O statement.

d.) The NAMELIST name may appear only in I/O statements.

e.) A dimensioned declaration of an array used in a NAMELIST must precede the NAMELIST statement.

A NAMELIST name may appear in a mathematical computation. True or false?_____

Answer: False

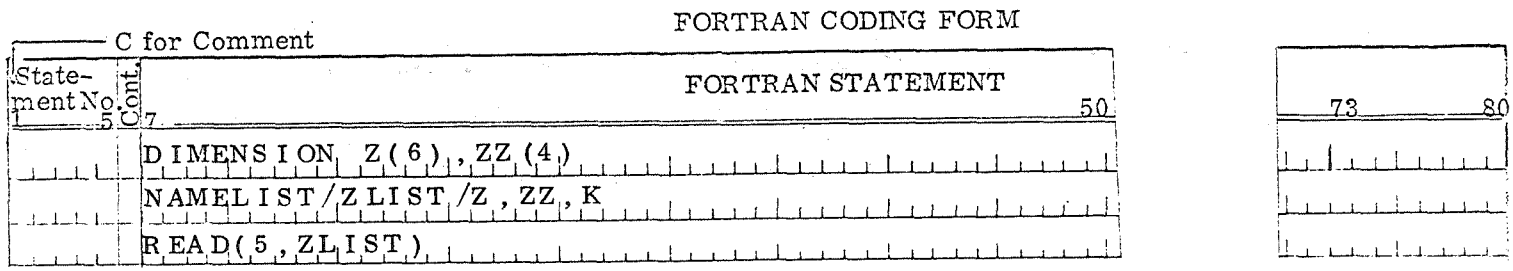
VI.N.3 Now that you know how to write NAMELIST statements, the next step is to learn how they can be used. The form of the READ or WRITE statement using NAMELIST is

```
READ (i, NAME)     or
WRITE (i, NAME)
```

where i is the I/O unit and NAME is a NAMELIST name. The NAMELIST statement must _____ the use of the NAMELIST name in an I/O statement.

Answer: precede

VI.N.4 To illustrate the use of the NAMELIST, consider the statements



These statements can be used to read values into as many of the locations in Z and ZZ as desired. This is determined by the form of the information which is actually read. This information must start with the character $ in column 2, immediately followed by the NAMELIST name, followed by a blank. This in turn is followed by the data and ended by $END. Each data item is composed of an array name, array item, or simple variable along with the values to be read for those variables. To show the exact form, suppose that values of Z(2), Z(5), K and all values of ZZ are to be input from cards. The cards might have the following form:

Card 1 b$Z LISTbZ(2)=25.6, Z(5)=6.2, K=3,

Card 2 bZZ=1.0, 3.4, 2*2.5, $END

The following rules apply to NAMELIST data cards.

1.) Column 1 must always be left blank.
2.) No space is left between the character $ and the NAMELIST name.
3.) A comma must follow each constant data value. This means that the constant data values must be separated from each other and from a following variable name.

VI.N.4 (Cont.)

4.) A blank must follow the NAMELIST name.

5.) If more than one card is required for data, each card except the last must end with a constant followed by a comma.

6.) Columns 2-80 may be used.

7.) The asterisk is used as a repeat symbol as was the case in the DATA statement.

8.) The variable names in the input do not have to appear in the same order as they appeared in the NAMELIST statement.

9.) In a NAMELIST record, the mode of the variable name supercedes the mode of the value.

10.) A $ or $END (either form is correct) ends a NAMELIST record.

11.) No information other than data may appear on a NAMELIST data card. (i. e., Serial numbers, identification, etc.)

Write a data card which will assign 36.0 to ZZ(3) when read by the three statements above. _____

Answer:
b$ZLISTbZZ(3)=36.0,$END

VI.N.5

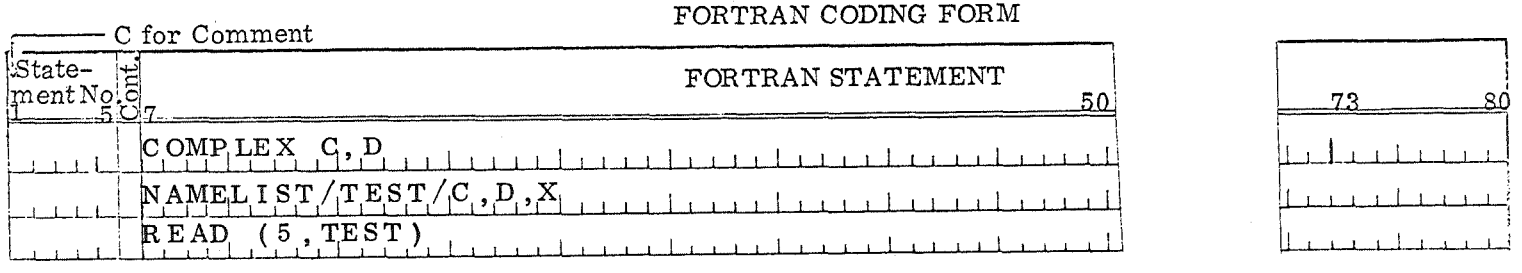
The use of the NAMELIST statement for output will output all values in the list in a format such that it may be read at a later time using the same NAMELIST.

