IBM®

FORTRAN for the IBM 1130

Chapter 1

Programmed Instruction Course

IBM

FORTRAN for the IBM 1130

Chapter 1

Programmed Instruction Course

FOREWORD

The function of this course is to acquaint the student with computer programming using the FORTRAN language. No previous experience with any kind of programming systems is assumed. For those who have such experience, an attempt is made to avoid conflicting ideas and to clarify typical misconceptions which might arise.

The version of the FORTRAN system referred to in this course is FORTRAN designed for the IBM 1130 Computing System. Additional information on FORTRAN for the 1130 may be found in the reference manual C26-5933. The FORTRAN systems designed for other computers utilize the same concepts and require only slight modifications of a few details to be completely understood. These differences are outlined in various reference manuals pertaining to the particular computer systems or may be found in the "General Information Manual - FORTRAN" (IBM No. F28-8074).

The text of the course is of the self-instructional variety permitting the student to progress at his own rate. There are four separate chapters in the text. The course is designed for independent self-study under the administrative control of an advisor. Four examinations and a choice of final test problems as well as periodic applied problems, are provided.

In this chapter, you will learn that the FORTRAN language makes it possible to express a mathematical problem as a sequence of statements. You will also learn how to express a formula in FORTRAN by means of an Arithmetic Statement.

Copies of this publication can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM DPD Education Development, Education Center, Endicott, New York.

INTRODUCTION

If you are already familiar with computers and data processing, you may skip the following material in this introduction and turn to the preface on page ix.

Basic Data Processing Ideas

Learning how to use a computer can be compared to learning how to drive an automobile. Thanks to many successful innovations in automotive development, the new driver is confronted with the rather simple process of learning how to steer and to recognize which buttons and pedals control, say, the lights or the driving speed. He has no need to understand the technical design and construction of the automobile. Similarly, the computer user need not be concerned with the engineering development and operation of a computer, but can concentrate on the problem of how the computer can assist him in solving a particular problem. Many people who have no detailed knowledge of how an automobile runs have become excellent drivers. In much the same way, a number of people have learned how to use the computer to obtain the kinds of calculated results that are best handled in the rapid fashion of a computer.

Electronic computers are called digital computers because they work with numbers and have the ability to count. The counting function has been greatly refined since the invention of the abacus, the adding machine and the desk calculator. Today's high-speed, electronic, digital computers can handle alphabetic as well as numerical data, and, instead of being restricted to simple mathematical operations, can perform complicated calculations, manipulate alphabetic information, and make logical decisions, all at tremendous speed.

Electronic computers are used in business for a variety of reasons. When properly applied, they can save time or money (or both) in producing reports for management and government, in preparing checks and earnings statements for employees, in writing statements to customers, in keeping records of accounts payable to suppliers. In many situations they make it possible to obtain information that would otherwise not be economically justifiable. In some cases they provide the basis for an improved management control of a business that would not be feasible for time or money reasons without a computer. They are also widely used for engineering and scientific computation.

In carrying out these functions, a number of basic computer operations are performed. Information appearing on punched cards is <u>listed</u> (printed). Various <u>calculations</u> are performed on data. Detailed information is <u>summarized</u> (totaled), often according to several classifications. Information is <u>edited</u>; this means two rather different things. On one hand, <u>source data</u> (input information) is checked for validity and accuracy before it is used in further processing. On the other, editing refers to the rearrangement of results for easy reading, such as by inserting dollar signs, decimal points and commas, deleting zeros in front of numbers, and providing adequate space between numbers.

These operations are performed on data. It is necessary also to consider how the data is organized, since the arrangement of the information has a most significant effect on how the processing is done. This brings up one of the most fundamental concepts in data processing, that of a file.

A file is a collection of records containing information about a group of related accounts, people, stock items, etc. For instance, an accounts receivable file contains a record for each customer, showing at least the customer's name, address, account number, and amount owed. It may also contain his credit limit, the length of time the amount owed has been due (the "age" of the account), and other information, depending on the needs of the particular business. In a payroll file, the record for each employee contains such information as name, payroll number, department, sex, social security number, number of dependents, pay rate, year-to-date gross earnings, year-to-date taxes withheld, year-to-date social security tax, and often many other things.

These examples involve master files, which contain semipermanent information, some of which is updated (modified) periodically. A transaction file, on the other hand, contains information used to update a master file. Examples: a file containing a record for each customer purchase, or a file of labor vouchers used to calculate gross pay. In addition to master and transaction files there are report files, which contain information extracted from a master file. Example: the quarterly social security reports required by the federal government.

It is obviously necessary to have some way of identifying each record in a file. This is usually accomplished by establishing one item in the record as the key or control field of the record. The key distinguishes each record from all others, and is used in almost all file operations. Examples of keys: the customer's account number in an accounts receivable application, the employee's pay number in a payroll, the part number in an inventory control application, the salesman's number in a sales commission job.

Almost all data processing involves operations on files. It is frequently necessary to sort the records in a file, that is, to put the records in ascending sequence (sometimes descending) according to the keys of the records. For example, it may be necessary to sort employee labor vouchers into payroll number sequence before this transaction file can be processed against the payroll master file. Data processing methods fall into two broad and rather different classes, according to whether the files involved are or are not required to be sorted before the primary processing can be done.

Another common file operation is the combining of two or more files to form one file. If the combined file contains all records from the separate files, this operation is called merging; if some of the original records are omitted from the combined file it is properly called collating. (The distinction between the two terms is not always observed in practice.)

Careful planning is required to combine the basic operations so that the files are properly processed and the desired result produced. It is necessary to establish goals of the application, the time schedules that must be met, the exact nature of the operations to be carried out, etc. All of this takes more time than might first be thought for two reasons that are fundamental to a proper understanding of electronic data processing.

1. All processing must be defined in advance, with very few exceptions. For instance, it often happens that a customer sends in a check for an amount different from the amount shown on his bill. The person planning the accounts receivable job cannot proceed on the assumption that all payments will be for exact billed amounts and say, "I'll worry about that problem when it happens." The processing operations for such a situation must be planned in advance. Again, it is necessary to decide what to do about processing errors before an application is placed on the machine.

2. A machine cannot exercise judgment unless it has been given explicit directions for making a decision. A machine can be set up to make relatively complex decisions, if they are expressible in quantitative terms, but it must be instructed how to make the decisions and what to do in each alternative. One can say to a computer, in suitable language, "If a man's deductions exceed his gross pay, omit as many deductions as necessary; the order in which to omit them is specified in the following table, where the first deduction is the least crucial." One cannot say, "If anything unusual comes up, do what you think is best."

When the task has been properly defined in terms of what is to be done, the next step is to decide how to do it with the computer. This step involves expressing the processing in terms of operations that can be carried out with the available computing equipment. One of the primary tools of this step is the flowchart, which shows the sequence of operations in graphic form.

The next step is programming. This includes two activities, one of which is detailed flowcharting, showing in greater depth exactly what is to be done at each stage of the computer processing. The other activity in programming is coding.

The fundamental problem is this: The "language" in which the computer can accept "instructions" is very different from the language in which we ordinarily describe data processing. One way or another, the procedure to be followed must be translated into the computer's language. For instance, one says, "Summarize sales by salesman and district." The computer understands only such instructions as, "Add these two numbers", or, "Go to the print steps if these two numbers are not the same", or, "Read a card and place the information in the card input area."

Coding is the process of stating a procedure in a language acceptable to the computer. (The term is derived from the fact that the computer's basic language consists of instructions that are written in a "coded" system of numbers and letters.) In few cases is it necessary to do the entire job of translation all the way to the final form of the instructions as they will be obeyed ("executed") by the computer. Usually, instructions are written in a symbolic form that is rather similar to the machine's language, but considerably more convenient for the writer. The final step of the translation is then performed with the machine's assistance. In other situations the machine procedure can be written in a language quite similar to ordinary English, the bulk of the translation being done with the aid of a special computer program (set of instructions).

Programming and coding involve so much detailed work that most programs do not operate correctly when first tried. Thus it is necessary to debug the program (locate and correct the errors) and to test it with hypothetical cases to be sure that it properly processes the data. All of this goes under the name of program checkout.

One more activity remains before the program is ready to be used; the master file must be prepared. This usually requires converting the file from the form in which it was used with previous manual methods. File conversion can be a sizable task in itself, and one that often must be started well before the program is completed.

Organization of a Digital Computer

A digital computer combines the following components in one computing system (Figure 1 shows the manner of combination):

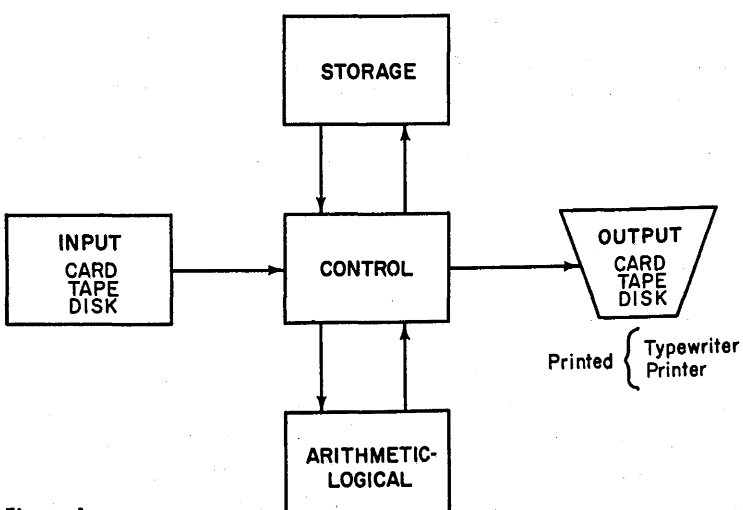

Figure 1

Input

Digital computers accept numbers, letters, and symbols which are fed into the system from punched cards, punched paper tape, or magnetic tape or disk. Information can also be manually inserted by means of a keyboard or switches.

