IBM

SYSTEM/360 COBOL

COBOL Program Fundamentals

Text

Programmed Instruction Course

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

PREFACE

The general objective of this book is to teach students to read System/360 COBOL programs with a high degree of comprehension. Some of the major topics dealt with are: the elements of the COBOL language -- reserved words, programmer-supplied names, symbols, literals, level numbers, and pictures; the organization of COBOL programs, and how the divisions of a program are related to each other; and the interpretation, in detail, of entries in each program division.

Additional information needed in order to compose original COBOL programs has deliberately been omitted. It will be found in other publications of this series. The subject matter has been arranged in this way to give the student a chance to see what makes up a complete COBOL program, before he is required to write one. The student can concentrate on the meaning and function of program entries, without being concerned about their formats and alternate options.

Several other topics are not discussed in this publication; for instance, the Report Writer feature, the Sort feature, floating-point operations, direct-access devices, and so on. However, care has been taken to teach concepts in a way that allows such topics to fall right into place later, with no contradictions or inconsistencies.

This programmed instruction textbook is designed to be studied in conjunction with the COBOL Program Fundamentals reference handbook (Form R29-0206). The reference handbook contains technical information, and is kept by the student for reference purposes after he completes his studies. This textbook gives the student reading assignments in the reference handbook, explains the concepts that make sense of the technical information, and lets the student apply the information to practice exercises and problems. In short, the textbook teaches the student to find and to apply the information that is in the reference handbook. The student learns to look up information when he needs it, instead of memorizing it.

ACKNOWLEDGEMENT

The following information is reprinted from COBOL-61 EXTENDED, published by the conference on Data Systems Languages (CODASYL), and printed by the U. S. Government Printing Office.

This publication is based on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command, United States Air Force
Bureau of Standards, Department of Commerce
David Taylor Model Basin, Bureau of Ships, U.S. Navy
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company
Burroughs Corporation
International Business Machines Corporation
Radio Corporation of America
Sylvania Electric Products, Inc.
Univac Division of Sperry-Rand Corporation

In addition to the organizations listed above, the following organizations participated in the work of the Maintenance Group:

Allstate Insurance Company
Bendix Corporation, Computer Division
Control Data Corporation
DuPont Company
General Electric Company
General Motors Corporation
Lockheed Aircraft Corporation
National Cash Register Company
Philco Corporation
Royal McBee Corporation
Standard Oil Company (N.J.)
United States Steel Corporation

This manual is the result of contributions made by all of the above-mentioned organizations. no warranty, express or implied, is made by any contributor or by the committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

It is reasonable to assume that a number of improvements and additions will be made to COBOL. Every effort will be made to insure that the improvements and corrections will be made in an orderly fashion, with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

Procedures have been established for the maintenance of COBOL. Inquiries concerning procedures and methods for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

The authors and copyright holders of the copyrighted material used herein: FLOW-MATIC (Trade-mark of the Sperry-Rand Corporation), Programming for the UNIVAC ® I and II, Data Automation Systems © 1958, 1959, Sperry-Rand Corporation; IBM Commercial Translator, Form No. F28-8013, copyrighted 1959 by IBM; FACT, DSO 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell; have specifically authorized the use of this material, in whole or in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

Any organization interested in reproducing the COBOL report and initial specifications in whole or in part, using ideas taken from this report or utilizing this report as the basis for an instruction manual or any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention "COBOL" in acknowledgement of the source, but need not quote this entire section.

TABLE OF CONTENTS

STUDENT INSTRUCTIONS

1. This textbook must be used in conjunction with a reference handbook, form R29-0206. If you don't have the reference handbook, get one before you go any further.

2. The general objective toward which you will be studying, and the division of information between this textbook and the reference handbook, are explained in the Preface of this book. Have you read it?

3. The purpose of this textbook is to guide and direct your study of System/360 COBOL. It is not meant to serve as a reference book - that's where the reference handbook comes in. In other words, once you have read this textbook, the odds are that you will have no further use for it; as a result, it is meant to be used by more than one student. The point is: Don't make any marks or notes in this book.

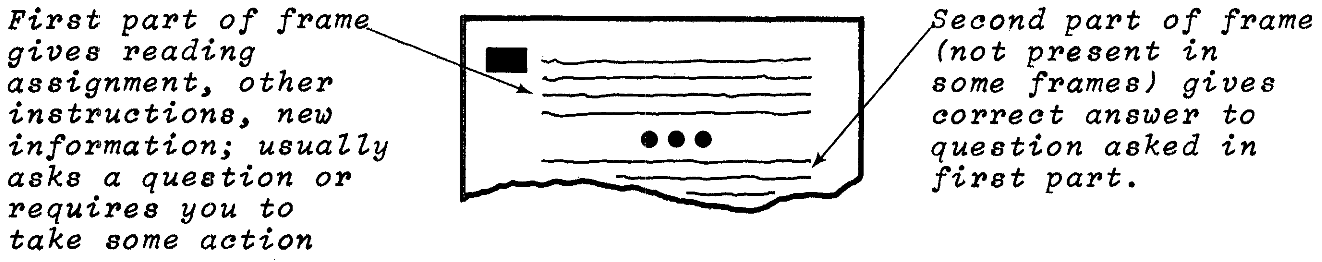
4. The reference handbook, on the other hand, is yours to keep. If you care to make any notes, make them in the reference handbook.

5. There are ten lessons in this book. In general, the lessons consist of reading assignments and questions. You will read specified parts of the reference handbook, and then answer questions about what you just read, or about something that you had previously read.

6. There may be several reading assignments in a lesson, not just one at the beginning. The reading assignments give the major titles and subtitles of what you are to read, rather than page numbers. The major titles appear in the table of contents in the reference handbook.

7. A lesson may also give you new information to supplement or explain what you read in the reference handbook.

8. Each lesson is broken up into a number of frames, which are simply convenient instructional steps. You proceed through the lesson one frame -- one step -- at a time. The format of a frame usually looks like this (in miniature):

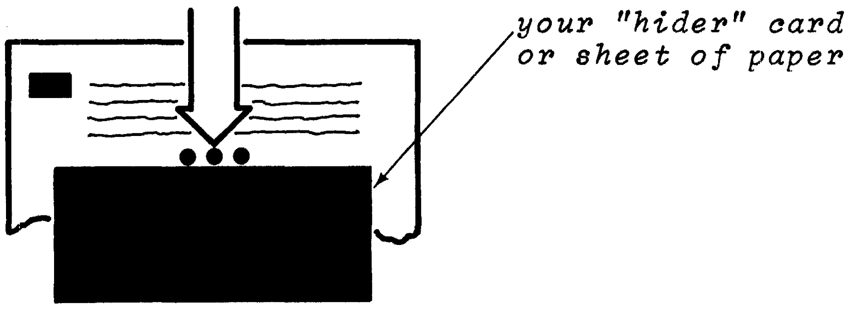

First part of frame gives reading assignment, other instructions, new information; usually asks a question or requires you to take some action
Second part of frame (not present in some frames) gives correct answer to question asked in first part.

(Continued on next page)

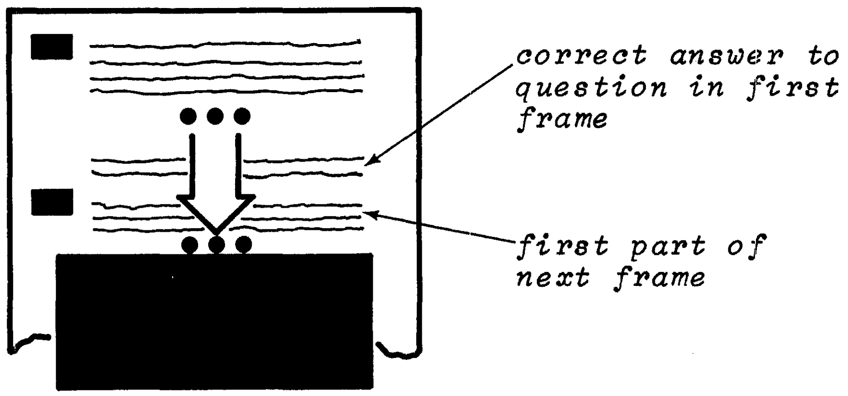
9. The first part of every frame is ended by a group of three dots. If the frame asks you a question, the correct answer is printed on the same page, below the three dots. As you study each frame, you must hide the correct answer from yourself, so that you will feel challenged to come up with your own answer -- and thereby learn the subject, instead of just reading words.

10. Use an ordinary sheet of paper or a card to hide the correct answers. Just make sure that the paper is heavy enough so you cannot see through it. (No onion skin or tracing paper allowed!)

11. Start each page by putting your "hider sheet" at the top. Then slide your sheet down until you just uncover a set of three dots, as illustrated below.



12. Then read the first part of the frame, and formulate your answer to the question or problem it poses. When you have your answer clearly in mind, slide the "hider" sheet down to the <u>next set of three dots</u>. This will not only reveal the correct answer, but also uncover the first part of the next frame.



13. Study at your own speed, but don't spend too much time with any one frame. Too much concentration on an isolated detail may destroy your comprehension of a general concept. (Which means: Don't let the trees block your view of the forest.)

14. You will find additional instructions, along with helpful remarks and the author's opinions, printed in italics in some of the frames of the book.