The statement WRITE(i, ZLIST) will write all variables named in ZLIST in a format which can be read by the statement READ(i, ZLIST). True or false ?_____.

Answer: True

VI.N.6 The variable types which may be included in a NAMELIST statement are integer, real, complex, double precision, and logical. The appropriate type declaration must precede its use in a NAMELIST statement. The constants input through the use of the NAMELIST must agree in type with the associated variable.

For example,



defines the complex variables C and D as NAMELIST variables.

When used with the previous statements, the following data card will set X = 25.4 and the complex variable D to (3.2 + 12.1i).

b$TESTbX=25.4, D=(3.2, 12.1) $END

VI.N.7 Work exercise VI.N in your workbook at this time.

VI.O Program Files

In Chapter IV we learned how to present data to the computer on punched cards or magnetic tape, and how to receive data from the computer on the printer, punched cards or magnetic tape. This transfer of data occurred during the execution of a program, according to I/O control statements and FORMAT statements written by the programmer. Now perhaps you wonder how the program itself gets into the computer--certainly a vital piece of knowledge, if the program is ever to be executed!

VI.O.1 Let us first consider the make-up of a typical FORTRAN program. The basic building block is one line of coding which, since it will be punched onto a single card, may be thought of as a card image. This card image may be any acceptable FORTRAN statement, continued statement, END card, or comment.

A card image is a single _____ ___ _____. Answer: line of coding

VI.O.2 Several lines of coding, or card images, grouped together (and following applicable FORTRAN rules!) make up a single main program or a single subprogram. As you have learned previously, the end of a single program is signified by an END card.

Is END an executable FORTRAN statement? _____. Answer: No

VI.O.3 For various reasons, it is desirable to separate some often-used coding into subprograms, and one main program may call upon many subprograms during the course of execution. A main program together with all its required subprograms may be thought of as a file of executable programs.

A program is made up of several _____ _____. Answer: card images

VI.O.4 Several high powered programs govern the course of action which the computer will follow. These routines are called executive routines or system routines, and are used for reading in other programs--your programs--compiling these programs into executable code and executing these programs.

Executive, or systems, routines are computer _____. Answer: programs

VI.O.5 The computer hardware itself is constructed so as to be able to read the necessary systems routines from magnetic tape, disc or other I/O device, and then give control of execution to these routines. This process is sometimes called bootstrap.

Bootstrap refers to the process of getting _____ programs into the computer. Answer: executive

VI.O.6 Just as your program will probably read in data, certain executive routines will also read in data; but, their "data" is your program itself. And just as your program operates upon its data in specified ways, the executive routines operate upon your program in specified ways; their operations consist of compiling (translating into executable code), error detection, loading into core, listing, punching etc.

Some systems routines use _____ as data.

Answer: programs

VI.O.7 Systems routines are capable of doing many things with your program, but you may specify just which systems operations you want performed on your program. You do this by means of control cards.

A special kind of data card for the executive routines, telling what system operations are desired, is the _____ _____.

Answer: control card

VI.O.8 Various control cards relay different types of information to the systems routines. For example, one card is for accounting purposes: it tells job number, priority, etc. Another card tells that the FORTRAN compiler will be needed. Another card may specify which input or output files will be used.

Executive routines can read many types of control cards. True or false? _____.

Answer: True

VI.O.9 Every FORTRAN main program to be compiled requires a program card which specifies the name of the program and the I/O files it will need to use.

The program card specifies the _____ and the _____.

Answer: program name, I/O files

VI.P System Files

We have previously discussed data files and program files. A system file, like the other type files, will denote a particular set of information which is related in some manner. The requirement for information to belong to a particular system file is that the information be input or output on a particular I/O file.

VI.P.1 To illustrate system files, consider a small program which reads some information from cards, performs calculations, and prints the results. Such a program makes use of two system files. First, the set of all information to be read by the card reader constitutes a system input file and second, the set of all information output to the printer makes up a system output file.