Control

The computer operates under the direction of a control unit. The sequence of steps to be performed by the computer must be translated into detailed instructions which the computer can understand. When a series of instructions called a program is placed in computer storage, the program is then available to the control unit, as needed, to direct and complete an entire sequence of operations. Special instructions enable the arithmetic-logical unit to make decisions based on intermediate results. These decisions cause the computer to select the proper action from several alternatives for solving a problem.

Storage or Memory

Data is stored internally by electromechanical, magnetic, or electronic devices in a manner similar to the way in which music or lectures are stored on a tape for playback on a tape recorder, although the notation used is quite different. Stored information is accessible, can be referred to once or many times, and can be replaced whenever desired. The computer can store original data, intermediate results, reference tables or instructions. Each storage location is identified by an individual location number called an address. Numerical addresses permit the computer to locate data and instructions as needed during the course of a problem. The speed of computer operations is often determined by the access time, i.e., the length of time required to obtain a number from storage and to make the number available to other units of the computer system.

Arithmetic-Logical

The arithmetic-logical unit can add, subtract, multiply, divide, compare numbers, and make logical decisions such as whether a number is positive, negative, or zero. All complex calculations are combinations of these basic operations.

Output

After performing calculations, the computer can produce answers in several forms. Results can be punched into cards, recorded on magnetic tapes or magnetic disks, or printed in report form. Printers provide high-speed computer output by printing an entire line of information at one time.

Basic Computer Operation

The functioning of the computer elements can be compared to the steps required to solve a problem by paper-and-pencil methods. The input corresponds to information given for the problem. A knowledge of arithmetic controls relates to the handling of the problem. The arithmetic-logical unit performs the same function as do manual calculations. Storage can be compared to the work papers on which intermediate results are noted. The answers to the problem are the output.

An examination of the computer's internal processes during actual operation provides insight into the way in which it can be used. The key component of these processes in the control unit; this can locate and collect the information stored at various points in the computer memory. When information is retrieved from a specific location in the memory, the information is still preserved at that location for future use, a process called nondestructive readout. Storage locations may contain either instructions to the computer, numbers, or data required in a specified operation. An instruction is brought from storage to the control unit, is interpreted, and is then executed. The execution of an instruction calls into action one or more other computer components. Thus, if the instruction is an order to perform some arithmetic calculation, the arithmetic unit and the memory are involved. An input instruction activates the card reader or a tape unit, and information is sent from the input unit to certain locations in memory for storage. Briefly, the computer operates on a sequence of instructions which are located in the computer memory and which are in turn brought to the control unit where they are interpreted and executed. The instructions are numerical or alphabetic codes which, while in memory, are indistinguishable from other information stored there, such as program data. To distinguish between the kinds of information stored in the computer memory, a specific program is followed. The program interprets and executes the instructions which pick up the data or the numbers on which it then performs the required calculations.

Communicating with the Computer

Since the computer can operate only on instructions in coded form, the program steps to be followed for a given program must be made available to the computer in this coded form (called machine language). Thus, the person (called a programmer) who writes a computer program might have to do so in the language of the machine.

However, very few people would find the writing of a program in machine language a simple task, since everything must be set forth in machine code, which has little, if any, relationship to the English language. To relieve the programmer of machine coding, special programs have been written which can convert easy-to-use English type language into machine language. The processor receives the programming language and converts or translates the instructions into machine-language instructions which are acceptable to the computer. This translation phase is often called "compilation". Thus we can also say "the computer compiles programming language to produce machine language." One of these programming languages is FORTRAN, which stands for FORmula TRANslation. The FORTRAN language is procedure-oriented rather than machine-oriented and is composed of a group of statements which are considerably fewer in number than the machine language instructions of the computer. When writing FORTRAN statements, the programmer must comply with the rules pertaining to that language. Often, each step of a procedure corresponds to a single FORTRAN statement, which in turn may require two or more machine-language instructions. Thus, coding in FORTRAN simplifies the programmer's task of coding a problem for the computer, because FORTRAN has fewer statements and rules to be learned.

The translation operation is initiated by placing the FORTRAN processor in the computer memory where it processes the FORTRAN statement cards and produces the machine-language program on cards. Following the translation process, the machine-language program is put into the computer to execute the program, using any required data to produce the results. The general FORTRAN translation procedure is shown in Figure 2.

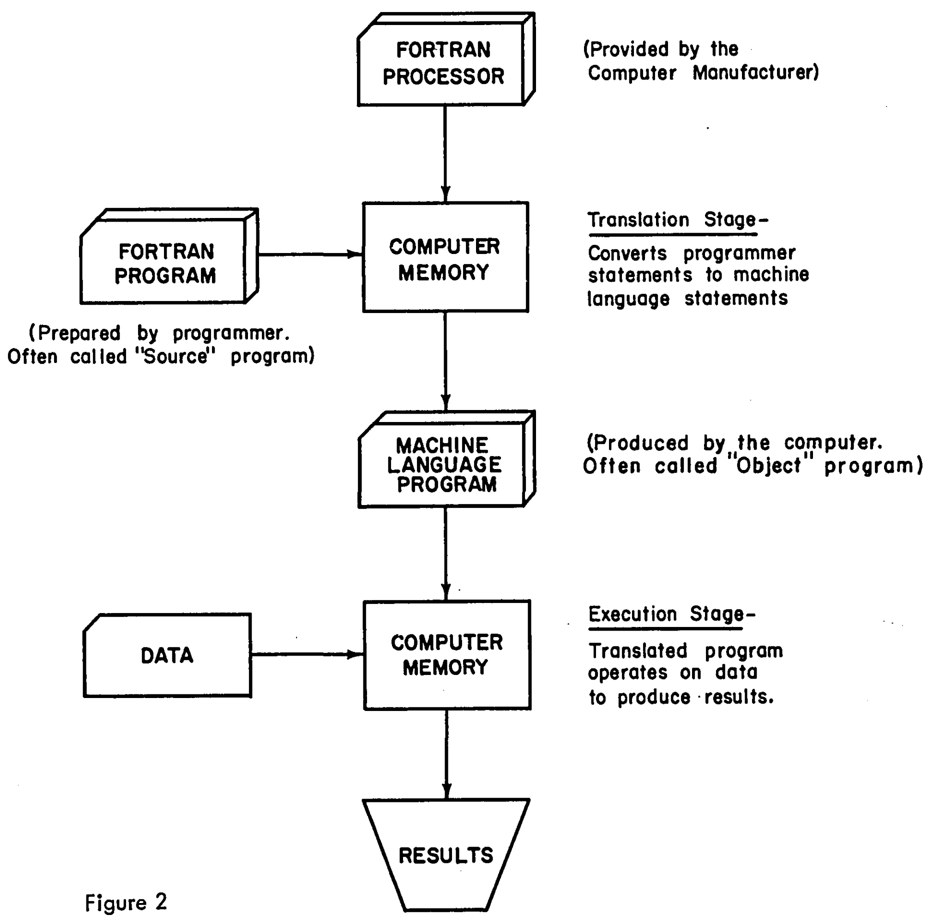

Figure 2

PREFACE

You are taking this course presumably to learn how a computer is programmed with "FORTRAN". The objective of the course is to provide the knowledge of some of the skills required to write computer programs using the FORTRAN system; it will not transform you into a "programmer". After completing the course you will probably need some assistance to run an actual program on a particular computer, and considerable experience is necessary in any programming system before a high degree of proficiency can be attained. However, this course will provide a sound basis for attaining these goals.

The self-teaching method applied in this text requires that you carefully read each "frame" in turn, and, if followed by a question, answer it. Do not write in the text, however. You should make your responses mentally where possible, or use scratch paper to compute your answers. Occasionally you will be told to skip review frames if a test question is answered successfully, and periodically you will be told to perform a sample problem exercise in the problem book. (These sample problems are not to be graded, but are merely used to demonstrate concepts. Attempt to solve each one before you look up the solution. There is a big disadvantage to the student who does not do this. These exercises are provided so that you can evaluate your own progress and seek help if your progress is not satisfactory as denoted by your ability to complete the exercises.) At the end of each chapter you will find an examination and at the end of the course are two suggested final test problems which may be run on a computer or checked out by the advisor.

You will note that each frame is usually followed by a question, which in turn is followed by an answer. To avoid seeing the answers before you respond to the question you will need to mask the answers as you proceed down the page. A card placed horizontally on the page may be used for this purpose. As you slide the card down the page you will uncover three black dots called "bullets" that immediately precede the answer. When you see the bullets, you should read the question and respond to it before moving the card any further.

For your convenience, a chapter Reference Index is provided at the back of each chapter. In addition, a course Reference Index is provided at the back of the Problem Book. These indices are provided to aid you in finding specific information within the course text.

Keep the Problem Book with you whenever you are working with the programmed text. You will be referred to it from the text for exercises and from it to the text when you complete an exercise. The Problem Book will be yours to keep at the end of the course.

1 Q. (question) Have you read the PREFACE? (yes or no) _______

●●●

If your answer is "yes", continue to the next frame.
If your answer is "no", better read it before continuing.
You'll find it on page ix following the Introduction. After reading it, go to frame 2.

2 A program is written for the purpose of "directing the computer" through the steps of the problem. If the computer is going to perform arithmetic,for example, it must be told each step of the operations.

Q. A program is a detailed description of each _______ of the solution of a problem.

●●●

A. step

3 You will soon learn, for example, that the FORTRAN statement

Y = A + B + C

represents a "program" to find the sum of the three quantities A, B, and C and to set the value of Y equal to that sum.

Q. The statement Y = A + B + C is actually a small computer _______.

●●●

A. program

4 A program language sucn as FORTRAN is like any language: it has both vocabulary and rules of grammar and punctuation. You will learn the rules and vocabulary of FORTRAN in this course.

5 The FORTRAN language is particularly well adapted to mathematical problems. The way in which mathematical problems are programmed with FORTRAN closely resembles ordinary algebraic notation.

Q. FORTRAN is particularly adapted to the programming of _______ problems.

●●●

A. mathematical

6 The basic unit of the FORTRAN language is called a "statement". There are approximately 22 different types of statements, for example, in the 1130 version of FORTRAN.