LESSON 1

1 *If you have ever studied a foreign language, it is possible that somebody challenged you to say something in the language when you had barely finished your first lesson! Well, this sort of thing happens when you study programming languages too. Although people are not likely to challenge you to say something in COBOL, you may get questions like, "What is COBOL, anyway?" For you to say that you're not sure, would sound like a forestry student admitting that he isn't sure what a tree is! In self defense, the thing for you to do first is to learn a few general facts and ideas about COBOL. So, our first lesson is concerned with introductory generalities.*

In your reading, you should pick up the information that you need in order to answer these two questions in your own words: What is COBOL? What are its purposes?

Reading assignment: INTRODUCTION TO COBOL
Origins
Aims
Differences

All reading assignments are in the reference handbook that accompanies this textbook. Always complete the reading assignment before going on to the next frame. The frames that follow will ask you to apply what you have read, or will supplement your reading with additional information.

●●●

2 *There are several types of frames in this book. Most of them have this much in common: they require you to formulate an answer mentally. This particular frame asks you to criticize a statement, based on what you have just read in the reference handbook. Your answer will probably be different from the printed answer, but it should mean the same. Do not write your answer in the book.*

What is wrong with this statement: "COBOL was invented by IBM".

●●●

No single person or company can take credit for inventing COBOL. IBM participated in the development of COBOL, together with several other computer manufacturers and users.

3 *The answer that is required by a frame may be a matter of opinion, judgment, or fact. In this frame, you are expected to recall a fact from your reading; however, if you cannot remember it, you may look it up in your reference handbook.*

In what year were the original specifications for COBOL drawn up?

●●●

1959

4 *Frames like this one require you to think of* one or more *words or numbers that complete the sentence. The length of the blank space will always be the same, so it will not be a clue to the length of the answer. Do not write your answer in the book.*

The name "COBOL" is derived from the words _____.

●●●

COmmon Business Oriented Language

5 *In this type of frame, you are given a choice of answers, which are stacked in braces. Select the* one *best answer.*

The name "COBOL" is pronounced KO-BALL.
In other words, "COBOL" should be pronounced to rhyme with

$$\left\{\begin{array}{l}\text{noble}\\ \text{hobble}\\ \text{snowball}\\ \text{low bowl}\end{array}\right\}$$

●●●

snowball

6 *Here is another kind of frame that offers you a choice of answers. In this kind of frame, each choice is enclosed in brackets. Your job is to select all the correct answers. To do this, you must examine every choice, instead of merely looking for a correct answer, because more than one of the choices may be correct. Perhaps all of the choices are correct. Or none of them.*

In an effort to standardize programming, COBOL provides the programmer with [a standard program structure] [standard entry formats] [standard solutions for business data processing jobs].

●●●

a standard program structure AND standard entry formats

But not standard solutions to data processing jobs. Problem solving is the programmer's job; COBOL is a language in which he can express his solutions.

7 You can think of "Common Business Oriented Language" as COBOL's official title. Unfortunately, each one of the words of the title can possibly be taken in various ways, so they require a little explaining.

The title is intended to convey the basic purpose of COBOL: to be one language for all computers -- a standard language for programming business problems.

$$\text{COBOL is "common" in that it is } \left\{\begin{array}{l}\text{ordinary}\\ \text{well-known}\\ \text{conversational}\\ \text{shared}\end{array}\right\}.$$

●●●

shared (shared by all computers)

8 Although COBOL is common to all computers, you could not take a COBOL program prepared for one computer (say, an IBM 1410) and drop it into the card reader of a different computer (say, an IBM System/360). You would not have to write a new program, starting from scratch, but you would have to make some changes.

It would be a mistake, then, to say that COBOL is identical for all computers. But it would be equally wrong to exaggerate the differences in COBOL for different computers. What it amounts to, is that there are several versions of a common language, much like dialects of a spoken language.

Which statement best sums up this idea?

{Several different programming languages bear the name "COBOL".
All COBOL systems except System/360 COBOL are alike.
The similarities of COBOL systems far exceed their differences.
It is a mistake to use the name "COBOL" for different languages.}

●●●

The similarities of COBOL systems far exceed their differences.

9 The words "business oriented" also require a little explaining. They have three implications: first, that COBOL is business procedure oriented, rather than machine oriented; second, that COBOL is particularly applicable to business data processing problems, as opposed to scientific problems; and third, that COBOL is a language that businessmen can understand.

Because COBOL is not machine oriented, the COBOL programmer does not need to know

[what the operation code for adding binary numbers is]
[what data items make up his input records]
[what the general registers are used for]
[whether the output data is to be punched, or written on magnetic tape].

●●●

what the operation code for adding binary numbers is AND what the general registers are used for

The COBOL programmer does need to know the layout of data records, and the media on which they are recorded.

10 Since it is machine independent to a large degree, COBOL

$\left\{\begin{array}{l}\text{is outdated}\\\text{is not outdated}\end{array}\right\}$ by new developments in computer technology.

●●●

is not outdated

11 Consider this statement: "COBOL must be obsolete, because there have been revolutionary changes in computers since 1959". There are at least a couple of good reasons why this statement is false. One is the point made in the previous frame. Another reason why COBOL is right up-to-date is that the COBOL specifications have been _____ several times since 1959.

●●●

revised and improved

12 We say that COBOL is oriented to business procedures because a COBOL program consists of descriptions of (1) the procedures according to which data files are to be processed, (2) what the contents of the data files are, and (3) what input-output devices the data files are to be assigned to.

You have undoubtedly seen definitions of "computer program" that go something like this: A program is a series of machine instructions that direct the computer to perform a sequence of operations. Would you say that this definition applies to COBOL programs?

●●●

No. A COBOL program is a "program" in a broader sense -- it is a solution for a problem. (Ultimately, to be sure, a series of machine instructions will be compiled from the COBOL program.)

13 COBOL is also business oriented in the sense that it is particularly applicable to business data processing problems, as opposed to scientific problems. Here I am paying lip service to that unfortunately vague line between business and scientific computer applications. Tasks like preparing reports and updating files fall into the business category, to which COBOL is oriented. Tasks that involve trigonometric functions (sines, cosines, etc.) or Boolean algebra (logical ands, logical ors, etc.) are examples of tasks that fall into the scientific category.

Decide which category each of these tasks belongs to:

1. Calculating logarithms
2. Sorting records

●●●

Calculating logarithms is a task that most people put into the scientific category; sorting records is in the business category.

14 For the most part, business data processing involves moving data around -- putting data in and out, rearranging it, changing its appearance, comparing items of data, locating desired items, etc. Arithmetic operations are involved, but they are generally limited to adding, subtracting, multiplying, dividing, and occasionally exponentiating (raising a number to a power, or finding a root of a number). COBOL is designed for these kinds of operations.

Jobs that involve complicated mathematics are harder to program in COBOL. As a general rule, such tasks are in the scientific category.

Categorize each of these tasks as business or scientific tasks:

1. Computing a hyperbolic tangent
2. Searching a table for a particular data item
3. Calculating the area of a circle

●●●

The hyperbolic function involves mathematics that seems complicated to me, so I would call it scientific. Searching a table is definitely a business task. The formula for the area of a circle (πr^2) really calls for two simple multiplications (3.1416 x r x r), which puts it into the business category.

Notice that we begin to get into a gray area here. Finding the area of a circle is simple to program in COBOL, while finding the area under a hyperbola is not.

15 COBOL is "business oriented" in a third sense: a businessman can understand COBOL programs after a short introduction to the language, whether or not he is a programmer.

A sample COBOL sentence is printed below. Notice that it is practically self-explanatory. It illustrates that the COBOL language resembles the _____ language.

```
MULTIPLY UNIT-PRICE BY QUANTITY,
    GIVING TOTAL-PRICE.
```

●●●

English

16 Computers can't understand English, so what's the point of writing programs in English?

●●●

Programs written in English serve to communicate data processing procedures to people, in addition to serving as source programs for computers.

17 When a programmer writes a program in machine language or in an abstract symbolic language, he prepares a detailed explanation of what the program does and how the program does it. This supplementary write-up, called "documentation" of the program, often takes as much time to prepare as it took to write the program itself.

Since COBOL programs are similar to English, they

{ require more documentation than other programs
require just as much documentation
require less documentation
require no documentation at all } .

●●●

require less documentation

Although COBOL programs are sometimes referred to as "self-documenting", some documentation is almost always needed -- possibly just remarks and notes added to the program.

18 *There is danger of going overboard on the idea that COBOL is like English. If COBOL were completely self-explanatory, there would be no point in studying a course on how to read COBOL, would there? You have already seen one sample COBOL entry that was an easily readable sentence. In order to get a balanced point of view, let's look at some entries that are not quite so readable:*

1. *Certain elements of the COBOL language have symbolic meanings, as illustrated by the 03 and the 9(6)V99 found in the entry below.*

```
           03  SALES-VOLUME    PICTURE 9(6)V99.
```

2. *In some cases, the words are easy enough to read, but their meaning is not obvious. For instance, the words below mean "the data in this item is recorded in binary code" -- but a person must have studied System/360 COBOL in order to know that.*

```
                USAGE IS COMPUTATIONAL
```

3. *It is also possible for a programmer to misuse the language, by writing entries that are hard to understand. Perhaps the example below is a trifle extreme, but it will serve to show what happens when programmers use abbreviations that are meaningful only to themselves, a common error.*

```
       MOVE TKVHL TO AFPTD S7PFDOTT.
```

●●●

19 *Even though a COBOL program is not exactly the same as a story written in English, at least COBOL makes an attempt to approximate English. And one thing is certain: writing programs in COBOL is a far cry from writing programs in the actual language of the computer.*

Of course, the idea that programs need not be written in machine language is one of the main ideas behind programming systems in general. Although those ideas are not new to you, it will be worth your while to review some of the familiar terms -- and to see how they apply to the COBOL system in particular.