The set of all information transmitted to or from a particular I/O file is called a _____ _____.

Answer: system file

VI.P.2 The two system files for the card reader and printer are used in almost all programs. Because of this frequent use, they have been assigned file names of INPUT and OUTPUT respectively.

The system file name for the card reader is _____.

Answer: INPUT

VI.P.3 The program card which immediately precedes your main program must have the form

PROGRAM name (file names)

where "name" is the name by which you want to refer to your program and "file names" specify the systems file names a program requires.

The initial P of PROGRAM should be punched in column 7 of the card.

Program card

PROGRAM MAIN (INPUT, OUTPUT)

will name the following program _____, and will call for how many I/O files? _____.

Answer: MAIN, two

VI.P.4 When INPUT and OUTPUT are used to specify I/O files on a program card, they refer to the standard units for card reader and printer, respectively. The system will recognize these references without any further information.

The standard files for card reader and printer are called _____ and _____.

Answer: INPUT, OUTPUT

VI.P.5 If you wish to use other I/O files in addition to or in place of INPUT and OUTPUT, they must be listed on the program card. As an example, suppose the program WORK will read data from cards, write results on the printer, use one file for "scratch" computations during execution, and will read data from a special data tape. The program card might look like

```
      PROGRAM WORK (INPUT,OUTPUT,TAPE5,TAPE1 )
```

How many I/O files are called for by the program card ?

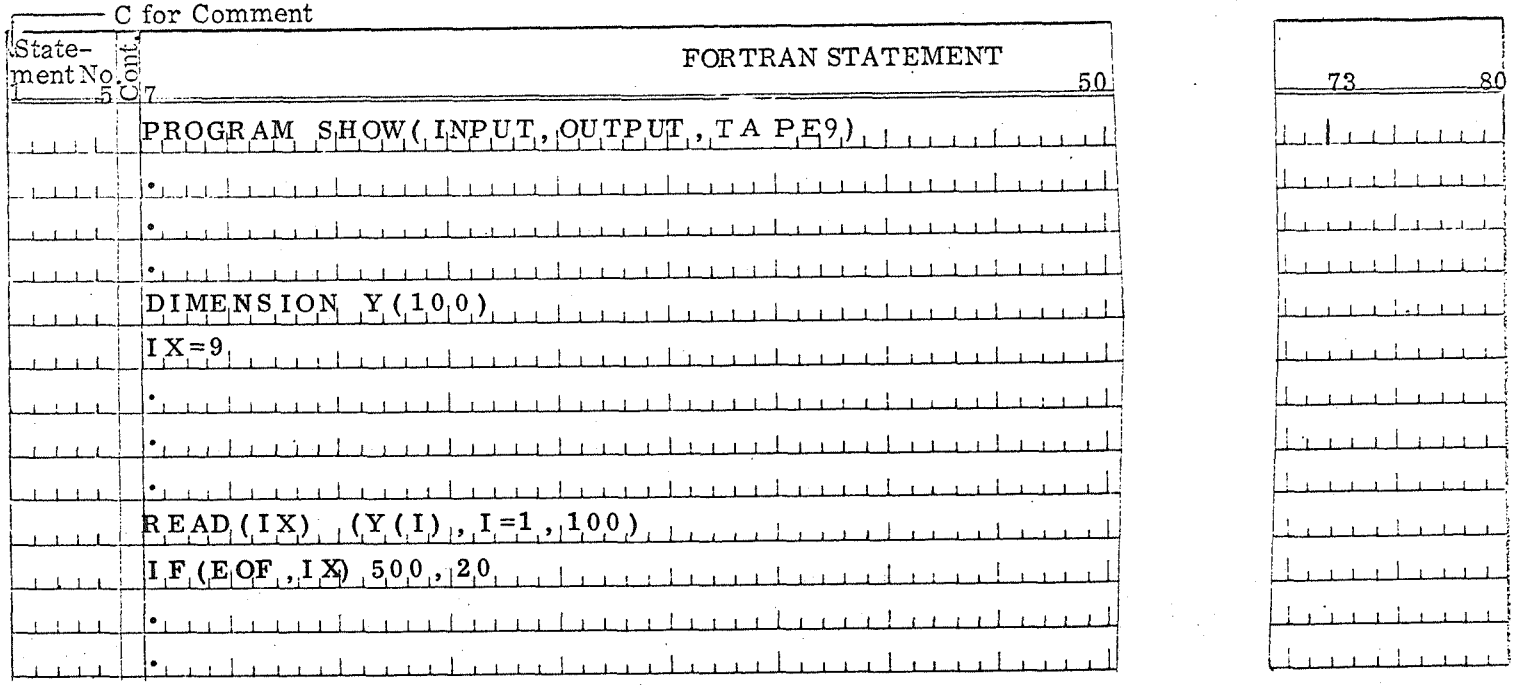
```
      PROGRAM UNAMIT (INPUT,OUTPUT,TAPE7,TAPE15,TAPE45)
```

_____.

Answer: Five

VI.P.6 All references to unit 9 in the FORTRAN program will use the system I/O file TAPE9. References to unit 7 will use system I/O file TAPE7, etc. If an integer variable is assigned a unit number and then used in the I/O reference, that variable name would be used to designate the appropriate system I/O file.