Q. The 1130 FORTRAN language is made up of 22 different types of _______.

●●●

A. statements

7 One particular type of statement is used to tell the computer to perform mathematical computation. This statement type is called an "Arithmetic statement".

Q. The statement that is used specifically for mathematical computation is called the " _______ statement".

●●●

A. Arithmetic

8 The previous illustration of a FORTRAN statement; Y = A + B + C, is an example of an _______.

●●●

A. Arithmetic statement

9 The FORTRAN language has a method by which the computer can be told to perform basic arithmetic operations. These operations are addition, subtraction, multiplication, and division.

Q. The example used earlier, Y = A + B + C, shows how FORTRAN tells the computer to perform the operation of _______.

●●●

A. addition

10 In familiar mathematical notation the basic arithmetic operations of addition and subtraction are denoted by the "signs" of +, -.

Q. The - sign in ordinary notation indicates the operation of _______.

●●●

A. subtraction

11 In the FORTRAN language the operations of addition and subtraction are indicated in exactly the same way: the symbols + and - are used. The expression in FORTRAN of A + B - C indicates that the computer is to "add the quantities A and B and subtract the quantity C from that result".

Q. The FORTRAN expression X + Y - Z + W involves the operations of _______ and _______.

●●●

A. addition, subtraction

12 Multiplication is denoted by a slightly different, but easily remembered symbol: the asterisk (*). The FORTRAN expression W*X indicates that the quantity W is multiplied by the quantity X.

Q. The FORTRAN expression A*X + B*Y involves the operations of _______ and _______.

●●●

A. multiplication, addition

13 FORTRAN tells the computer to divide quantities that are separated by the slash (/) sign. For example, the expression A/B tells the computer to "divide the quantity A by the quantity B".

Q. The FORTRAN expression A*X/Z involves the operations of _______ and _______.

●●●

A. multiplication, division

14 The basic mathematical operations are indicated in FORTRAN language by the signs (also called operators):

addition	+	multiplication	*
subtraction	-	division	/

15 Q. Let's see how well you understand the material presented so far. The basic operations are indicated in FORTRAN expressions by the following signs:

multiplication _______
addition _______
division _______
subtraction _______

●●●

A. multiplication: *
addition: +
division: /
subtraction: -

16 Q. The FORTRAN expression A + B + C - D indicates that the quantities A, B, and C are to be _______ and the quantity D is to be _______ from the result.

●●●

A. added, subtracted

17 If you did not answer that question correctly, continue to the next frame. If you answered correctly, go to frame 26.

18 First let's look again at the "signs" or operators by which the computer is told to perform arithmetic:

+ addition
- subtraction
* multiplication
/ division

These operators are used in FORTRAN to tell the computer to perform _______.

●●●

A. arithmetic

19 One program statement may indicate more than one arithmetic operation. In the example

Y = A+B+C

the three quantities A, B, and C are to be added. Similarly, the Arithmetic statement

Y = A+B+C-D

denotes that the computer is to perform both _______ and _______.

●●●

A. addition
subtraction

20 Q. Let's try another sample question: The FORTRAN expression A*B/C indicates that the quantity _______ is to be multiplied by the quantity _______ and the result divided by the quantity _______.

●●●

A. A, B, C

21 If you answered that question correctly, skip to frame 26. If you did not answer correctly, continue with the next frame.

22 You will have to read more carefully and take more pains with the questions if you intend to continue. The material becomes much more complex in the later frames. This question is essentially the same as the previous test question: Given an expression involving mathematical operations and the quantities involved, you are asked, "What does the expression say?"

23 A person writing a program using FORTRAN or any other system is telling the computer what he wants the computer to do. In the FORTRAN language, if you want the computer to add the quantities A and B you construct the expression A+B. Simple, isn't it?

Q. If you want the computer to subtract B from X, the expression in FORTRAN is _______.

●●●

A. X - B

24 The FORTRAN expression X*Y-Z tells the computer to subtract the quantity Z from the product of X and Y.

Q. The FORTRAN expression A*S+B/T tells the computer to multiply _______ and _______ and add the result to _______ divided by _______.

●●●

A. A, S, B, T

25 If you answered that question correctly, you may go on to the next frame.

If you didn't get the right answer, see your advisor.

26 Perform Exercise 1.1 in your problem book.

By now you have learned how to tell the computer to perform basic arithmetic operations. You will now learn one more basic operation that can be selected by an operation sign in FORTRAN: exponentiation. The term exponentiation refers to the operation of raising a quantity to a power (squaring, cubing, etc.).

Q. FORTRAN also permits the basic operation of _______ as well as addition, subtraction, division and multiplication.

●●●

A. exponentiation (if you are familiar with squaring, cubing, etc., you may skip the next frame.)

27 Exponentiation means using a quantity as a multiplication factor the number of times indicated by its exponent. In mathematics, the exponent is the value written to the upper right of a quantity. For example, X^4 means using X as a factor 4 times, or (X)(X)(X)(X).

Q. 2^{15} means the quantity 2 used as a factor _______ times.

●●●

A. 15

28 The FORTRAN language has an operation sign to tell the computer to perform exponentiations: ** (double asterisk). For example, to tell the computer to raise the quantity X to the power of 2 (X squared) the expression X**2 is used.

Q. The operator ** tells the computer to raise a quantity to the indicated _______.

●●●

A. power

29 The rule for using the exponentiation operator is simple: the quantity to the left of the operator is raised to the power indicated on the right.

Q. In the FORTRAN expression A**8 the quantity _______ is raised to the power of _______.

●●●

A. A, 8

30 A statement may involve exponentiation along with other operations. For example the expression A**2+B**2 would tell the computer to calculate "A squared plus B squared."

Q. The expression: A*Y**2-B*Y**3 involves the operations of _______, _______ and _______.

●●●

A. multiplication, exponentiation, subtraction

31 The familiar notation x^2 is interpreted as "x-squared" or "x to the second power." The equivalent expression in FORTRAN is the notation X**2.

Q. Given the formula $r = x^2 + y^3$ the equivalent expression in FORTRAN is the statement R = X _______ +Y _______.

●●●

A. **2, **3

32 By now you may have noticed that FORTRAN expressions shown in the examples have consisted of capital letters. This convention will be continued, and lower-case letters will be used only for examples of algebraic formula notation.

Q. Alphabetic symbols used in FORTRAN programs must be _______ letters.

•••

A. capital

33 The order in which the operations are carried out in a mathematical expression is most important. The algebraic expression ab+cd would tell the mathematician to multiply a and b, and multiply c and d, and then finally to add the two products.

Q. It is important to know the _______ in which mathematical operations are carried out.

•••

A. order

34 The same formula, ab+cd, would produce an entirely different value if the operations were carried out in a different order; for example, if c were added to the product of a and b before the step of multiplying by d were carried out, the result would be different.

Q. In the formula x+yz the first operation to be performed would be _______.

•••

A. multiplication (yz)

35 The computer solves a formula in much the same manner as a mathematician: one operation at a time. Therefore it is necessary for a language such as FORTRAN to have a clear-cut order or "hierarchy" of operations to convey the correct meaning to the computer.

36 To show an example of the "hierarchy" of operations, the FORTRAN statement Y = A*B+X*Y tells the computer to perform the two multiplication operations first, and then add the two products.

Q. In the FORTRAN statement A = B*C+D the first operation performed is _______.

●●●

A. multiplication (B*C)

37 In the simplest form of FORTRAN expression the order of operations is determined by reading the expression from left to right.

Q. The "hierarchy" of operations in a FORTRAN expression is dependent on scanning the expression in a _______ to _______ direction.

●●●

A. left, right

38 The computer evaluates FORTRAN expressions in a left-to-right manner if the sense of the expression is not violated by doing so. For instance, in the expression A+B*C, the computer, even though scanning left to right, cannot compute the value of A+B, because B must be multiplied by C before the value of A is added.

39 The sequence by which the computer performs computations is determined by the hierarchy of operations. The operation with the highest priority is exponentiation. For instance, in the expression B*C**2+A, the value of C**2 is computed first.

Q. The operation with the highest priority is _______.

●●●

A. Exponentiation

40 After the indicated exponentiations are performed, multiplications or divisions are performed next, reading again from left to right. If an expression contains both types of operation they are still executed in the order they appear in the left-to-right direction.

Q. In the FORTRAN expression F/G*H, applying the rule stated above, the operations of multiplication and division are performed in the following order: _______, then _______.

●●●

A. division, multiplication

41 Finally, all additions or subtractions are performed in the order they appear. If both types of operation are contained in an expression, they are again executed in the order they appear in the left-to-right direction.

Q. In the FORTRAN expression A+X-B the last operation, by name, to be performed is _______.

●●●

A. subtraction

42 The FORTRAN rule of hierarchy consists, then, of three parts:

1. All exponentiation (if any) is done first.
2. All multiplication and/or division (if any) is done second.
3. All addition and/or subtraction (if any) is done last.

Q. The operation with the highest priority performed in a FORTRAN expression would be _______ if any were indicated.

●●●

A. exponentiation

43 Each of the three groups of operations is called a "level of hierarchy." All operations on a given level must be completed before going to the next level.

Q. The operations of multiplication and division represent a _______ of hierarchy of operations.

●●●

A. level

44 The FORTRAN statement Y = A*X**2+B*X+C contains all three levels of hierarchy. The computer would interpret this statement to mean:

1. Compute X**2 (exponentiation level)
2. Compute A* (quantity computed in #1) and B*X (multiplication level)
3. Compute the sum of the three terms as directed (addition level) in that order.

Q. In the statement A = B/C+D the first operation to be performed would be _______.

●●●

A. division (B/C)

45 These rules of hierarchy are consistent with those of ordinary notation. For example, the ordinary expression x^2+bx+c would be computed by hand in the order: "square x, multiply b and x, and finally add the three quantities, x, bx, c". The levels of hierarchy are exactly the same as those in the FORTRAN rules. There is nothing new about these rules -- you have been following them for years.