Reading assignment: COBOL PROGRAMMING SYSTEM TERMS

●●●

20 The ultimate objective of using the COBOL system is to produce correct, efficient machine language programs, which are called _____.

●●●

object programs

21 Object programs are produced by $\left\{\begin{matrix}\text{programmers}\\\text{computers}\end{matrix}\right\}$.

●●●

computers

22 COBOL programs are $\left\{\begin{matrix}\text{source programs}\\\text{object programs}\end{matrix}\right\}$.

●●●

source programs

23 Source programs are produced by $\left\{\begin{matrix}\text{programmers}\\\text{computers}\end{matrix}\right\}$.

●●●

programmers

24 Can a COBOL program be executed by a computer?

●●●

No, because it is not in machine language.

25 Every programming system includes a processor -- a program that directs the computer to produce an object program from a source program. In the COBOL system, the processor is called the _____.

●●●

COBOL compiler

26 The program that is compiled is the

$$\left\{\begin{array}{l}\text{COBOL program}\\ \text{COBOL compiler}\\ \text{source program}\\ \text{object program}\end{array}\right\}.$$

●●●

object program

Make sure that you have the relationship between these programs straight. An object program is compiled from a COBOL program. A COBOL program is a source program written in COBOL. The COBOL compiler is executed during compilation, to cause an object program to be compiled.

27 The System/360 used to compile an object program is the

$$\left\{\begin{array}{l}\text{object computer}\\ \text{source computer}\end{array}\right\}.$$

●●●

source computer

28 The object computer is a System/360 that is used to [compile] [execute] the object program.

●●●

execute ONLY

29 For a given System/360 COBOL program, the source computer and the object computer

[may be the same System/360]
[must be the same System/360]
[may be different System/360s]
[must be different System/360s].

●●●

may be the same System/360 OR may be different System/360s

This is strictly a matter of hardware configurations. A System/360 may be used both as the source computer and as the object computer provided that it contains the hardware required to execute the COBOL compiler and the object program. On the other hand, a System/360 Model 30 might be used to compile a program, while a System/360 Model 50 might be used to run the job.

30 *This review of terms was pretty superficial -- and deliberately so. There is nothing to be gained by spending any more time on a subject that you are already familiar with. However, you should feel fairly confident of your knowledge of these terms before you continue. You may find it useful to read the definitions in the reference handbook once again. Incidentally, if you didn't recognize the terms, or if our discussion of them seemed like a lot of gibberish, then that is a good indication that you do not have the background you need to put the rest of this course to good use; in that case, you ought to put this course aside for now, and study "Basic Computer Systems Principles" instead.*

●●●

LESSON 2

31 *So far, we have talked about COBOL in general terms. Now, let's get down to specifics, and find out what elements make up the COBOL language. I use the term "element" for the basic units that are found in COBOL programs: the various kinds of words, numbers, and symbols. "Element" is not an "official" term, but it will do the job in the absence of a better term.*

Find out how many elements there are, what they are called, and where they come from.

Reading assignment: LANGUAGE ELEMENTS
Examples of elements

●●●

32 How many elements are there?

●●●

six

33 See how many of the elements you can name.

●●●

reserved words, programmer-supplied names, symbols, literals, level numbers, pictures

34 Which elements are composed by programmers, following certain rules?

●●●

programmer-supplied names, literals, and pictures

35 Which elements are selected from fixed sets as they are needed?

●●●

reserved words, symbols, and level numbers

36 *At this point, you will learn detailed information and rules about each element in turn. Don't waste your time memorizing the rules; after all, you can always look them up when you need them. The important thing is to be aware of the areas in which rules exist, and how much latitude they give the programmers, or how severely they limit him.*

Reading assignment: RESERVED WORDS
- Types of reserved words
- Figurative constants
- Complete list of reserved words for System/360 COBOL

●●●

37 Reserved words are $\left\{\begin{array}{l}\text{selected}\\ \text{composed}\end{array}\right\}$ by programmers.

●●●

selected

38 Examine the reserved word list. Notice that most of the reserved words are simple English words, made up of letters only. A few of the words contain digits. Some contain hyphens.

Spaces $\left\{\begin{array}{l}\text{are}\\ \text{are not}\end{array}\right\}$ found in some of the words.

●●●

are not

Spaces, you will later learn, are used to separate words and other elements in entries, so they are never found within words.

39 The reserved words are an inherent part of the COBOL compiler's vocabulary, but of course, the words must be spelled exactly right in order for the compiler to recognize them. Exact spelling includes hyphens, too.

Under this rule, is WORKING STORAGE equivalent to WORKING-STORAGE?

●●●

No, it is not.

40 No substitutions are allowed in the reserved word list. Webster's Dictionary may prefer "numerical", but COBOL insists on NUMERIC.

Refer to your reserved word list to pick the correct spelling in each pair of words below:

1. ALPHABETIC or ALPHABETICAL?
2. PROGRAM-IDENTIFICATION or PROGRAM-ID?
3. IDENTIFICATION or ID?
4. ZEROS or ZEROES?

●●●

1. ALPHABETIC
2. PROGRAM-ID
3. IDENTIFICATION
4. Both ZEROS and ZEROES are correct.

You can see that there is a certain arbitrariness about the list of reserved words. COBOL programmers generally keep the list right at their elbows when they write a program.

41 *You will see many of the reserved words as you proceed through this course, in the context of the entries in which they are used. Therefore, we will not go down the list to discuss what each word is used for. However, one type of reserved word, the figurative constant, has been singled out for a little extra attention in the reference handbook. This has been done because the figurative constants ZERO and SPACE are found in many different entries.*

"Figurative constant" is a mouthful to say, but it is a reasonable term, since words like ZERO stand for constant data, rather than actually being the data. In other words, they are constants in a _____ sense, rather than a literal sense.

●●●

figurative

42 *The reference handbook discusses the figurative constants ZERO and SPACE, and their respective plural forms. The other figurative constants are HIGH-VALUE, LOW-VALUE, QUOTE and ALL (plus their plurals); these figurative constants have relatively specialized uses, which you will learn in due time.*

●●●

43 *Now let's turn to the subject of programmer-supplied names. "Programmer-supplied names" -- there is another mouthful for you! This term is useful to us, though, because we must distinguish names that are composed by programmers from names that are reserved words. We will give some attention first to the rules that govern the formation of programmer-supplied names; then we will see how programmer-supplied names and reserved words are used together.*

Reading assignment: PROGRAMMER-SUPPLIED NAMES
Rules governing programmer-supplied names
Examples of programmer-supplied names in an entry

●●●

44 Reserved words have preassigned meanings. By contrast, programmer-supplied names must be _____ within the program in which they are used.

●●●

defined

45 Programmers supply names for _____.

●●●

data items, data conditions, and procedures

46 According to the rules, programmers may compose names like PAYROLL, ACCOUNTS-RECEIVABLE, or 265. PAYROLL and ACCOUNTS-RECEIVABLE might be names of data items, data conditions, or procedures; however, 265 could only be the name of a _____.

●●●

procedure (because names of data items and data conditions must contain at least one letter, while procedure names can be composed entirely of digits)

47 Any letter or digit may be used in a name, but the only special character allowed is the _____.

●●●

hyphen

48 A hyphen is not allowed to be [the first character] [the last character] of a name.

●●●

NEITHER the first character NOR the last character may be a hyphen.

49 The word CODE is not an acceptable programmer-supplied name. Why not?

●●●

CODE is a reserved word, and therefore cannot be used as a programmer-supplied name. It violates the rule that a programmer-supplied name must not be spelled exactly the same as a reserved word.

50 Is RECORD-CODE an acceptable programmer-supplied name?

●●●

Yes. Even though it contains two reserved words, RECORD and CODE, they have been joined by a hyphen to form a new word whose spelling is different from either of these words.

51 Is the following statement true or false? 90-DAY-ACCOUNT is an illegal name for a data item because the first character of a name must be a letter.

●●●

False. The rule states that names of data items must contain at least one letter, but does not specify that the first character must be a letter. 90-DAY-ACCOUNT is a perfectly legal name.

52 Check your understanding of the rules for programmer-supplied names. For each name below, decide what rule, if any, is being violated.

1. 5
2. SYSTEM/360
3. OVERFLOW
4. ECONOMIC-ORDER-QUANTITY-COMPUTATION
5. ENTRY-PROCESS
6. HEADING LINE
7. F.I.C.A.

●●●

1. O.K. (acceptable for procedures, though not for data items or conditions).
2. Contains an illegal special character (SYSTEM-360 would be correct).
3. Not allowed because OVERFLOW is a reserved word.
4. Exceeds 30 characters.
5. O.K.
6. No spaces are allowed.
7. Contains illegal special characters (FICA or F-I-C-A) would be all right, but F-I-C-A- would violate still another rule).