As an example, consider the system I/O file TAPE9 which is shown in the PROGRAM card below. The variable IX must be assigned a unit number prior to its use in an I/O statement. The unit number assignment may be through input, DATA statement or calculation. The following code will illustrate this procedure.



VI.P.6 (Cont.) Indicate the name of the third system I/O file if the previous coding is modified by removing the statement IX = 9 and rewriting the READ statement as

```
READ (9)  (Y(I), I = 1,100)
```

Answer: TAPE9

VI.P.7 When tapes other than the standard input and output units are used, they must have further identification, in addition to being listed on the PROGRAM card. See your instructor for the formats of all control cards.

VI.P.8 Work exercise VI.P in your workbook at this time.

VII. PROGRAMMING TECHNIQUES

VII. A. With the use of the FORTRAN tools now mastered, it is possible to write computer programs of many degrees of complexity. At this point, it is desirable to give some thought to the overall planning and execution of problem solving on computers. Generally, to solve a problem, the programmer must do four things:

1. Identify (clearly express) the problem
2. Outline the steps (logic) for its solution
3. Write the computer instructions (code)
4. Check-out or correct (debug) the program

VII. A. 1 The four steps in problem-solving may be summed up as _____, outline, _____, and checkout.

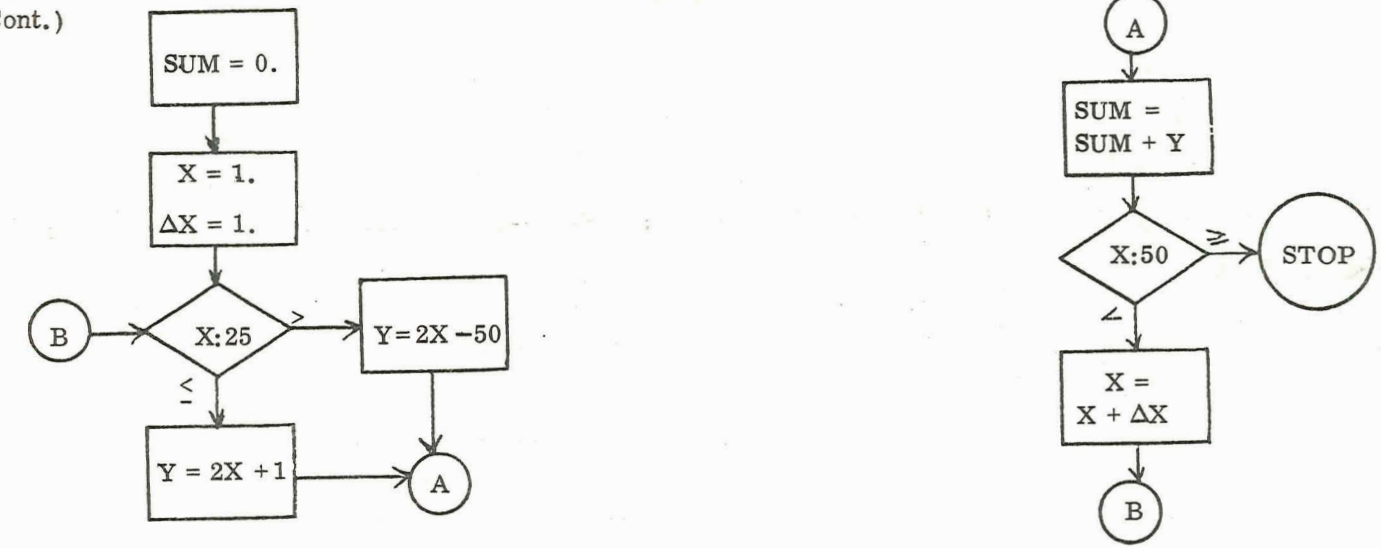
Answer: identify
code

Once the problem is identified, the second step is to outline the steps required to produce a solution. This outline is often in the form of a graphical representation called a flow chart. A flow chart pictures the sequence in which arithmetic and logical operations should occur, and shows the relationship of one part of a program to other parts. It may look something like this:

PART VII

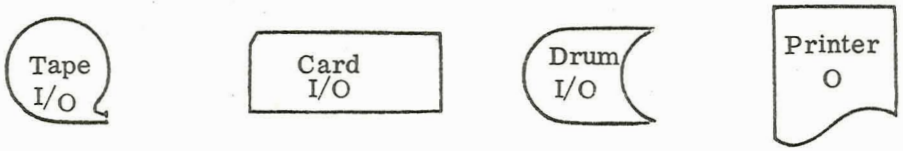
PROGRAMMING TECHNIQUES

VII. A. 1
(Cont.)



Although there is no real standardization in the drawing of flow charts, the following shapes may be accepted as generally recognizable: A rectangle represents a function, a diamond represents a decision, a small circle indicates a connector, an elongated hexagon indicates a predefined process or subroutine, a large circle indicates termination.

Various input/output media have these distinctive representations:

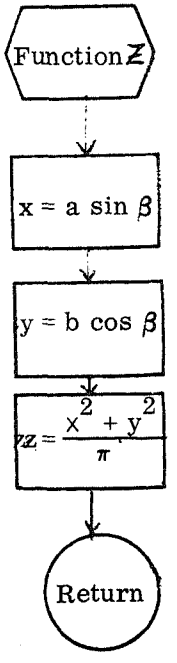


VII. A. 1 (Cont.) Use of these various shapes greatly facilitates the reading of a flow chart, and comprehension of the overall plan of execution, from input, through calculations, to output.

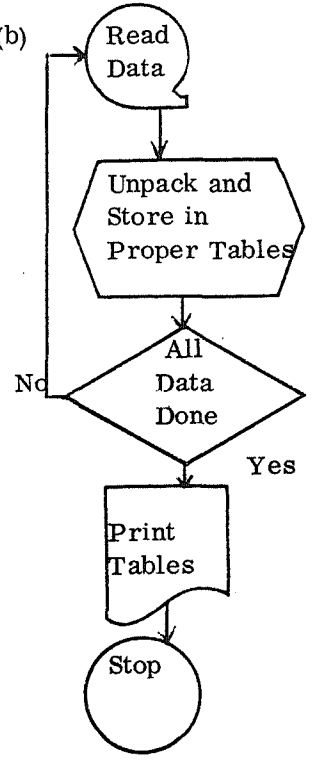
Flow charts may be drawn in varying degrees of detail. For overall planning, a flow chart should be quite general, indicating only major functions of various sections of the program. Each major section may have its own flow chart, containing more detail, which may serve as a guide to coding.

VII. A. 2. Would you consider the following flow charts to be general or detailed:

(a)



(b)



Answer:

(a) Detailed

(b) General

VII. A. 2 (Cont.)

A good flow chart will provide the following services:

(1) It will serve as a means of experimenting with various approaches to solving the problem.

(2) It will provide a sound basis for coding.

(3) It will be a useful piece of documentation, enabling others to understand readily the purpose and plan of the program.

General guidelines to drawing flowcharts include:

(1) In drawing a flow chart, it is desirable to work from left to right and top to bottom of the page.

(2) Direction of flow of execution is indicated by lines with arrowheads connecting the various boxes.

(3) Connectors are used to indicate connections between remote parts of the flow chart, so as to avoid a clutter of crossing lines.