46 In summary, then, the FORTRAN Arithmetic statement tells the computer to perform arithmetic operations, one at a time, to compute an entire expression. Just as an engineer would do in performing hand calculations, the operations are carried out in a specific order: exponentiation, multiplication and/or division, and addition and/or subtraction, in each case moving from left to right in the written expression until all operations at a given level of hierarchy are completed as directed.

47 Q. Given the FORTRAN statement R = A+B/C+D, indicate which of the two following formulae this statement represents.

a) $r = \frac{a + b}{c + d}$

b) $r = a + \frac{b}{c} + d$

●●●

A. b) $r = a + \frac{b}{c} + d$

48 If your answer was (b) and therefore correct, skip to frame 54. If your answer was (a) continue to the next frame.

49 Your answer, $\frac{a + b}{c + d}$, was incorrect. The FORTRAN statement R = A+B/C+D can be interpreted only one way according to the rules of hierarchy of operations: $a + \frac{b}{c} + d$.

50 To review, in a FORTRAN Arithmetic statement exponentiation is done first; multiplication and/or division is performed next; and addition and/or subtraction is done last.

Q. Given the statement Y = X**2+B*X+C the operations (by name) are carried out in the order _______, _______, and _______.

●●●

A. exponentiation (X**2), multiplication (B*X), addition (of the three terms)

51 Applying these rules of hierarchy to the statement R = A+B/C+D the interpretation is as follows: there are no exponentiations, there is one division (B/C) which is carried out first, and finally, there are three quantities (A, D, and B/C) to be added.

52 Q. Given the statement X = A/B+C, indicate which of the two following formulae this statement represents.

a) $x = \frac{a}{b} + c$

b) $x = \frac{a}{b+c}$

●●●

A. a) $\frac{a}{b} + c$

53 If your answer was (a) and therefore correct, continue to the next frame. If your answer was (b), see your advisor.

54 Perform Exercise 1.2 in your problem book.

So far you have learned to use FORTRAN to instruct the computer in only simple combinations of arithmetic operations. In the next few frames, you will be shown how to use parentheses to provide more flexibility in constructing mathematical expressions.

55 Going back to familiar formula notation, for example, the expression (a+b)(c+d) would mean "add a and b; add c and d: and finally, multiply the two sums together". The parentheses have changed the normal order of operations.

Q. Increased flexibility may be gained in writing mathematical expressions when ______ are used.

●●●

A. parentheses

56 FORTRAN Arithmetic statements make use of parentheses in exactly the same way they are used in familiar mathematical notation. The FORTRAN statement R = A*(B+C) tells the computer to "add the quantities B and C, then multiply this sum by A".

Q. Indicate which of the given FORTRAN statements truly represents the formula $y = \frac{a}{b+c}$

a) Y = A/(B+C)

b) Y = A/B+C

●●●

A. a) Y = A/(B+C)

57 The rules covering the use of parentheses in FORTRAN are exactly the same as those taught to the student of high-school algebra. The expression (A+B)/(C+D) tells the computer to "add the quantities A and B, add the quantities C and D, then divide the two results."

Q. The expression above without parentheses: A+B/C+D would tell the computer to perform the operation of _______ first.

•••

A. division (B/C)

58 Parentheses may be used freely in FORTRAN Arithmetic statements to construct expressions of any desired degree of complexity. The use of parentheses is pretty much governed by common sense; the instances when they are required are usually obvious.

Q. Write an equivalent FORTRAN expression for each of the following mathematical expressions:

a) a(b+c)

b) ab+c

•••

A. a) A*(B+C)
b) A*B+C

59 Parentheses are usually used only when the normal hierarchy of operations has to be altered to provide a meaningful expression.

The formula $y = \frac{x}{a+b}$ cannot be represented in a single FORTRAN expression without the use of parentheses. Y = X/A+B certainly is not correct.

Q. The correct FORTRAN representation of the formula $y = \frac{x}{a+b}$ is _______.

•••

A. Y = X/(A+B)

60 Each use of a left-parenthesis, (, must be balanced by the use of a right-parenthesis,), and vice versa. For example, the expression (F+(X*Y) is not valid because of the unbalanced parentheses.

Q. For every left-parenthesis in an expression, there must be a corresponding _______.

●●●

A. right-parenthesis

61 Parentheses may be used even where not necessarily required. Extra parentheses (as long as they are balanced in pairs) will not harm the computation.

Q. (True or False) The expression A*B+C*D and (A*B)+(C*D) would be considered equivalent.

●●●

A. True

62 Each balanced pair of parentheses sets off a group of operations which must be completely carried out before those operations outside the parentheses can be executed. For example, the expression A*(B+C+D) required that the three quantities be added before the multiplication operation can be done.

Q. In the example shown above, the operation of addition is carried out before the operation of multiplication; the parentheses have changed the _______ of operations.

●●●

A. hierarchy (or order)

63 Within each pair of parentheses, "exponentiation first, multiplication-division next, and addition-subtraction last" still applies. In other words, each parenthesized expression is computed as if it were a complete expression subject to the usual rules of hierarchy.

Q. In the expression X*(A**2+B) the first operation to be performed by the computer would be _______.

●●●

A. exponentiation (A**2)

64 When a group of operations within a set of parentheses is completed, that computed value usually becomes a quantity in the larger expression. For example, (A + B)*X tells the computer to "add A and B, then multiply that quantity by X".

Q. In the expression F/(G + H) the quantity F is divided by the value of the expression _______.

●●●

A. (G + H)

65 To summarize, FORTRAN expressions denoting arithmetic operations are constructed in pretty much the same fashion as in ordinary notation. A set of strict rules of hierarchy must be observed, but when the expression demands it, parentheses may be used to modify the over-all hierarchy of operations. The next frame contains a few examples of expressions in which parentheses are required.

66

Conventional Notation	FORTRAN Notation
$(a + b + c)\ (x + y + z)$	(A + B + C)*(X + Y + Z)
u^{n-1}	U**(N-1)
$\frac{1}{k + m + n}$	1/(K+M+N)
$((a+b)\ (x+y))^{n+1}$	((A + B)*(X + Y))**(N+1)

67 A pair of parentheses may be completely contained within another pair, as in the example ((A + B)*C + D)/X. This tells the computer to "add A and B, multiply this by C, add this entire result to D, and finally divide the result by X."

Q. The innermost pair of parentheses in the example above encloses the operation of _______.

●●●

A. addition (A+B)

68 When sets of parentheses are "nested" (contained within one another) the innermost set of operations must be completely carried out before going on to the remaining parts of the overall expression.

Q. In the expression ((A*X + B)*X + C)*X + D the innermost parentheses contain the expression _______.

●●●

A. A*X + B

69 The computer will interpret the expression ((A*X + B)*X + C)*X + D beginning with the innermost pair of parentheses: (A*X + B).

Once that value is computed, it becomes part of the expression in the outer parentheses, and so on.

Q. When parenthesized expressions are "nested" the computing begins in the _______ pair of parentheses.

●●●

A. innermost

70 The general rules for parentheses are:

1) Computing begins at the innermost pair when nested.
2) Parenthesized expressions must be completely executed before other operations are executed.
3) The usual rules of hierarchy of operations apply within parentheses.
4) The result of the operations within the parentheses becomes the quantity used in the over-all expression.

71 It should be noted that parenthesized expressions are connected to the over-all expression by an operation sign in all cases. For instance, the expression (A + B)*(C + D) requires that the * (asterisk) sign be present to denote multiplication of the two parenthesized sets. (A + B) (C + D) alone is not sufficient.

Q. Identify the correct representation below of the formula $y = \frac{a(b + c)}{d}$

a) Y = A(B + C)/D

b) Y = A*(B + C)/D

●●●

A. b) Y = A*(B + C)/D

72 Parentheses are not needed when the natural hierarchy of operations is appropriate, but it is easy to make a slight error and end up with the wrong answer. For example, the simple expression $\frac{x}{yz}$ might be accidentally misrepresented by a programmer to be X/Y*Z in a FORTRAN expression, while that expression actually means $\frac{x}{y}(z)$ to the computer due to the left-to-right nature of the hierarchy rule.

Q. By adding parentheses the FORTRAN expression X/Y*Z can be made to truly represent the expression $\frac{x}{yz}$ as follows:

_______.

●●●

A. X/(Y*Z)

73 It is a good idea to use parentheses wherever any doubt exists, such as the preceding example shows. Excess parentheses will not harm a FORTRAN statement, but the absence of parentheses when they are needed can be disastrous!

Q. The FORTRAN expression A/B/C means:

a) $\frac{a}{bc}$ or b) $\frac{a}{b/c}$

●●●

A. a) $\frac{a}{bc}$

74 You have seen that the FORTRAN expression A/B/C is interpreted as if it were parenthesized as ((A/B)/C) due to the left-to-right nature of the hierarchy rule. This demonstrates that, even without parentheses, there is only one way to interpret any FORTRAN expression within the rules.

Q. Show how the basic expression A/B/C must be parenthesized to represent the formula $\frac{a}{b/c}$.

●●●

A. A/(B/C)

75 Q. Write a FORTRAN expression to represent the formula

$$\frac{(x)(y)}{(w)(z)}$$

Use corresponding capital letters as symbols.

●●●

A. X*Y/(W*Z) (note: parentheses are necessary). Other possible answers: X/W*Y/Z or (X*Y)/(W*Z) or similar combinations.

76 If your answer agrees with any of the above, skip to frame 81. If you did not have the correct answer, continue to the next frame for the explanation.

77 Perhaps you forgot some necessary parentheses. Given the expression $\frac{xy}{wz}$ to construct an equivalent FORTRAN expression, the most obvious approach might be to construct the numerator X*Y, the denominator W*Z, and then place a division sign between them: X*Y/W*Z.

78 This answer X*Y/W*Z is incorrect because the computer interprets this as follows (left-to-right): multiply X and Y, divide that result by W, then multiply that entire result by Z, which is incorrect. This answer is equivalent to $\frac{xy}{w}(z)$, not $\frac{xy}{wz}$.

79 In order to make sure that a computer calculates W*Z before the division takes place, a pair of parentheses should be placed around the denominator, as in X*Y/(W*Z) which is a correct answer.