53 *You can see that the rules for programmer-supplied names are not especially restrictive. They permit programmers to invent just about any name under the sun. In fact, they give the programmer so much freedom to invent names, that he must learn to hold back his inventive genius. Remember that one of the aims of COBOL is to produce programs that read like English, which mean that the programmer should either use English words for names (SALARY, DEDUCTIONS, etc.), or compose names from two or more English words, connecting them with hyphens (SOCIAL-SECURITY, FEDERAL-INCOME-TAX, etc.). Since names can be 30 characters long, there is no need to abbreviate or to make up code names.*

●●●

54 But suppose that a programmer decides to use a name like ZQX-3. First, is ZQX-3 a legal name (does it violate any rules)? Second, is it a good name?

●●●

ZQX-3 is a legal name, inasmuch as it does not violate any rules.

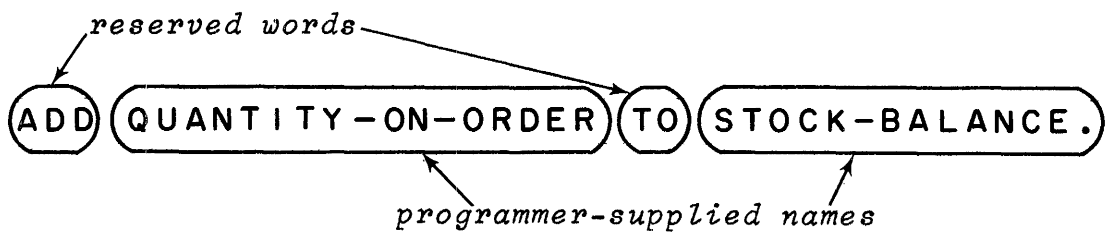
The answer to the second question is a matter of opinion. I hope you agree that meaningless names have no place in a COBOL program because they defeat the purpose of making programs easy to read and understand. By this standard, ZQX-3 is a good name only if it is readily understood by everyone who has to read the program. It is a bad name if it makes sense only to the programmer.

For instance, if the program has something to do with the manufacture of self-starting charcoal briquets, and ZQX-3 is the secret new ingredient that makes light fluid obsolete, then ZQX-3 is the name to use. But if the programmer decided to use ZQX-3 in place of MINIMUM-BALANCE because it is shorter, or because he once met a cab driver named Zarathustra Q. Xerxes III, then it is a bad name.

55 Reserved words and programmer-supplied names together account for nearly all of every COBOL program. The reserved words may be thought of as forming the skeleton, while the programmer-supplied names are the meat of most program entries. To get the idea, pick out the reserved words (with the aid of the reserved word list) and the programmer-supplied names in the entry below.

```
    ADD QUANTITY-ON-ORDER TO STOCK-BALANCE.
```

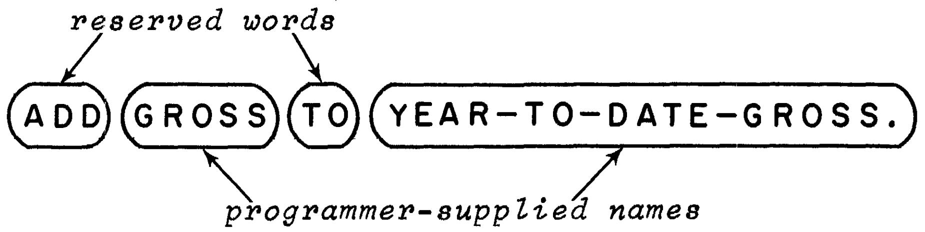
●●●



56 Pick out the reserved words and programmer-supplied names in the entry below. Notice that this entry has the same framework of reserved words as the entry in the preceding frame.

```
    ADD GROSS TO YEAR-TO-DATE-GROSS.
```

●●●



57 Here is a slightly longer program entry, with a considerably different ratio of reserved words to programmer-supplied names. Using the reserved word list, find out which words in the entry are reserved words, and which are programmer-supplied names.

```
FD  PAYROLL-FILE
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS EMPLOYEE-MASTER.
```

●●●

<u>Reserved words</u>	<u>Programmer-supplied names</u>
FD	PAYROLL-NAME
LABEL	EMPLOYEE-MASTER
RECORDS	
ARE	
STANDARD	
DATA	
RECORD	
IS	

In this example, the programmer has supplied the names of a data file and a data record. You have read that names for data items, data conditions, and procedures are supplied by programmers; for the moment, you can think of the file and the record as types of data items.

58 You can see that two of the main tasks involved in COBOL programming are to select the reserved words required for the framework of entries, and to supply names to fill in the framework -- especially names for data items. You know that it is up to the programmers to define the names that he uses.

You also know that there is an exception to this general rule; namely, that there are a few predefined data items with reserved names, which are called _____.

●●●

figurative constants

59 The most commonly used figurative constants are _____.

●●●

ZERO (ZEROS, ZEROES) and SPACE (SPACES)

60 *In the sample entries that you have seen, you have undoubtedly noticed some symbols -- specifically, periods and commas. These are only two of a large number of symbols that play an important role in COBOL programs. The typical procedure entries below illustrate a few more of the symbols used in COBOL.*

```
OPEN INPUT, MASTER, CHANGES; OUTPUT,
    NEW-MASTER.
```

```
COMPUTE AVERAGE-COST = ((TOTAL-UNITS -
    NEW-UNITS) * OLD-AVERAGE-COST +
    NEW-UNITS * NEW-COST) / TOTAL-UNITS.
```

```
IF MASTER-NUMBER > TRANSACTION-NUMBER,
    GO TO WRITE-MASTER.
```

Learn what groups the symbols are divided into, and get a general idea of what symbols are in each group. Look for symbols that appear in more than one of the groups.

Reading assignment: SYMBOLS
Punctuation symbols
Arithmetic symbols
Condition symbols

61 In the reference handbook, symbols are defined as "special characters which, _____, have particular meanings for the compiler".

●●●

individually

62 Some of the special characters which are used as symbols also appear in other elements, but there they are not used individually, and there also, they have different meanings. For example, a hyphen and a minus sign are the same character; however, they are used differently and obviously have different meanings. A hyphen is found embedded between other characters in reserved words and programmer-supplied names, where it has no special meaning, and serves only to improve readability. A minus sign stands apart in an arithmetic formula, with spaces before and after it,and signifies subtraction.

Accordingly, the correct way to read the formula for computing CURVE in the entry below is

$$\left\{\begin{array}{l}\text{RANGE minus ARCH minus RADIUS}\\ \text{RANGE-ARCH minus RADIUS}\\ \text{RANGE minus ARCH-RADIUS}\end{array}\right\}.$$

```
    COMPUTE CURVE = RANGE - ARCH-RADIUS.
```

●●●

RANGE minus ARCH-RADIUS

63 COBOL symbols fall into three groups: _____ symbols, _____ symbols, and _____ symbols.

●●●

punctuation, arithmetic, condition

64 Punctuation symbols make program entries more readable. In most cases, the use of punctuation marks is left up to the programmer. At his discretion, he may use commas to separate a series of operands or clauses. Or he may use semicolons to separate a series of clauses, in place of commas.

Certain punctuation is mandatory, though. A symbol that is required in every program entry is the _____ that terminates the entry.

●●●

period

In this course, the term "entry" is used in the precise sense of a series of language elements, the last of which is a period. That is to say, an "entry" will not be spoken of as just something entered on a coding sheet. This is discussed further in a later lesson.

65 Arithmetic symbols allow the COBOL programmer to write formulas in mathematical notation. There are, then, two general ways of doing arithmetic in COBOL: narratives and formulas. The difference between the two can easily be shown by an example; these two entries mean the same in COBOL:

```
    ADD REGULAR, OVERTIME GIVING GROSS.

    COMPUTE GROSS = REGULAR + OVERTIME.
```

In this example, the entries are exactly the same length, but you might guess that, in most cases,

$\left\{\begin{array}{l}\text{a narrative is shorter than the equivalent formula}\\ \text{a formula is shorter than the equivalent narrative}\end{array}\right\}$.

●●●

a formula is shorter than the equivalent narrative

66 Like arithmetic symbols, condition symbols are shorthand equivalents of written narratives. The two entries below have identical meanings.

```
    IF MASTER-NUMBER < DETAIL-NUMBER,
           WRITE MASTER-RECORD.
```

```
    IF MASTER-NUMBER IS LESS THAN
           DETAIL-NUMBER, WRITE MASTER-RECORD.
```

This example shows that the symbol < is equivalent to what reserved words?

●●●

IS LESS THAN

67 Certain symbols appear in more than one group of symbols. For instance, the equal sign is both an arithmetic symbol and a condition symbol. In the first entry below, the equal sign is used as a condition symbol; here it calls for two items to be compared to see if they are equal. In the second entry, it is used as an arithmetic symbol; there it causes the data named at the left of the equal sign to be made equal to the result of the computation -- that is, it calls for the result to be moved into that data item.

```
    IF BALANCE = MINIMUM, GO TO WARNING.
```

```
    COMPUTE AMOUNT = QUANTITY * PRICE
```

Which statement best sums up this idea?

An equal sign never has exactly the same meaning twice in succession.
No matter what they mean, all equal signs look alike to the untrained eye.
Equal signs are unpredictable, but the compiler can usually figure them out.
Equal signs in arithmetic and conditional entries have different meanings.
All equal signs are created equal, but some are more equal than others.

●●●

Equal signs in arithmetic and conditional entries have different meanings.