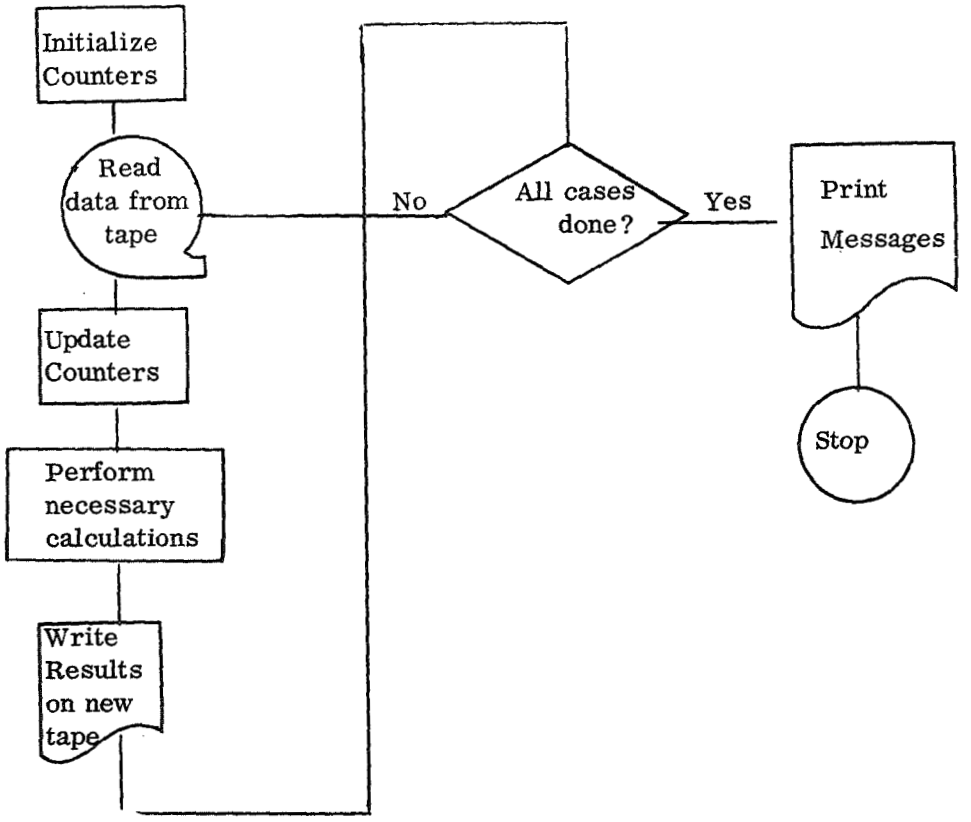
(4) Multiple entries to a box should combine into one line before actual entry.

(5) Whenever possible, a flow chart should relate to a source-language listing by using statement numbers or page numbers for cross referencing.

(6) Programmer's name and date should be included.

(7) Every major section of the program and every major decision should be represented on the flow chart.

VII. A. 3. What three undesirable features do you find in this flow chart ?

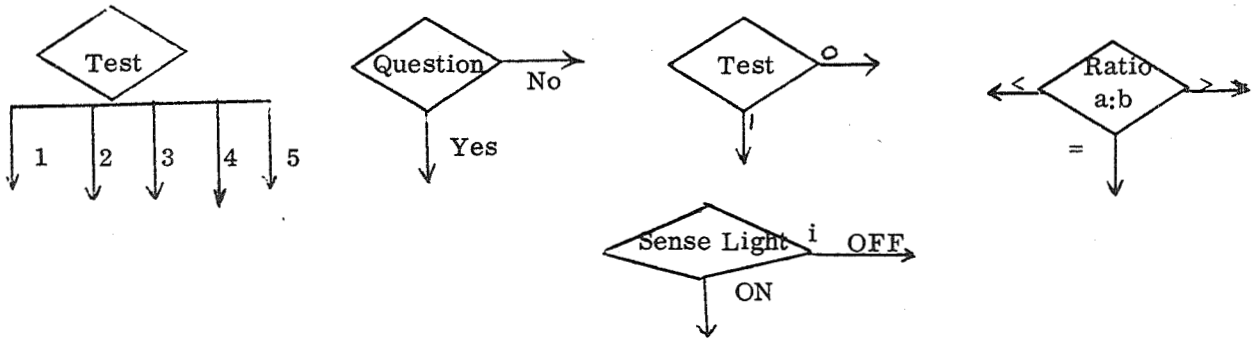


Answer:

1. Printer Symbol for tape output
2. Arrowheads omitted
3. Crossing lines

VII. A. 3 (Cont.)

Any box of a flow chart may have several paths leading to it. A function box will have only one path leading from it, but a decision box may have several exits. Each exit should be labelled so as to indicate under what conditions it is used. For example, these are commonly used decision techniques:



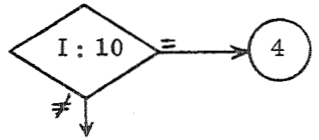
VII. A. 4. Draw a flow chart for the FORTRAN statements:

(a) IF (I . EQ . 10) GO TO 4

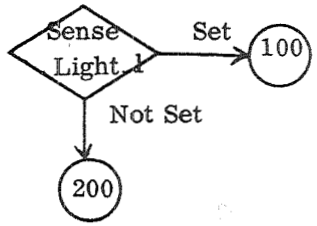
(b) IF (SENSE LIGHT 1) 100, 200

Answers:

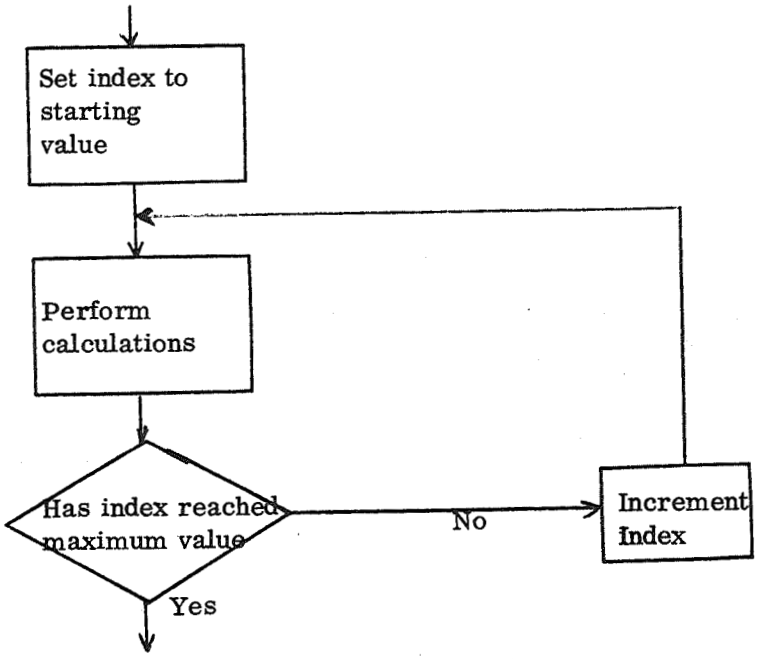
(a)



(b)



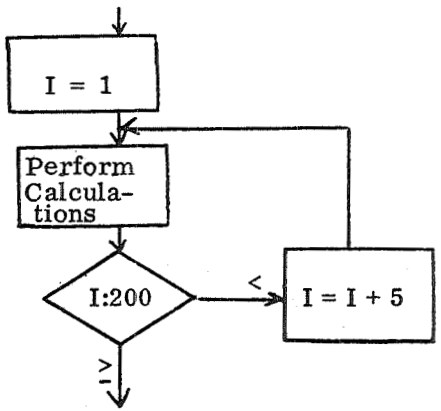
VII. A. 4 (Cont.) A DO loop may be represented in a flow chart in this manner:



VII. A. 5. Draw a flow chart for the FORTRAN statement:

```
      DO      10      I = 1,  200,5
      .
      .
      .
10    CONTINUE
```

Answer:



VII. A. 5 (Cont.)

Four objectives of computer programming are accuracy, speed, simplicity and economy of storage. In order to achieve these objectives, several aids are available to today's programmers. In the course of trying to obtain the greatest possible accuracy, an entire field of study, called numerical analysis, has developed. This branch of mathemetics has studied techniques for handling common mathematical problems on digital computers, considering such problems as error due to round-off or truncation, large (or small) numbers, etc. From these studies have come generalized techniques, or algorithms, for coping with such problems as evaluating various types of functions, handling matrices, and solving various systems of equations. A good many of these techniques are available from computer manufacturers or large computer users in the form of generalized subroutines. Also, many books are available, giving detailed descriptions of various techniques, on which programs may be based.

VII. B. 1

Numerical analysis is concerned with finding methods of solving mathematical problems on digital computers with great accuracy. True or false? _____

Answer: True

In other computer problems accuracy is not at stake, but a selection from various alternative logical techniques will determine the speed with which the problem is solved. A classical example is the problem of sorting a group of items into some specified order. Again, various techniques have been worked out and are available to the programmer. (See Chapin, Ned, An Introduction to Automatic Computers, Chapter 14, for a discussion of sorting techniques.)

VII. C. 1 Accuracy is always the prime concern of the programmer. True or false? _____ Answer: False

While some programmers may take a certain pride in producing a very sophisticated program, based upon complex and tricky logic, it must be admitted that simplicity is a most desirable feature. Often a programmer must assume responsibility for a program written by another. Excessive complexity and clever tricks here are an obstacle to understanding the program. Frequently a programmer is required to modify or up-date a program he had written some time previously. His own earlier trickiness may now be a hindrance to himself as he tries to refresh his memory. Simplicity pays!

After a program has been written, it must be checked out or "debugged" to find and correct errors in the program.

VII. D. 1 "Debug" means to correct or "get the bugs out" of a program. True or false? _____ Answer: True

Several types of errors may exist at first--key-punch errors, coding errors, logical errors. Key-punch and coding errors may often be found by a careful perusal of the coding sheets and of the machine listing of the program. Common mistakes to look for are I instead of 1, O instead of 0, Z instead of 2, S instead of 5, misspelled variable names, punctuation errors.