Q. Construct a FORTRAN expression equivalent to $\dfrac{a+\frac{b}{c}}{xy}$ using corresponding capital letter symbols: _______.

•••

A. (A+B/C)/(X*Y)

80 If your answer agrees with the one shown above, continue to the next frame. If your answer is incorrect, see your advisor.

81 Perform Exercise 1.3 in your problem book.

Up to this point we have not been concerned with the quantities involved in the arithmetic operations. We have simply used alphabetic symbols to represent these quantities, as in the example:

Y = A+B+C

In this Arithmetic statement, A, B, C, and Y represent the quantities or numbers used in the arithmetic operations.

Q. _______ symbols may be used to represent quantities in an Arithmetic statement.

•••

A. Alphabetic

82 You will now be shown that the quantities specified in an Arithmetic statement may be represented in two ways: constants and variables. These terms will be defined and illustrated in the next few frames.

Q. There are two ways to represent the _______ specified in an Arithmetic statement.

•••

A. quantities

83 One way to specify a quantity in a FORTRAN Arithmetic statement is to represent the quantity by the actual number itself. This type of specification is called a constant. For example:

Y = 2. + 1.

tells the computer to add the constants two and one.

Q. A number used to specify a quantity (in an Arithmetic statement) is called a ________.

●●●

A. constant

84 In general, a constant is a known quantity in a mathematical expression. For example, the formula for the approximate area of a circle is $A = 3.14159r^2$ in which the quantities 3.14159(pi) and the exponent 2 are known values.

Q. The constants in the formula $0.5gt^2$ are ________ and ________.

●●●

A. 0.5
2

85 In a FORTRAN expression, all constants are represented by the numbers themselves. An Arithmetic statement to find the area of a circle, for example, might be A = 3.14159*R**2

Q. The FORTRAN constants in the expression above are ________ and ________.

●●●

A. 3.14159
2

86 Any quantity in a FORTRAN expression which is known at the time the expression is written can be expressed as a constant. The desired value is simply written into the expression as the number itself.

Q. Write a FORTRAN expression to represent the expression $3.5x^2 - 2.9x - 0.5$

•••

A. 3.5*X**2-2.9*X-0.5

87 The other method of representing quantities in a FORTRAN expression is through the use of variables. The symbol X in the answer shown above is a variable.

Q. Quantities may be represented in a FORTRAN expression as either constants or _______.

•••

A. variables

88 A variable is defined as a symbol in a FORTRAN expression which represents a quantity whose value is specified elsewhere in the program, usually by some previous statement.

Q. In the expression A*X**2+B the symbols A, X, and B are called _______.

•••

A. variables

89 The numeric value associated with a variable is referred to by the use of its variable name in an expression. For example, in the expression A+B the computer is told to "add the quantity whose name is A to the quantity whose name is B".

Q. The value of a variable is obtained for use in an expression by the reference to its _______.

•••

A. name (or symbol)

90 The value of a variable may change repeatedly during the execution of a program. The value of a constant, as the term implies, may not be altered, however.

Q. In the expression 3.0*X the value of the variable named _______ is multiplied by the constant _______.

●●●

A. X
3.0

91 The variable name acts as a symbolic "handle" of a number whose value is to be used in an expression. Use of this symbolic name permits the programmer to write the expression without knowing the exact value to be used.

Q. In the FORTRAN expression X+Y-Z the variables _______, _______, and _______ represent numbers which are to be added and subtracted as the expression directs.

●●●

A. X, Y, Z

92 When a variable is used in a FORTRAN expression it is assumed that the programmer will have defined its value in some other part of the program.

Q. The FORTRAN expression A*B+X/Y uses previously defined _______ in place of the symbols when the expression is computed.

●●●

A. values (or numbers)

93 At the time a program is written a variable name is invented for each such "unknown" quantity and the name is written in the FORTRAN mathematical expression in place of the number it represents.

Q. In the FORTRAN expression R**20-S*T the first operation to be executed is raising the pre-defined value of the variable _______ to the power of _______.

●●●

A. R, 20

94 FORTRAN has rules for naming the variables used in an expression. In 1130 FORTRAN, for example, a variable may have from 1 to 5 alphabetic or numeric characters in its name. Other versions of the FORTRAN language have similar rules for the naming of variables.

Q. In 1130 FORTRAN a variable name may have from _______ to _______ alphabetic or numeric characters.

●●●

A. 1, 5

95 While a variable may have more than one character in its name and may also use numeric characters, the first character in any variable name must be alphabetic. Thus X1 is a legitimate variable, but 1X is not.

Q. The first character in any variable name must be _______.

●●●

A. alphabetic

96 Another rule regarding variable names is that a variable name may not have blanks between any of its letters or numbers. For example, BETA1 is a legitimate variable name, but BETA 1 is not because it contains an embedded blank.

Q. (True or False) JOB NO is an acceptable name.

●●●

A. False. (contains an embedded blank). JOBNO would be correct.

97 Being able to use more than one character in a variable name gives a great deal of added flexibility in the construction of expressions. The Ohm's Law formula could be written in FORTRAN as Y = A*B, but the symbols become more significant if the formula is written VOLTS = CURRN*RESIS, for example.

98 A list of typical variable names is shown below:

Z	DELTA
ANSWR	X12Y2
PI	NUMBR
XI	Y2

Q. The variable name CURRENT would not be a legitimate variable name in S/360 FORTRAN because it contains more than _______ characters.

●●●

A. 5

99 Here is a summary of variable naming rules:

1. The name may have from 1 to 5 alphabetic or numeric characters. (No special characters allowed, that is, / * &, etc. The dollar sign, however, is an allowable exception. It is arbitrarily considered to be an alphabetic character, and as such, its use in a name is legitimate.)
2. The first character of the name must be alphabetic.
3. No embedded blanks are permitted.

Q. Which of the following variable names are incorrect?

a. XRAY
b. SHARKS
c. 4XB
d. NO Y
e. FORTRAN
f. HELP
g. DOLR$
h. RHO
i. GO*F

●●●

A.
b. SHARKS - too many letters
c. 4XB - first character not alphabetic
d. NO Y - embedded blank
e. FORTRAN - too many letters
i. GO*F - contains a special character

100 1130 FORTRAN can tell the computer to perform two different types of arithmetic: real and integer arithmetic. These two types of arithmetic are called "modes". (In other versions of FORTRAN real is called "floating-point" and integer is called "fixed point". In this course we shall use the terms real and integer only.)

Q. The computer's two forms of arithmetic, integer and real, are called _______ of arithmetic.

●●●

A. modes

101 The real mode of arithmetic deals with real numbers. A real number is defined simply as a number which contains a decimal point. The integer mode of arithmetic deals with integers. An integer is a number which does not contain a decimal point.

Q. _______ numbers contain decimal points, _______ numbers do not.

●●●

A. real, integer

102 The real mode of arithmetic should be used for nearly all computations. This mode of arithmetic automatically takes care of correctly positioning the decimal point in the result.

Q. Nearly all computations should be performed in the _______ mode.

●●●

A. real

103 The integer mode of arithmetic makes use of integer, or whole number, quantities and, as such, is useful as a counter.

Q. The integer mode of arithmetic is used for counting and other operations involving _______ numbers.

●●●

A. integer (whole)

104 Since most computer work involves other than whole number quantities, the real mode is used almost exclusively for ordinary computation. In the real mode, almost all numbers, including fractional quantities, may be automatically handled by the computer.

Q. Numbers which contain fractional parts must be handled in the _______ mode.

●●●

A. real

105 The programmer indicates the mode of arithmetic that the computer is to use in any computation by the nature of the variables and constants used in the FORTRAN expressions. Thus, constants and variables have a definite mode associated with each.

Q. The mode of the arithmetic performed is determined by the mode of the _______ and _______ written in the FORTRAN expressions.

●●●

A. constants, variables

106 An expression containing real constants and variables will be computed using real arithmetic procedures; similarly, expressions containing integer constants and variables will be executed in the integer mode.

107 A constant in the real mode is identified by the presence of a decimal point. All constants that have decimal points are real constants.

Q. (True or False) All the constants in the list below are real constants:

3.141592
5280.0
11001
0.000001
100000.
32.174

●●●

A. False (11001 does not contain a decimal point)

108 Integer constants are numbers that do not contain a decimal point. Since all integer numbers are whole number quantities, no decimal point would be needed and this provides a convenient means of distinguishing between the modes of constants.

Q. (True or False) 10259 is an example of an integer constant.

●●●

A. True

109 A real number may be a whole number, but as a constant it must still contain a decimal point.

Q. Write the quantity "one million" as a real constant:

●●●

A. 1000000. (must include the decimal point)

110 Incidentally, commas are not permitted in constants of either mode; that is, the constant 1,000,000 is not permitted as a FORTRAN constant.

Q. The presence of a decimal point in a constant identifies it as a _______ number.

●●●

A. real

111 Zeroes after a decimal point are not necessary for a real whole number. For example, the constants 1.0 and 1. are identical as FORTRAN constants.

Q. (True or False) The constants 5,280.0 and 5,280. are legitimate real constants.

●●●

A. False (comma not permitted)

112 The quantities represented by variable names must also be identified with one of the two modes. The mode of a variable may be determined by the first letter in the variable name. This is an arbitrary convention of FORTRAN, and is known as an implicit specification of variable modes. We shall see later that it is possible to define variable modes in a manner which does not depend upon the first character in a name. For the time being, however, you will learn the rules of first letter (implicit) mode specification.

113 Let's review the rule for naming a variable: a variable name may have from one to five alphabetic or numeric characters, and the first character must be alphabetic.

Q. (True or False) All the variable names in the list below are legitimate:

X	A1234
X1Y2	DELTA

●●●

A. True

114 The mode of a variable may be determined by the first character in its name. If the variable name begins with any of the letters I, J, K, L, M, or N it is an integer variable; if it begins with any other letter it is a real variable.