LESSON 3

68 *By this time, you should feel quite at ease with the various sample entries you have been looking at. You should be able to identify the reserved words and the programmer-supplied names in any entry, and you should be able to explain the difference between those elements. You also have a general idea of the kinds of symbols that you will run across in COBOL programs.*

In this lesson, we will discuss the three remaining elements -- literals, level numbers, and pictures. In contrast with the first three elements you have studied, these elements have comparatively specialized purposes, and do not occur as frequently in a program. However, it is every bit as important for you to know how these elements are used, and to be able to recognize them in program entries.

You will study literals first. Learn what a literal is, and what the two main types of literals are. Determine how you can tell the difference between the two types.

Reading assignment: LITERALS
Rules governing numeric literals
An example of a numeric literal in an entry
Rules governing non-numeric literals
An example of a non-numeric literal in an entry

●●●

69 A literal is an actual _____ used in a program.

●●●

value

70 The two main types of literals are _____.

●●●

numeric and non-numeric

System/360 COBOL permits the use of a third type of literal, the floating-point literal, which we will not deal with in this book.

71 Pick out the literal in the entry below. What type of literal is it?

```
    ADD 1 TO NUMBER-OF-RECORDS.
```

●●●

The digit 1 in this entry is a numeric literal.

72 Which characters are not permitted in numeric literals?

●●●

Letters and most special characters are not permitted. The only special characters permitted are a plus or minus sign as the leftmost character, and a decimal point as any character except the rightmost.

73 The correct way to write "minus one-half" as a numeric literal is

$\left\{\begin{array}{l}-1/2\\ -.05\\ .-5\end{array}\right\}$.

●●●

-.5

74 One of the rules for numeric literals is that whole numbers are written without decimal points. A decimal point may not be the rightmost character of the literal. The entry below appears to violate this rule, yet it is a correct entry. Can you explain this discrepancy?

```
    COMPUTE LOAD = WEIGHT * 30.
```

●●●

The last character is not a decimal point; it is the <u>period</u> that ends the entry. This becomes clear when the entry is rewritten as COMPUTE LOAD = 30 * WEIGHT.

This is another example of a character having different meanings in different contexts. However, this should not seem unreasonable to you, inasmuch as the same character is used in the same two ways in everyday English.

75 Non-numeric literals are easily recognized, because they are enclosed in _____.

●●●

quotation marks

Notice that we use the single quotation mark (') and not the double quotation mark ("). Single and double quotation marks are distinctly different special characters, with different data codes in System/360. Non-numeric literals in System/360 COBOL must be enclosed by single quotation marks, one before and one after the literal.

76 Don't be misled by the name "non-numeric". Non-numeric literals may contain digits, and they may be numbers. Which of the literals in this list are valid non-numeric literals?

```
'JANUARY, 1966'
'NOT IN FILE'
'50'
'1'
```

●●●

These are all valid non-numeric literals.

77 Suppose that all of the items in the list below are literals. Which are numeric literals and which are non-numeric literals?

```
50
30565
'-5.03'
3.1416
'TOTAL'
```

●●●

The numeric literals are 50, 30565, and 3.1416. The non-numeric literals are '-5.03' and 'TOTAL'.

I have reprinted the quotation marks here because they are essential to the identification of the type of literal. Remember, though, that the quotation marks merely enclose a non-numeric literal, and are not part of the literal.

78 Which character is not permitted in non-numeric literals?

●●●

The quotation mark is not permitted, because it signifies "end of literal" to the compiler.

79 In non-numeric literals, you can write anything you want between the quotation marks, as long as you don't throw in another quotation mark. You might, for example, use a non-numeric literal for the message 'NUMERIC SECURITY CONTROL IS OMITTED', even though every word in this message is a _____ word.

●●●

reserved

80 Can the last character of a non-numeric literal be a decimal point, for example, '$4500.'?

●●●

Yes

81 *From time to time, I have mentioned that programmers must define the names that they use in programs. To define the name of a data item, the programmer merely includes the name in an item description entry. Level numbers and pictures are also found in these item description entries. The subject of item description entries will be explored in depth in a later lesson, but we will touch on it briefly at this point. The thing for you to learn now is how to recognize level numbers and pictures in a program.*

Reading assignment: LEVEL NUMBERS
  Examples of level numbers in entries
PICTURES
  Examples of pictures in entries

●●●

82 A number that designates the level of a data item, in relation to other data items, is called a _____.

●●●

level number

83 Level numbers are found in entries that assign _____ to data items and data values.

●●●

names

84 The numbers 01 through 49, 77, and 88 may be used as level numbers. Level numbers 01 through 49 are used for data items that form _____.

●●●

records

85 Level number 77 is used for items that are neither records nor part of records. Such items are called _____.

●●●

independent items

86 Level number 88 is used to assign names to _____ that data items may assume.

●●●

values

87 There are three elements in this entry. The last element, as always, is a punctuation symbol -- the period that indicates the end of an entry. The first element, 01, is a _____, while PARTS-CATALOG-ITEM is a _____.

```
01  PARTS-CATALOG-ITEM.
```

●●●

level number
programmer-supplied name

88 As in every entry that contains a level number, the level number is the _____ element in this entry.

```
    02  ACCOUNT-NUMBER, PICTURE 9(6).
```

●●●

first

89 A string of characters with special meanings that describes certain characteristics of a data item is called a _____.

●●●

picture

90 A picture is easy to recognize in a program because it is preceded by the reserved word _____.

●●●

PICTURE (or PICTURE IS)

91 The picture in this entry is $\left\{\begin{array}{l}\text{PICTURE 99}\\\text{PICTURE}\\\text{99}\end{array}\right\}$.

```
        03  MONTH   PICTURE 99.
```

●●●

99

The reserved word PICTURE merely signals that 99 is a picture, just as on a sign that reads "SPEED LIMIT 50", 50 is the speed limit and the words that precede it are there merely to distinguish the number from other numbers on signs along the road. The two elements PICTURE 99 together are called a "picture clause". The clause might also have been written as PICTURE IS 99.

92 Pictures present us with the interesting situation of sometimes having an element within an element, since pictures may contain numeric literals. These literals must be unsigned whole numbers, and are easy to spot because they must be enclosed in _____.

●●●

parentheses

93 The picture in the entry below is the abbreviated way of writing 56 Xs. Aside from the fact that a programmer would be out of his mind if he didn't write X(56) instead of 56 Xs, it would be <u>illegal</u> to write a string of 56 Xs. Why?

```
    02  FILLER, PICTURE X(56).
```

●●●

A picture must not be over 30 characters long.

The picture X(56) is only five characters long, even though it represents 56 Xs.

94 *The literals help to make the picture more readable. You see, one of the things that a person might want to know when he is checking out, debugging, or analyzing a program, is just how often a picture character of a certain kind is repeated. He might miscount the repetitions if the picture were 9999999999, while 9(10) practically eliminates the possibility of a mistake.*

The number in parentheses indicates the number of times the picture character occurs without interruption by some other character. Thus, the correct abbreviation of S999999V99999 is

$$\left\{\begin{array}{l}\text{SV9(11)}\\ \text{S9(11)V}\\ \text{9(11)SV}\\ \text{S9(6)V9(5)}\end{array}\right\}.$$

●●●

S9(6)V9(5) (The character V interrupts to form two strings of 9s.)

95 Pictures are found only in entries that begin with level numbers, as the first entry below illustrates. However, as the second example illustrates, an entry with a level number

$$\left\{\begin{array}{l}\text{does not necessarily contain a picture}\\ \text{must always be written with a picture}\\ \text{is not complete if it lacks a picture}\end{array}\right\}.$$

```
    02  REDUCTION-FACTOR, PICTURE IS 9(18),
        USAGE IS COMPUTATIONAL.
```

```
    02  PURCHASE-DATE, COMPUTATIONAL-3.
```

●●●

does not necessarily contain a picture

96 *You will learn much more about pictures in a later lesson, so don't be concerned about not knowing what the Xs and 9s and other picture characters actually mean. However, you should now be aware that pictures appear in some item description entries, and that they play a role in the process of defining data items. And you should be able to recognize a picture when you see one.*

●●●

97 You have now studied all six COBOL language elements. You will find these elements in all COBOL entries, in all COBOL programs. Knowledge of the elements has little value in and of itself, though -- unless you can recognize each element when you see it in a program.

To make the task a little harder, some of the elements may look alike. For instance, the number 99 could be a [reserved word] [programmer-supplied name] [literal] [level number] [picture].

●●●

programmer-supplied name OR literal OR picture

The number 99 might be the name of a procedure. It is also a valid literal or picture. It is not a reserved word, since it does not appear on the reserved word list. And it is not a level number, because it is not a number between 01 and 49; 77; or 88.

98 The meaning of a number in a COBOL program is determined by the context in which it is used. It is possible for the same number to be used in several different ways within one program.

Applying this general rule to a specific case, you can conclude that the number 25

{ cannot be used as a literal because it is reserved for use as a level number
can be used as a literal even though it is also used as a level number
can be used as a literal only if it has not previously been used as a level number }

●●●

can be used as a literal even though it is also used as a level number

99 Often it is fairly easy to identify an element. For example, the number 999 in the entry below must be a _____ because _____.

```
       03  DISCOUNT-CODE, PICTURE 999.
```

●●●

picture, because it is preceded by the word PICTURE

100 The task isn't always as easy as it was in the previous example, but it is usually possible to make a positive identification based on the rules that you are already familiar with. See how well you do in the following cases.