VII. D. 2 Find the errors in the following FORTRAN statements:

a.) D0 10 I = 1, 5, M

b.) FORMAT (1 X 4HTEST/ (1X10E10. 2)

c.) DIMINSION ALPHA (50), BETA(50, 10)

d.) IDX = 3* (M+N) **AVAL/4

Answers:

a.) Zero instead of O in DO

b.) Missing terminal right parenthesis

c.) DIMENSION misspelled

d.) Improper mode for exponent

Some errors are detected by the compiler itself, and result in messages to the programmer. "AC" - the number of arguments in a current reference to a subroutine differs from the number which occurred in a prior reference; or "CT" - CONTINUE statement is missing a statement number; etc. (See Control Data 6000 Series FORTRAN reference manual for list of error codes.)

Logical errors are harder to find. There are several general approaches to this problem, some of which are assisted by the computer itself.

VII. D. 2 (Cont.) For any arithmetic calculations, it is necessary to have a hand-check -- a hand-calculated answer for a check case. The check case should be such as to point up errors, if any, in input values as well as erroneous sequence of operations. Critical values, as well as normal values, should be tested. If the problem originated with someone other than the programmer, as is most often the case, it is desirable for the originator of the problem to provide realistic check-case values for the hand-check, if not the entire check-case solution.

VII. D. 3 Hand calculations are often tedious and should be performed by the originator of the problem, rather than by the programmer. True or false? _____ Answer: False

Another desirable debug technique is to catch mistakes before they go on the machine. A careful study of the coding will often turn up mistakes of various types; and often, too, another programmer will see problems to which the original programmer may be "blind". Care and time should be taken to find as many bugs as possible before going onto the computer.

Items to look for while studying the coding are:

1.) Loop parameters -- will the computer execute the loop the desired number of times?

2.) Calling sequences -- are they of proper length and variables of proper mode?

3.) Common storage -- is the proper value available at the desired time?

VII. D. 3
(Cont.)

4.) Flow chart -- does the program follow the flow of logic prescribed in the flow chart?

5.) Tapes -- are they rewound when they should be?

6.) Constants -- are they correct values?

7.) Fractions -- can the denominator ever become zero?

8.) Inverse trig functions -- are the results in the proper quadrant?

VII. D. 4

A programmer who is unfamiliar with a program will be less likely to find coding or grammatical errors in the program than will the author of the program. True or false? _____

Answer: False

Tests of accuracy may be written into the program. For example, when solving a system of equations, the solution may be tried out in the equations during execution of the program.

Every branch of a program should be tested to assure that flow of execution is correct for every possible input. This testing is very difficult for large, complex problems, where contingencies are many or multiple, but it is important that it be done.

VII. D. 5 A programmer who is unfamiliar with a program will be less likely to find logical errors in the program than will the author of the program. True or false? _____

Answer: True

The most simple debugging technique is self-programmed. Be generous with the print-out of intermediate information. Some of this information could be identification of a particular program branch reached, intermediate data to help verify a hand computed check case or a count of a particular number of iterations required. As the program is checked, the print statements no longer needed may be removed. On the other hand, if the programmer sees a need for more intermediate information, other print statements may be added.

To assist in debugging, most computers and installations have available various debugging aids. These include memory dump -- a printout of all or part of memory at any one given time; and tracing programs -- a printout of certain computer registers at each step of execution or at certain requested points during execution. Memory dumps are time-consuming, particularly if the computer is to continue execution of the program following the dump. Tracing routines are even slower, but are of value as a last-resort debug technique.

Before a programmer is through with a program he should provide adequate documentation of the program. This documentation is valuable for his own records, for his employer, and for other programmers who may have occasion to use his program. Documentation should include a clear statement of the problem, a summary of any mathematical analysis involved in solving the problem, a flow chart indicating every basic decision point in the program, and a listing of the checked-out program, complete with comments and cross-referencing to the flow chart.

VII.E.1 Three essential elements of documentation are: _____, _____, _____, _____.

Answer: Statement of problem
Flow Chart
Listing

A few final helpful hints:

1.) It is usually unwise to assume an initial condition -- always set it in the program.

2.) Use variables instead of constants whenever possible -- this often prevents trouble in communicating between subroutines and main program, and makes modifications of constant values easier as requirements or conditions change.

3.) Simplicity is valuable -- in general, the straightforward method is just as economical and more desirable than the devious method.

4.) Make use of existing programs -- large subroutine libraries are available to most programmers for most computers. Use them.

5.) Be liberal with comments in coding -- these are an aid to the programmer and to others who may use the program.

6.) Put sequence numbers on source cards -- A good insurance policy to take out on your source deck is to have your cards sequenced. An accident that causes your source deck to become shuffled can seem like a minor catastrophe.