Q. The mode of a variable may depend on the _______ character in the variable name.

●●●

A. first

115 An integer variable name begins with the letter I, J, K, L, M, or N. For example I, N, JOB, LETTR, NUMB, KAPPA, etc., are names of integer variables.

Q. The integer variable in the list below is ______.

FRANK
JOE
HENRY

•••

A. JOE

116 A real variable name begins with any alphabetic character except I, J, K, L, M, or N. Under this rule, the names X, Y, DELTA, YPRIM, A1DOT, etc., are all legitimate real variables.

Q. The name FIXED is a ______ variable name.

•••

A. real

117 A convenient aid for remembering the first letter rule in naming variables is to note that the letters I and N are the first two letters of the word INteger. All letters other than I through N would denote a real variable when used as the first letter of a variable name.

118 Remember it is only the first character in a variable that is important to the mode. The letters I, J, K, L, M, and N may be used in a real variable name as long as they are not the first character.

Q. (True or False) The variable name XIJKL is a real variable.

•••

A. True

119 Variable names are often invented to coincide with a meaningful word describing the quantity in the expression. You must be careful to avoid accidentally using the letters I, J, K, L, M, or N as the first character in an intended real variable name.

Q. (True or False) The FORTRAN statement FORCE = MASS*ACCEL contains only real variables.

●●●

A. False (MASS is an integer variable)

120 Sometimes a desirable name for a real variable, such as in the preceding example, begins with one of the forbidden letters. In this case an extra letter is often added to the front of the name to make it a legal real variable. For example, MASS might be re-named as AMASS or XMASS, etc.

Q. By adding a letter X in the appropriate place, the statement FORCE = MASS*ACCEL can be made to contain only real quantities as follows: _______.

●●●

A. FORCE = XMASS*ACCEL

121 To review for a moment, you now know that quantities or numbers can be represented in a FORTRAN expression as their exact value (constants) or as a symbolic name (variables), and that the arithmetic indicated in an expression is performed in one of two modes. The mode of the arithmetic is determined by the mode of the variables and constants in the expression. The mode of a constant is defined by the presence or absence of a decimal point; the mode of a variable depends on the first letter.

122 Perform Exercise 1.4 in your problem book.

In FORTRAN terminology an expression is defined as any combination of constants, variables, and operation signs. For example A*X/2.+B*X-C is called an "expression" or more exactly a "FORTRAN expression".

Q. F+G-Z is a combination of variables and operation signs and, as such, falls within the definition of a FORTRAN ________.

●●●

A. expression

123 By the definition in the preceding frame, a single constant or single variable may also be considered an "expression". Even a variable or constant preceded by a minus sign is a legitimate expression.

Q. (True or False) All the examples in the list below are legitimate expressions:

A*X+B	X/2.-4.0
FRAME-FRAME	-FRAME

●●●

A. True

124 The "expression" is FORTRAN's method of telling the computer exactly what arithmetic operations you wish to have executed and exactly which quantities to use. The expression A+B says "add the quantities A and B," while the expression -X says "find the negative of the quantity called X".

Q. If the numeric quantity -2.5 is represented by the name ROOT, the expression -ROOT will have a computed value of ________.

●●●

A. +2.5
(-(-2.5) equals +2.5)

125 An expression defines a series of related mathematical operations which the computer is to carry out. Because integer and real values must be handled quite differently by the computer, quantities used in an expression will normally be of the same mode. If they are not, a "mixed-mode" expression results and the computer cannot perform the desired operations without altering the quantities involved to make them agree in mode. (Just how this is done will be explained in later frames.) The normal way of writing expressions, therefore, is to write them without mixing modes. (There are some exceptions which will be explained as you progress through the text.)

Q. (True or False) X+1 is a mixed mode expression.

●●●

A. True.

126 The expression X+1 is indeed mixed mode: the variable X is a real quantity and the constant 1 is an integer number.

Q. The expression X+1 can be changed and written in real mode as _______.

●●●

A. X+1. (Notice the decimal point)

127 Integer and real numbers look quite different to the computer. It would be impossible to perform normal operations with numbers of opposite modes. Therefore expressions are normally consistent in mode.

Q. (True or False) The expression 2*I-3.*X is a mixed mode expression.

●●●

A. True

128 One of the exceptions to mixing modes is exponentiation. Specifically, real quantities may be raised to either real or integer exponents, and integer quantities may be raised to either real or integer exponents. Thus, all of the following expressions are acceptable: X**2, A**5.0, J**2.0, K**3, Z**N.

Q. (True or False) Exponents may not differ in mode from the base number.

●●●

A. False

129 Incidentally, exponents themselves may be expressions. That is, an exponent may be a constant, variable, or an expression involving more than one quantity. The latter case, however, will always require an extra set of parentheses. For example, X**2, X**N, or X**(2*N - 1) are all legitimate.

Q. Using an integer expression for an exponent, write an expression to compute the formula u^{2i+j}

●●●

A. U**(2*I + J)

130 While we're talking about exponents, it should be noted that, wherever possible, FORTRAN exponents should be integer in mode, regardless of the mode of the quantity being raised to the indicated power. A real exponent should be used only when a fractional exponent is necessary.

Q. The expressions X**2.0 and X**2 are exactly equivalent mathematically, but _______ (choose one) is preferred.

●●●

A. X**2 (integer exponent)

131 An example of the use of a real exponent would be the computation of a fourth root which might be expressed as X**0.25. The need for real exponents seldom arises so a good rule of thumb would be to avoid their use and stick to integer exponents for either mode of expression.

132 So far, reference has been made to the FORTRAN Arithmetic statement without actually defining it. An Arithmetic statement consists of a variable name, followed by an "equal" sign, followed in turn by any desired expression.

Q. The variable name and the expression of an Arithmetic statement are separated by an " _______" sign.

●●●

A. equal

133 All arithmetic statements conform to this format. The computer executes the following steps for any Arithmetic statement: "compute the complete expression on the right of the equal sign and assign that computed value to the variable whose name appears on the left of the equal sign."

Q. In the Arithmetic statement A = B*C/D the last step the computer does is to assign the expression's computed value to the variable _______.

•••

A. A

134 An expression, remember, is any desired combination of variables, constants, and operation signs describing a series of arithmetic operations. The Arithmetic statement always has some expression on the right of the equal sign.

Q. In executing an Arithmetic statement, the computer first obtains the computed value of the _______.

•••

A. expression

135 Once the value of the expression on the right of the equal sign has been obtained by the computer, that quantity becomes the value of the variable on the left of the equal sign. For example, if the variable A has a value of 5.2 at the time the statement Y = A + 1. is executed, the variable Y will end up with the value 6.2.

Q. Assuming the value for A given above, the value of X after executing the statement X = 2.*A is _______.

•••

A. 10.4

136 The format of the Arithmetic statement is limited in that only a single variable may appear on the left of the equal sign. The programmer may construct almost any expression on the right of the equal sign, however.

Q. (True or False) The statement A + B = X*Y is an Arithmetic statement.

•••

A. False (only a single variable is permitted on the left of the equal sign)

137 You were told a while back that there were several ways of defining the value of a variable; the Arithmetic statement is one such method.

Q. After executing the statement Y = -0.55 the value of the variable Y will be _______.

●●●

A. -0.55

138 After executing an Arithmetic statement the variable on the left of the equal sign will have the value of the computed expression, regardless of what the previous value of that variable may have been.

Q. If the statements B = 4., A = 3., and B = A**2 were executed in that order, the final value of B would be _______.

●●●

A. 9.

139 To review, then, you have been shown that an Arithmetic statement consists of a variable, an equal sign, and some expression. The expression may be a single constant or variable or a complex combination of operations. The statement is executed in a series of two steps: the expression, however complicated, is computed and that computed value becomes the value of the variable on the left of the equal sign.

140 Q. Assume the following statements are executed in the order written. After execution by the computer show the values of the variables A, B, C, X, and Y.

```
A = 2.0                    A ______
B = -3.0                   B ______
C = -2.0                   C ______
X = 2.0                    X ______
Y = A*X**2+B*X+C           Y ______
```

●●●

A.
A 2.0
B -3.0
C -2.0
X 2.0
Y 0.0

141 If your answers were correct, skip to frame 149. If your answers did not agree with the ones shown above, continue to the next frame for an explanation.

142 The question showed a series of Arithmetic statements to be executed in the order written. The first such statement A = 2.0 tells the computer to assign the value of the expression (2.0) to the variable called A.

Q. The value of A after executing that statement must be ________.

●●●

A. 2.0

143 Similarly, the next three statements: B = -3.0, C = -2.0, and X = 2.0 replace the former values of B, C, and X with the corresponding expression values.

Q. The value of X after completing the statements above must be ________.

●●●

A. 2.0

144 The last statement Y = A*X**2+B*X+C tells the computer to perform the indicated operations using the most recently defined value of each variable.

Q. (True or False) After the statement X = 2.0 is executed the variable X will have the value of 2.0 regardless of its previous value.

●●●

A. True

145 The expression in the final statement was A*X**2+B*X+C. According to the rules of hierarchy this means: compute X**2(4.0), multiply this by A (2.0 times 4.0 equals 8.0), multiply B*X (2.0 times -3.0 equals -6.0), and add the indicated quantities (8.0+(-6.0)+(-2.0) equals 0.0).

Q. The computed value of the expression shown above is _______.

●●●

A. 0.0

146 That computed value, 0.0, of the expression then becomes the value of the variable on the left of the equals sign: in this case, the variable name is Y.

Q. (True or False) The expression on the right of the equals sign in any Arithmetic statement must be completely computed before any value is assigned to the variable on the left.

●●●

A. True

147 Q. Given the following statements to be executed in the order written:

```
X = 3.0
Y = 2.0
A = X**2-Y**3
```

Show the values of the variables X _______, Y _______, and A _______ after execution of all three statements.

●●●

A. X 3.0
Y 2.0
A 1.0

148 If your answers agree with the ones shown, continue to the next frame. If your answers disagree with the correct answers, see your advisor.

149 Perform Exercise 1.5 in your problem book.

When an Arithmetic statement is executed the most recently defined values of the indicated variables are used.