In this entry, 15 is a _____ because _____.

```
       15  DATE-OF-LAST-PAYMENT, PICTURE 9(6).
```

●●●

level number because it is the first element of the entry and is followed by a programmer-supplied name

101 In this entry, 65 is a _____ because _____.

```
77  UPPER-LIMIT, PICTURE 99, VALUE '65'.
```

●●●

non-numeric literal because it is enclosed in quotation marks

102 *The rules for identifying pictures, level numbers, non-numeric literals, and reserved words are fairly simple. And you can generally identify other elements by a process of elimination. If, for instance, you come across a name that contains letters, and it is neither a reserved word nor a non-numeric literal, then it must be a programmer-supplied name. Similarly, if you find a number that is not a level number, not a picture, and not a non-numeric literal, then it must be either a numeric literal or the programmer-supplied name of a procedure -- and the context in which you find it will generally tell you whether the number is a literal or a procedure name.*

●●●

103 In the entry below, you can determine that FD is definitely a _____. Also, you can figure out that 32 must be a _____.

```
FD  INVENTORY-FILE
    BLOCK CONTAINS 32 RECORDS;
    LABEL RECORDS ARE STANDARD;
    DATA RECORD IS INVENTORY-RECORD.
```

●●●

reserved word; numeric literal

You should have identified FD easily and positively with the aid of the reserved word list. And you should have figured out that 32 is a literal from the sense of the entry, which says that a block in the inventory file actually -- literally -- contains 32 records.

In certain entries, such as the one in this frame, literals are required to be whole numbers, and to have no sign. You have already seen such literals in pictures. You will find that these unsigned whole numbers are called "integers" in COBOL manuals. For the present time, the important thing for you to realize is that the values of these numbers are taken literally by the compiler, so we will call them all "literals".

104 One of the number in this entry is a procedure name, and one is a literal. Which is which?

```
    IF AREA-CODE = 607, GO TO 582.
```

●●●

The procedure name is 582. The literal is 607.

Notice that the name 582 reveals nothing about the processing done in that procedure. This entry would have meant more to the reader: IF AREA-CODE = 607, GO TO COMPUTE-TOLL. This illustrates why English words are usually preferred to numbers so far as names are concerned.

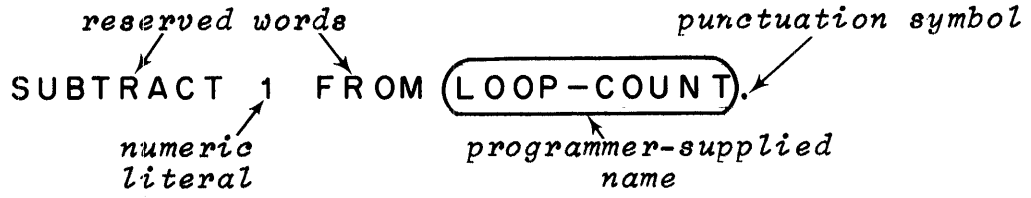
105 *In the next four frames, identify all of the elements in each entry.*

●●●

106

```
    SUBTRACT 1 FROM LOOP-COUNT.
```

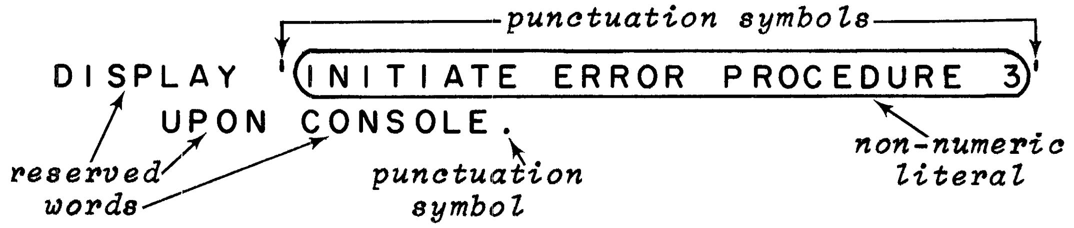
●●●



107

```
    DISPLAY 'INITIATE ERROR PROCEDURE 3'
        UPON CONSOLE.
```

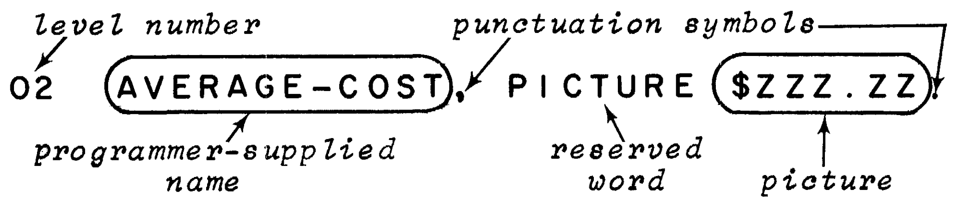
●●●



108

```
    02  AVERAGE-COST, PICTURE $ZZZ.ZZ.
```

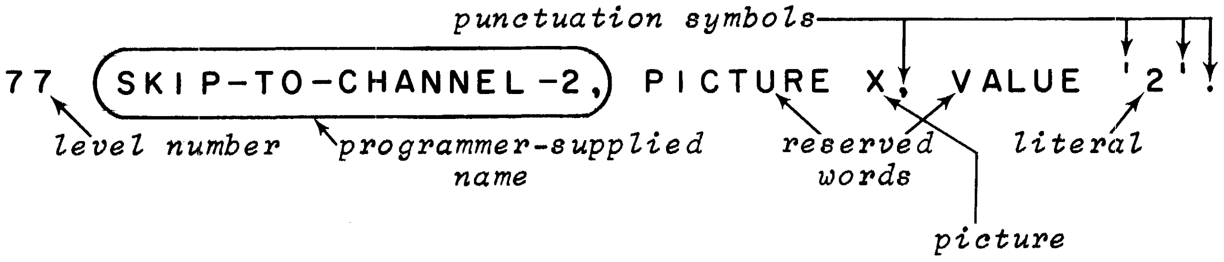
●●●



109

```
77  SKIP-TO-CHANNEL-2, PICTURE X, VALUE '2'.
```

●●●



110 *The information you have learned about the six COBOL elements is, of course, not an end in itself. Rather, it provides you with a foundation of terminology, and with the ability to recognize the elements in programs. The elements will come up again and again in future lessons; whenever a reference is made to an element, and you are not quite sure of the facts about it, make a point of looking back at the reference handbook. By repeatedly looking up facts when you need them, you will solidify your knowledge of them; in particular, you will see much more clearly how new facts relate to facts you have previously studied.*

●●●

111 *Let's quickly review some of the material you have been studying.*

1. Which elements are selected by the programmer from fixed sets?

2. Which elements are composed by the programmer?

●●●

1. reserved words, symbols, level numbers
2. programmer-supplied names, literals, pictures

112 One of the major restrictions imposed on elements composed by the programmer is their length. See if you can recall the maximum length of each of these elements:

1. programmer-supplied names
2. non-numeric literals
3. numeric literals
4. pictures

●●●

1. 30 characters
2. 120 characters
3. 18 digits
4. 30 characters

LESSON 4

113 *In lesson 1, you learned that standardized programming is one of the aims of the COBOL system. To this end, the system provides standard language elements, standard entry formats, and a standard program structure. Lessons 2 and 3 concerned themselves with the COBOL language elements. In this lesson, we will turn to the COBOL program structure. As in the preceding lessons, the emphasis will really be on mastering the vocabulary of COBOL, in preparation for further work with the language. This time, you will learn terms such as "division" and "section". And as before, I hope that you will not waste your time and energy memorizing the definitions of these terms. Rather, your efforts should be directed toward learning to apply the definitions -- to recognize a division in a program, for instance, and to perceive the relationship between divisions and sections.*

Find out how many divisions there are, their names, and their sequence. Determine how you would recognize the beginning of a division, section, and paragraph. Learn what distinguishes an entry.

Reading assigment: PROGRAM STRUCTURE
- Divisions
- Sections
- Paragraphs
- Entries
- Sample division with structural units identified

●●●

114 How many divisions are there in a COBOL program?

●●●

four

115 The four divisions are Data, Environment, Identification, and Procedure, but this is not their correct order. What is their correct order?

●●●

Identification, Environment, Data, Procedure

116 The entries that comprise a COBOL program are divided into four divisions according to the purpose served by the entries. This allows the programmer to give his full attention to one division -- one set of entries -- at a time, instead of having to worry about the whole program at once.

Since every program is divided into the same divisions, which always appear in the same order, it is easy for someone other than the author of the program to pick it up and know where to find whatever he is looking for.

You can conclude that the standardized program structure is [an aid to program writers] [an aid to program readers].

●●●

an aid to program writers AND an aid to program readers

117 Name the divisions of a COBOL program, in order.

●●●

Identification, Environment, Data, Procedure

118 As part of your reading assignment, you examined a sample division. Which division was it?

●●●

Environment division (The first entry of the division gives the division name.)

119 The beginning of a division is marked by a division header entry. What marks the end of a division?

●●●

There is no special "end of division" entry. You know that the preceding division has ended when you come to the next division header entry.

120 Every division header entry contains the word _____, and every section header entry contains the word _____.

●●●

DIVISION; SECTION

121 *Note that all header entries begin to the left of the broken vertical line on the program sheet, while all other entries are written to the right of the broken line. The rules that govern this are explained in detail in the next course in this series. At this time, it will suffice to say that the headers are brought out a little to the left for emphasis.*

You also note that each header, like every other entry, is terminated by a _____.