Q. Write an Arithmetic statement that will set the value of a variable named G equal to the value 32.174: _______.

●●●

A. G = 32.174

150 The execution of an Arithmetic statement changes the value of only one variable - the one on the left of the equal sign. All variables used on the right of the equal sign remain the same value.

Q. (True or False) Before the statement S = S0+Y0*T+.5*G*T**2 can be successfully executed (with the correct answer) the variables S0, Y0, G, and T must have their values defined.

●●●

A. True

151 Remember, the expression on the right of the equal sign normally will contain quantities of only one mode (except for exponents). When this is not done, a "mixed expression" results.

Q. Identify the single mixed expression in the examples shown below (by letter) _______.

a) (A+B)**(N-2*J)
b) I*J**LUMP
c) ABLE*BAKER**K-1

●●●

A. c) ABLE*BAKER**K-1 (ABLE and BAKER versus 1)

152 The mode of the expression is determined by the mode of the quantities involved. If the expression contains real variables and constants (even if integer exponents are used), it is considered a real expression.

Q. The expression F**(I+J - K*M-1) is a _______ expression.

●●●

A. real

153 If an expression contains integer variables and constants, it is considered to be an integer expression. Remember, integer arithmetic involves only whole number quantities. The results, therefore, are also integer quantities.

Q. The expression INDEX**(NUMBR-1) is an _______ expression.

●●●

A. integer

154 The fact that integer arithmetic produces integer results may cause errors under a certain condition. Addition, subtraction, and multiplication of integers can produce only integer values anyway, but division of integers may result in numbers with fractional parts.

Q. The result of dividing 5 by 2 is _______.

●●●

A. 2 1/2

155 Integer arithmetic can permit only integer results, even for division operations. If a division operation results in a value with fractional parts, the fraction is dropped without rounding (truncated) to the nearest whole number. Thus, in the computer's integer arithmetic, 5/2 comes out as 2.

Q. If the result of an integer division were 1.9999999 (true result) the computer would consider the result to be _______.

●●●

A. 1 (dropping the fractional part without rounding)

156 The reason for warning you of this characteristic is to help you avoid harmless-looking situations that could produce incorrect results. For example, assume I, J, and K have the values 5, 2 and 4, respectively. The expression I/J*K might appear to compute $\frac{(5)(4)}{2} = 10$, but due to the hierarchy rules the result is (5/2)(4) which equals 8 in integer arithmetic.

Q. The expression X**(1/2) is not equivalent to X**.5 because the exponent in the first example (in integer terms) is _______ (number).

●●●

A. zero (0)

157 In review, then, Arithmetic statements are used in FORTRAN programs to evaluate an expression (anywhere from a single quantity to a complicated series of operations) and assign that value to an indicated variable. These statements may be used in a program to initialize variables, change the values of variables, or provide the steps to solve a mathematical formula. Most actual arithmetic is done in the real mode while special purpose operations such as exponents and counting may use integer mode.

158 Perform Exercise 1.6 in your problem book.

An Arithmetic statement is executed in two distinct steps. That is, the expression is computed with no regard to the variable on the left of the equals sign. Thus, FORTRAN rules permit quantities on opposite sides of the equal sign to be of opposite modes if desired.

159 An Arithmetic statement such as A = I*J-K is perfectly legitimate under FORTRAN rules. The statement is executed as follows: compute the integer expression (in integer arithmetic steps, naturally), convert the result to real form (the computer changes the form of the number), and finally assign this real value to the variable A.

Q. (True or False) The variable on the left of the equal sign in an Arithmetic statement need not be of the same mode as the expression on the right.

●●●

A. True

160 The general rule describing the execution of an Arithmetic statement becomes, then: "the expression is completely computed in its basic mode and that value is assigned to the variable on the left of the equal sign, converting its mode if indicated".

Q. Under this rule, the value of the variable K after executing the statement K = 1.999999 will be _______.

●●●

A. 1 (the value 1.999999 is converted to an integer)

161 For example, you might wish to initialize the variable X with a value of one and the statement X = 1 is perfectly legal. This is not good programming, however, since the computer must go through the steps of converting the constant (1) to real form (1.), thus wasting time.

Q. To avoid wasting time setting the value of X to a value of one the statement should be _______.

●●●

A. X = 1.0 or X = 1.

162 This feature of changing modes is useful under special conditions, but care should be taken to see that this doesn't happen when it is not intended. This would happen when a variable is accidentally assigned an integer name when a real name is intended.

Q. (True or False) The statement I = V/R is legitimate but will produce an integer result for the variable I.

●●●

A. True

163 Remember, when real quantities are converted to integer mode form, they lose their fractional parts without rounding. A result that was intended to be 3.8759 might accidentally end up as 3 if the mode were accidentally changed from real to integer by an Arithmetic statement.

Q. Care must be taken in assigning variable names (with respect to the first letters of the names) so that the result does not end up in the wrong _______.

●●●

A. mode

164 If you accidentally mix modes in an expression, many FORTRAN systems (1401, 1440, 1460, 7040/44) for example) will not permit that statement to be converted to machine language since it contains a mixed mode expression. As you may recall, all FORTRAN statements prepared by the programmer are converted to machine language statements by the computer during a translation or compiling stage. (See the Introduction section of this volume on "Communicating with the Computer"). Most FORTRAN systems will, at this time, perform an edit of the incoming programmer statements to check them for adherence to the rules of FORTRAN. In the 1130 if a mixed expression is found, 1130 FORTRAN, unlike many FORTRAN compilers, will allow the mixed mode statement to be translated into machine language.

Q. (True or False) The statement ALPHA=BETA*K+1.0 is a permissible 1130 FORTRAN statement.

●●●

A. True

165 Even though the 1130 will not reject mixed mode expressions, it cannot perform arithmetic operations with numbers which are not of the same mode. Therefore, when a mixed mode expression is encountered, the 1130 FORTRAN translator converts integer quantities in a mixed expression to real quantities and performs the arithmetic in the real mode.

Q. In a mixed mode expression, _______ quantities will be converted to _______ quantities and the arithmetic will be performed in _______ mode.

●●●

A. integer, real, real

166 According to the preceding rule, the mixed mode statement A=B+2+C**3+N will become equivalent to A = B+2.0+C**3+ (the real equivalent of the integer variable N. If N=5, the real equivalent would be 5.0).

Q. Write an equivalent real mode statement for the mixed mode statement X = X+1+3/2.+Q

●●●

A. X = X+1.0+3.0/2.+Q

167 Because the 1130 converts mixed mode expressions automatically, the programmer may end up with a different result from what he had intended. The best plan, therefore, is to check all FORTRAN statements carefully to make sure that they correctly express the intentions of the programmer.

168 Since an Arithmetic statement is executed in two phases (compute the expression and assign the value to the variable on the left of the equal sign), the expression on the right may contain the same variable name that appears on the left of the equal sign.

Q. (True or False) By the above rule the statement <u>I = I+1</u> is legal.

●●●

A. True

169 In this sense the Arithmetic statement is not an <u>equation</u> and the equal sign really means "is replaced by" rather than "equals". Thus the statement <u>I = I+1</u> is perfectly legal and means to the computer that "the quantity <u>1</u> is added to the quantity called <u>I</u>, and that result replaces the old value of the quantity called <u>I</u>".

170 This form of statement is very common in programming. In the statement I = I+1 the quantity I can be considered a counter which has 1 added to it each time this statement or a similar one is executed. The value of I may indicate how many times the statement has been executed, for example.

Q. If the variable N has the value of 63 before executing the statement N = N-3, it will have a value of _______ after the statement is executed.

●●●

A. 60

171 The next few frames will present some simple examples of Arithmetic statements and explain the execution of each.

172 The examples in the next few frames will involve very simple steps intended to demonstrate exactly what goes on when an Arithmetic statement is executed.

Statement:	A = 3./2.	Computer divides the constant 3. by the constant 2. and places the result (1.5) in A.

173

Statement:	A = 3/2	Computer divides (integer mode) the constant 3 by the constant 2; the result is 1 (truncated) which is converted to real mode (1.0) and placed in A.

174 Up to now variables and their associated variable names have been defined as a way of referring to a single quantity by means of a symbolic label. In the FORTRAN language a variable may represent a list or array of numbers instead of a single quantity.

175 When a single variable name is to represent a list of numbers there must be a means of referring to a specific member of that list. FORTRAN accomplishes this through the use of subscripts. If you know what subscripts are you may skip the next two frames.

176 Suppose we have a list of 5 numbers. Let us call the list MAX. Our list called MAX might look like this:

105
099
217
004
198

Let's assume that we need the third value in the list (217) for a calculation. Before we can use the third value, however, we must have a way of identifying it. That is, we must be able to tell the computer that we need precisely the third value and not any other. In mathematics the third value in the list would be uniquely identified by writing the name MAX_3. The 3 printed to the lower right of the name MAX is called a subscript. Because we cannot print lowered numbers on computers, FORTRAN uses parentheses to denote subscripts. Thus MAX(3) is equivalent to the mathematical notation MAX_3.

Q. FORTRAN uses _______ to denote subscripts.

●●●

A. parentheses

177 In the following frames you will learn the rules for using and forming subscripts. Although the details may seem confusing, just keep in mind that the purpose of a subscript is simply to identify an individual number in a group of numbers.

Q. In the preceding frame, how would the value 198 be identified?

●●●

A. MAX(5)

178 A "subscript" in FORTRAN notation consists of a constant or variable (or limited expression) attached to a variable name and contained in parentheses. X(I), NAME(20), and ARRAY(N+1) are examples of variables with subscripts, and as such are called "subscripted variables."

Q. Variables which refer to a list or array of numbers are called _______ variables.

●●●

A. subscripted

179 The subscript serves to indicate exactly which number in the list is being referenced. For example, the subscripted variable X(25) refers to the 25th number in the list called X. The use of the form X(I) will refer to "Ith" number in the X list, depending on the value of the variable I.