●●●

period

There is no "official" definition for "entry" in COBOL, but the definition given in the reference handbook does not contradict anything you may later read in other COBOL publications. Most COBOL manuals refer to some entries as "entries", to others as "sentences", and have no term at all for still others. Throughout this course, however, we will use the term "entry" in one, precise way.

122 Which of these is a valid paragraph header entry:

$$\left\{\begin{array}{l}\text{DATE-WRITTEN}\\ \text{SEQUENCE-CHECK-PROCEDURE.}\\ \text{FILE-CONTROL PARAGRAPH}\\ \text{MINOR-TOTAL PARAGRAPH}\end{array}\right\}.$$

●●●

SEQUENCE-CHECK-PROCEDURE.

DATE-WRITTEN is an acceptable paragraph name, but is not a valid header entry because the period is missing. (If looking for periods seems to you like picking nits, let me assure you that one of the hallmarks of an effective COBOL programmer is his meticulous attention to the smallest details.)

123 Inasmuch as periods are so important to the compiler, it behooves us to be able to tell the difference between a period and a decimal point in an entry. A period is always followed by a space; a decimal point is never followed by a space.

This distinction explains why a decimal point must never be written as the rightmost character of a numeric literal. How would the entry below appear to the compiler if the literal were written with a decimal point (66365.)?

```
    MOVE 66365 TO CONTROL-DATE.
```

●●●

The entry would appear as two entries, MOVE 66365. and TO CONTROL-DATE. Neither of these "entries" would make sense to the compiler, of course, and both would be regarded as being incomplete.

124 The correctly written section header entry is

{INPUT/OUTPUT SECTION.
INPUT-OUTPUT SECTION.
INPUT-OUTPUT-SECTION.}

●●●

INPUT-OUTPUT SECTION.

125 Every section begins with a section header entry. There is no "section trailer entry", but you will know that you have passed the end of a section when you come to the next

[division header entry]
[section header entry]
[paragraph header entry].

●●●

division header entry OR section header entry

126 One of the divisions is too small and too simple to require sectionalizing. This division, which never contains sections, is the _____ division.

●●●

Identification

127 Although he is not required to do so, the programmer may, at his own discretion, choose to establish sections in the _____ division of a program.

●●●

Procedure

128 The Data division contains sections, but not paragraphs. In place of paragraphs, the division contains entries arranged in a form that somewhat resembles an outline. Most of the entries in the Data division begin with numbers like 01, 02, etc. These numbers are definitely not paragraph headers; rather, as you already know, they are _____.

●●●

level numbers

129 The only division that always contains both paragraphs and sections is the _____ division.

●●●

Environment

130 The division that always contains paragraphs, and sometimes contains sections, is the _____ division.

●●●

Procedure

131 There are always sections, but never paragraphs, in the _____ division.

●●●

Data

132 There are always paragraphs, but never sections, in the _____ division.

●●●

Identification

133 *It is necessary that you get a good idea of how divisions, sections, and paragraphs appear in an actual program. For this purpose, turn to the CASE STUDY program at the end of the reference handbook, and locate the four divisions. The next seven frames all refer to that program.*

●●●

134 The longest division in the case study program is the _____ division.

●●●

Data

135 How many paragraphs are in the Identification division?

●●●

four

136 How many paragraphs are in the Environment division?

●●●

three (two in the Configuration section, and one in the File-Control section)

137 How many paragraphs are in the Data division?

●●●

none (The Data division never contains paragraphs.)

138 What are the names of the sections of the Data division?

●●●

File section and Working-Storage section

139 How many sections are in the Procedure division?

●●●

none

140 How many paragraphs are in the Procedure division?

●●●

eight

141 *So far we have been looking at COBOL programs in bits and pieces, to become conversant with the elements that are found in programs, and with the structure of programs. From this point on, we are going to start paying more attention to the content of COBOL programs, and the meaning of various entries. In your next reading assignment, you will take an overall look at the four divisions, to see what kinds of information each one contains.*

You should learn to describe the kinds of information found in the divisions, to name the division in which you would look for specified information, and to discuss how the divisions are related to each other.

Reading assignment: PROGRAM CONTENTS
  Identification division
  Environment division
  Data division
  Procedure division

●●●

142 Their names are excellent clues to the kind of information in the divisions. So if you remember the names of the divisions, the rest is almost too easy.

As its name indicates, the Identification division _____ the program. The Environment division describes the machine _____ for compiling and executing the program. The Data division describes the _____ to be processed. And the Procedure division tells the _____ to be followed in processing the data.

●●●

identifies; environment; data; procedures

143 While each division contains information of a different kind, all of the divisions have certain functions in common. For example, every division is intended, to a large extent, to inform people who read the program. However, the division that exists almost solely for that purpose is the _____ division.

●●●

Identification

144 The name of the game is "data processing", so all of the divisions have something to do with data. Only one division is called the Data division, though, and in it you would see the

[assignment of data files to input-output devices]
[breakdown of data files into records]
[reading and writing of data files].

●●●

ONLY the breakdown of data files into records

145 In which division would you look to find out:

How the output records are organized?
What actions will follow the processing of the last input record?
What input-output devices are used in the program?
What the name of the program is?

●●●

Organization of records:	Data division
Actions:	Procedure division
Input-output devices:	Environment division
Program name:	Identification division

146 While the divisions work together as a team, each division is independent. That is, each has a separate and distinct role to play. For example, the Data division describes data files and records, but does not tell what input-output device will read or write the data. Assigning files to input-output devices is a job of the _____ division.

●●●

Environment

147 Likewise, the Procedure division uses the names of data items, but does not describe the items. If you wanted to know how many characters a data item contains, or where the item appears within a data record, you would look in the _____ division.

●●●

Data

148 *Let's get more specific, by taking a closer look at each division. Remember that we are not concerned with the rules for composing program entries, even though -- unavoidably -- some of the rules will pop up in our discussion. You are not expected to be able to write a program at the conclusion of this course; only to be able to read a program, and comprehend what you read. The rules for program entries have some bearing on this, for they help you to realize the ways in which other programs may be different from the sample programs.*

Reading assignment: IDENTIFICATION DIVISION ENTRIES
Sample Identification division

●●●

149 Although the sample Identification division given in the reference handbook occupies several lines, only a small part of it is actually required by COBOL rules. Determine which part is required.

●●●

```
IDENTIFICATION DIVISION.

PROGRAM-ID.
    'PURCHASE'.
```

All of the other entries are optional, but very useful. They all fall into the category of "program documentation", and are well worth the few extra moments it takes to write them.

150 Seven fixed paragraph names are mentioned in the reference handbook. Each name is a reserved word. The reference handbook makes it clear that

the programmer must write a Program-Id paragraph and no more than six other paragraphs, for which he can invent names or use fixed names.
no more than seven paragraphs may be written, and their names must be chosen from among the seven fixed names.
the programmer is allowed to write more than seven paragraphs, provided that he uses reserved words for the names of the paragraphs.

●●●

no more than seven paragraphs may be written, and their names must be chosen from among the seven fixed names.

One paragraph, the Program-Id paragraph is required. The programmer is not allowed to invent names for the other paragraphs. Nor is he allowed to use any reserved words for names of paragraphs in the Identification division, other than the seven fixed names that are provided.

151 How many more paragraphs might the sample division have had?

●●●

two (Date-Compiled and Security)

152 The program name, which is required to appear in the Program-Id paragraph, is the name by which the program will be identified on a job control card at the time that the object program is executed. Program name is an example of a special kind of non-numeric literal called an "external name". Like all non-numeric literals, external names are enclosed in quotation marks. Unlike other non-numeric literals (which can contain as many as 120 characters), external names are limited to eight characters -- a letter followed by up to seven letters and/or digits.

●●●

153 *There is not much more that can be said about the Identification division, so we will go on to the Environment division. Here our sample division involves a card input file and a printed output file; however, the same sorts of entries are used for all sequentially organized files, whether they are on tape, disk, card, or printer devices. Additional entries, which we will not deal with in this course, are used for non-sequentially organized data files, on disk or other direct-access devices.*

Reading assignment: ENVIRONMENT DIVISION ENTRIES
Sample Environment division
System/360 model numbers
Assignment of files to input-output devices

●●●

154 *Knowing how to decipher the model number and storage capacity codes in the Source-Computer and Object-Computer paragraphs is a minor, but useful bit of information. The actual model number is given, so only the storage capacity code represents a problem. An easily solved problem, however, since you can look up the meanings of the code letters whenever you need them.*

Use the table of code letters in the reference handbook to interpret the sample Environment division. Both the source computer and the object computer are IBM System/360 Model _____, with a storage capacity of _____ bytes.

●●●

30; 65,536

The precise number of bytes in storage is rarely useful to anyone. It is sufficient to know the approximate number. So, we generally talk in terms of round numbers; for instance, the letter E can be said to stand for 32,000 (32K) bytes of storage. These code letters are used in many references to System/360 -- not merely in System/360 COBOL. Reference manuals, for example, may speak of "F compilers", by which they mean compilers that require a system with 64K bytes of storage. One way to remember the approximate values of the code letters is to remember that the table starts with C, which stands for 8,000 (8K). Then just double the number of bytes for each succeeding letter; D is 16K, E is 32K, F is 64K, and so on. By this method, you will be off by several thousand bytes by the time you get to I, but at least you will know what ball park you are playing in.

155 Turn to the Environment division of the case study program in your reference handbook. What is the source computer for that program?