Q. The expression A(7)*B will tell the computer to multiply the _______th number in the A array by the quantity B.

●●●

A. 7

180 Thus, numbers which can be meaningfully referenced in the list form can be referred to in a program by the subscripted variable form. A variable name followed by a pair of parentheses containing a subscript will refer to a particular quantity in a list, the order in the list being determined by the value of the subscript.

Q. If the variable I has a value of 3, the subscripted variable G(2*I) refers to the _______th number in the G array.

●●●

A. 6

181 The list of numbers which may be referenced by a single variable name is of a predetermined length. A special statement covered by a later chapter is used to declare all variables whose names will be used to represent such lists as well as indicating the exact length of each list.

Q. A subscript indicates a particular member of a list according to the numerical _______ of the subscript.

●●●

A. value

182 Since all lists have certain predetermined lengths, care must be taken to ensure that no subscript value exceeds the length of the list to which it refers. For example, a subscript which refers to a list called X which consists of twenty numbers should not exceed a value of twenty.

Q. A subscript used with the variable X described above may have a value from _______ to _______.

●●●

A. one, twenty

183 A subscripted variable may be either integer or real mode, but all subscripts are in the integer mode. This is the second exception to the mixed expression rule: real variable names must have an integer subscript if subscript notation is used.

Q. (True or False) The subscripted variable Y (T) is not a legal use of subscript notation.

●●●

A. True (subscript must be an integer quantity)

184 One final rule regarding subscripts -- constants and variables used in forming subscripts must be unsigned. Or to put it more precisely, they must not be negative values. Thus, X(2) is legal, but X(-5) is not. Incidentally, a subscript can never have a value of zero.

Q. Is OMEGA(-M) a legal subscript?

●●●

A. No

185 To review, then: expressions of either mode may contain variables with subscripts attached which refer to a particular single quantity in a list; the variable(s) must be consistent with the mode of the expression but regardless of the expression's mode the subscript must be an integer; a particular subscripted variable refers to a specific number in the list by order according to the value of the subscript; the constants and variables used in subscripts must be unsigned; and the subscript can never have a value of zero nor a value greater than the number of items in the list.

186 Subscripts are either constants or variables, and all are in the integer mode. That is, to refer to the 101st number in an array, the subscript 101 is used, or to refer to the Nth number in a list the subscript N is used with its value computed elsewhere. Certain limited expressions are permitted as subscripts also, and their rules are listed in the following frame.

187

Permitted expression form	Example
Variable plus or minus a constant	X(N-1) or X(N+1)
Constant multiplied by a variable	X(2*N)
Constant multiplied by a variable plus or minus a constant	X(2*N+5) or X(2*N-1)

The above forms of subscripted expressions must be rigidly followed. For instance, although a constant multiplied by a variable might seem to be equivalent to a variable multiplied by a constant, only the former expression is permitted as a subscript.

Q. (True or False) J(2-M) is a valid subscript.

●●●

A. False. J(M-2) would be correct (variable minus a constant).

188 Q. Identify the invalid subscripts in the following list:

a) N(IMAX)
b) I(19)
c) BINGO(A+2)
d) X(-I)
e) LUMP(2+JOB)
f) GAMMA(8*IQUAN)
g) BOND(JAMES)
h) PI(IN+SKY)
i) RHO(5*L+7)
j) KAPPA(I+2.)
k) N(-2*J)
l) ZETA(I(3))
m) KAPPA(0*M)

●●●

A. c) A is not an integer variable.
d) The variable may not be signed.
e) For addition, the variable must precede the constant, thus, JOB+2 is correct.
h) SKY is not an integer variable, and, even if it were, it could not be added to another variable in a subscript expression.
j) 2. is not an integer constant.
k) The constant must be unsigned.
l) This is a tricky one. A subscript can be attached only to a variable name and not to another subscript. To put it another way, subscripts cannot themselves be subscripted.
m) A subscript can never have the value of zero.

189 Perform Exercise 1.7 in your problem book.

A computer, like a desk calculator, can hold a certain size of number as a maximum. In the 1130 the size of numbers is $2^{\pm 127}$ (approximately $10^{\pm 38}$) in the real mode and $\pm 2^{15} -1$ in the integer mode.

Q. In real calculations on the 1130 the largest allowable quantity is _______.

●●●

A. 2^{127} or 10^{38}

190 The restriction on the size of numbers applies to both variables and constants. The value of any real variable or constant must be within the range of $10^{\pm 38}$ and all integer variables and constants must have values between ($\pm 2^{15}-1$) which equals ±32767.

Q. (True or False) The value 42100 is permissible in integer calculations on the 1130.

●●●

A. False (value exceeds limit)

191 In addition to size considerations, however, there is also the question of allowable significant digits. Although the real mode allows a number of the 10^{38} magnitude (a 1 followed by 38 zeroes), a maximum of either 7 or 10 significant digits is permissible.

Q. The maximum number of significant digits in the real mode is either _______ or _______.

●●●

A. 7 or 10

192 The choice of either 7 or 10 significant digits is made by the programmer just prior to the time the source program is compiled. He will inform the compiler program whether 7- or 10- digit significance should be built into the object program. The object program will then be geared to handle the selected number of significant digits.

193 You might think that the 10 digit selection should always be made because it seems more powerful, but, generally speaking, there is a trade-off between higher significance and the amount of computer storage used by the object program. The more significance you have, the more space your numbers use inside the computer.

194 You might reasonably question the value of having a large number like 10^{38} if only 7 or 10 digits of that number are significant. However, in many scientific calculations, accuracy to seven or ten places is sufficient because physical measurement may not be possible, or meaningful, beyond that.

195 You have been shown that a real constant must be represented by the number itself with decimal point included (even when it is a whole-number quantity). A real constant may also be represented in FORTRAN as a constant times a power of 10.

196 In ordinary notation very large or very small numbers are often noted this way, as in 3.5×10^7 which means 35000000. With real FORTRAN constants this form may be used with the letter E following the constant, followed in turn by the desired power of 10, as in 3.5E+7 (Note: The power of 10 must be expressed as an integer constant in one or two digits.)

Q. The FORTRAN constant 3.5E+4 stands for a value (written out) of ______.

●●●

A. 35000.

197 The same notation may be used with negative powers of ten for very small numbers. For example, the constant 3.5E-7 stands for the number 0.00000035. The general form is, again, any real constant followed by the letter E followed in turn by the desired power of ten (positive or negative) by which the original constant is to be multiplied.

Q. A shorthand version of the number 0.000001 might be ______ (using the form described above with the smallest whole number constant).

●●●

A. 1.E-6

198 You might ask, "How is this form different than the double-asterisk sign, such as 10.**6"? This very logical question is answered by the fact that the double-asterisk sign tells the computer to actually compute the indicated power while the E notation defines the number directly as in any constant.

Q. Assuming that 10^{38} is the largest number permitted in the computer, show how a constant of this value would be represented in the E notation.

●●●

A. 1.E38 (or 1.E+38; the "plus" sign may be omitted)

199 This example illustrates the usefulness of the E notation: without such a shorthand form, the number 10^{38} would be written: 100000000000000000000000000000000000000.

200 E notation has still another important use. By using E notation, the programmer can write constants which are larger than 7 digits, although only 7 digits of the constant will be significant (assuming we have selected 7-digit significance). If E notation is not used, however, a real constant cannot exceed 7 digits.

Q. Would 10000000. be an acceptable real constant?

●●●

A. No. It exceeds 7 digits. E notation must be used to represent this constant as a real number.

201 To repeat, it is illegal to define (write) a real constant that has more than seven digits without using E notation (again assuming we have selected 7-digit significance). Therefore, if a number like ten million (10,000,000) is to be used as a real constant, E notation must be used. 1.E7 would be a legal way of writing the number ten million. Note, however, that even though this method of expression allows us to create the constant ten million, the constant itself is still limited to 7 significant digits. E notation doesn't give us any additional significant digits; it simply allows us to express numbers of great magnitude which cannot be legally expressed directly (i.e. without E notation).

Q. Write the real constant which best represents 69870060042.

●●●

A. 6987006.E4 (Note that the last four digits of the number cannot be retained - the limit is seven significant positions in the real mode. If we had selected 10-digit significance, however, this number could best be written as 6987006004.E1)

202 A good deal of variation is permissible in writing E (or D) notation. These variations are illustrated in the following example which shows different ways of representing the number 7000.

```
7.E3          7.0E+03
7.0E3         +7.0E+03
7.0E03        7.0E 03
```

As you can see, the constant may or may not have a zero following the decimal point, the exponent may or may not have a leading zero, and the exponent may or may not have a plus sign if it is positive.

Q. What number does 7.0E-03 represent?

●●●

A. .007

203 Here is a review of the rules for defining real constants (assuming selection of 7-digit significance):

1. A real constant may be positive or negative, and it must be in the allowable range. If E notation is used, the range is from 10^{-38} to 10^{38} with 7 significant digits. If E notation is not used, a real constant is limited to 7 digits.

2. Real constants may not contain embedded commas or blanks.

3. Real constants may be followed by a decimal exponent written as the letter E, which in turn is followed by a signed or unsigned one or two-digit integer constant. This decimal exponent permits the expression of a real constant as the product of a real constant times 10 raised to the desired power.

Q. Identify the invalid real constants in the following list:

a) +0.
b) -91437.143
c) 1.E
d) 7.0E+10
e) 0.0
f) 7.0E+013
g) 5,764.198
h) 23.5E+97
i) 19761.25E+1.
j) -7.2E-03
k) -9999.9900

●●●

A. b) Exceeds seven digits.
c) Missing a one or two-digit integer constant following the E. Note that this is <u>not</u> interpreted as 1.0×10^1.
f) E is followed by a 3-digit integer constant.
g) contains a comma.
h) Value exceeds the magnitude permitted; that is, $23.5 \times 10^{97} > 10^{38}$.
i) E is not followed by an integer constant.
k) Exceeds seven digits.

204 This completes the material on Arithmetic statements. On page 27 of your problem book you will find the examination on the chapter you have just finished.

REFERENCE INDEX

R29-0101-0

Printed in U.S.A.

R29-0101-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York