●●●

An IBM System/360 Model 50 with 128K (actually 131,072) bytes of storage

156 The File-Control paragraph of the Environment division is important because in it every input or output _____ is assigned to an input or output _____.

●●●

file (data file); device

157 Each entry in the File-Control paragraph begins with the word SELECT, followed immediately by the file name. The file name is

- the name by which that file will be identified on a job control card.
- a name which indicates the type of input-output device used for that file.
- the name which that file is called in the Data and Procedure divisions.
- a name which has been filed with the Program Registrar in Washington.

●●●

the name which that file is called in the Data and Procedure divisions

158 The external name that appears in each SELECT entry is the job-control-card name for

- the file
- the device to which the file is assigned

●●●

the file

You encountered another external name earlier -- the program name given in the Program-Id paragraph of the Identification division.

159 The file name and the external name are, then, two names for the same data file. It is permissible for these names to be the same; however, external names are limited to eight characters, whereas file names (like all programmer-supplied names) may be up to _____ characters long.

●●●

30

160 The SELECT entry below states that a file known as _____ within the program will be called _____ on a job control card.

```
    SELECT PARTS-EXPLOSION-FILE
        ASSIGN TO 'PARTS'
        UTILITY 2400 UNITS.
```

●●●

PARTS-EXPLOSION-FILE; PARTS

161 The assigning of a file to a device consists of naming a device class and a device number. Which statement below gives the correct rule regarding device classes and numbers?

- Both device class and device number must be specified for every file.
- Device class must be given for every file; number is sometimes omitted.
- Device number is required for every file; device class may be omitted.
- Device class and device number are optional; both may be omitted.

●●●

Device class must be given for every file; number is sometimes omitted.

Device number is sometimes omitted in order to make the program "device independent". This means that an object program may be compiled without specifying exactly which input-output devices are to be used. The choice of devices can then be made when the object program is executed.

162 What are the three device classes?

●●●

UTILITY, DIRECT-ACCESS, and UNIT-RECORD

163 Magnetic disk devices are included in

[UTILITY]
[DIRECT-ACCESS]
[UNIT-RECORD].

●●●

BOTH UTILITY AND DIRECT-ACCESS (because they can read and write data both sequentially and randomly)

164 In which device class or classes does each of the following devices belong?

1. IBM 2540 Card Read Punch
2. IBM 7320 Drum Storage
3. IBM 2403 Magnetic Tape Unit
4. IBM 2302 Disk Storage
5. IBM 1403 Printer

●●●

1. UNIT-RECORD
2. Both UTILITY and DIRECT-ACCESS
3. UTILITY
4. Both UTILITY and DIRECT-ACCESS
5. UNIT-RECORD

165 We have seen that there are three classes of devices; UTILITY, DIRECT-ACCESS, and UNIT-RECORD. But our main concern is with data files rather than the devices used for reading and writing. Files basically fall into two categories: sequential and non-sequential. In sequential files, records are read or written in the order in which they are physically stored. In non-sequential files, records are not read or written in the order in which they physically appear on the storage medium.

Sequential files can be assigned to all three device classes. Non-sequential files can be assigned only to DIRECT-ACCESS devices. This means that _____ devices can be used only for sequential files, while _____ devices can be used for both sequential and non-sequential files.

●●●

UTILITY and UNIT-RECORD devices can be used only for sequential files, while DIRECT-ACCESS devices can be used for both sequential and non-sequential files.

(Card files and printed files must be assigned to UNIT-RECORD; only these files may be assigned to this device class.)

166 The file named in the entry below is a $\left\{\begin{array}{l}\text{sequential}\\ \text{non-sequential}\end{array}\right\}$ file.

```
    SELECT DEMAND, ASSIGN 'DEMAND' UTILITY.
```

●●●

sequential (A file must be sequential if it is assigned to a UTILITY device.)

As it is used here, the term "sequential file" does not necessarily imply that the records in the file are "in sequence". It does mean that the records are read from the storage medium (for instance, magnetic tape) one after another, in the order in which they appear; or that the records are written in consecutive places on the storage medium. Note, however, that the records might be out of order as far as their control numbers are concerned; a simple example of this is a file of unsorted records on magnetic tape -- these records will be read sequentially by the sort program, but they are not in sequence.

167 SELECT entries $\left\{\begin{matrix}\text{do}\\\text{do not}\end{matrix}\right\}$ specify whether a file is to serve as input or as output.

●●●

do not

The file will be identified as input or output in the Procedure division entry that "opens" the file (makes it ready for reading or writing). This arrangement makes it possible for the COBOL programmer to use the same SELECT entry in at least a couple of programs -- the program that creates the file, and any program that uses that file as input.

168 Read the SELECT entry below, and then answer the following questions.

```
SELECT INVENTORY-MASTER
    ASSIGN TO 'INVMST'
    UTILITY 2400 UNITS.
```

1. Is the file a sequential file, or a non-sequential file? Why?
2. Is it an input file, or an output file? How can you tell?
3. What class of device is assigned to the file? What specific device?
4. What is the file's external name?
5. What name will be used in the Procedure division to refer to the file?

●●●

1. A sequential file, because it is assigned to a UTILITY device.
2. There is no way of telling from the SELECT entry whether this is an input or output file; it would be necessary to examine the Procedure division of the program to see how the file is processed.
3. UTILITY; specifically, magnetic tape units of the IBM 2400 series.
4. INVMST
5. INVENTORY-MASTER

169 Keeping in mind that there must be a SELECT entry for every file that is processed by the program, look at the Environment division of the case study program in the reference handbook.

1. How many files are processed by the program?
2. What are the file names?
3. No device number is given for one of the files; what kinds of input-output devices might be used for this file?
4. For each file, determine whether it is sequential or non-sequential.

●●●

1. two
2. BILLING-FILE, CUSTOMER-BILL-FARE
3. Magnetic tape, disk, drum, or data cell devices might be used for the BILLING-FILE.
4. Both files are sequential.

170 *A final word about sequential and non-sequential files: This course is restricted to COBOL programs that process sequential files. This has been done because the logic of sequential input, processing, and output is familiar to even the beginning programming student -- which makes it possible for us to concentrate on COBOL instead of taking excursions into the worlds of systems design, data management, and so on, to explain what the COBOL program is trying to accomplish.*

Non-sequential file processing is not necessarily more difficult to program in COBOL than sequential file processing, but many people are unfamiliar with its underlying principles. Besides, there are different methods of non-sequential file organization: "indexed", "direct", and "relative" -- each with its own processing logic. So, we will steer clear of non-sequential files in this course, just to keep your introduction to COBOL from being complicated by too many other programming considerations.

●●●

LESSON 5

171 *The Data division of most COBOL programs is quite a bit longer than the Environment division or the Identification division. But don't get the idea that this division is harder to understand just because it is longer. The fact is that the Data division consists of many entries of similar types, repeated again and again; when you get the hang of one or two item description entries, for instance, you will be able to cope with most of the item description entries that you may find in any program -- and these account for the bulk of the Data division.*

As you examine the sample Data divisions in the reference handbook, it will be worth your while to read every entry individually, instead of merely glancing at the page. That will help you to grasp the pattern of entries.

Reading assignment: DATA DIVISION ENTRIES
Sample Data division
Another sample Data division, with entry types identified

●●●

172 File description entries are found in the _____ section.

●●●

File

173 File description entries are made up of

[record descriptions]
[item description entries]
[descriptions of independent items]

●●●

NONE of these

File description entries are clearly separate from other entries and groups of entries in the division.

174 Record descriptions may appear in the

[Working-Storage section]
[File section]

●●●

BOTH the Working-Storage section AND the File section

The second sample Data division in the reference handbook illustrates this.

175 Record descriptions are made up of

[file description entries]
[item description entries]
[descriptions of independent items]

●●●

item description entries

176 The descriptions of independent items are

$\left\{\begin{array}{l}\text{record descriptions}\\ \text{item description entries}\end{array}\right\}$

●●●

item description entries

177 Review the two previous frames. Is it logical to conclude that record descriptions are made up of descriptions of independent items?

●●●

No

Record descriptions and independent item descriptions are separate and distinct parts of the Data division. This is shown in the second sample division.

178 *We have been dealing with several terms in looking at the make-up of a typical Data division -- terms like "file", "record", "item", and "independent item". Each of these terms has a precise meaning that you will need to know before you can go further in your study of this division. Along with the definitions of the above terms, I have included definitions of other terms that you will be using very shortly, like "elementary item" and "group item". I don't want you to learn the definitions word for word; just make sure that you can explain to yourself the differences and similarities between the various terms.*

Reading assignment: SYSTEM/360 COBOL TERMS FOR UNITS OF DATA

●●●

179 An item is $\left\{\begin{array}{l}\text{a piece of data}\\ \text{a storage area that will hold a piece of data}\end{array}\right\}$.

●●●

a storage area that will hold a piece of data

This is a useful distinction to make. What the COBOL programmer actually accomplishes in the Data division is to reserve areas in which data will be stored while it is being processed. The data itself will change with each record that is put in or out -- except in the case of constants.

180 Suppose that an item named DATE is made up of three smaller items, MONTH, DAY, and YEAR. The three smaller items are <u>not</u> made up of still smaller items.

In COBOL terms, DATE is

$\left\{\begin{array}{l}\text{a group item}\\ \text{an elementary item}\end{array}\right\}$,

while MONTH, DAY, and YEAR are

$\left\{\begin{array}{l}\text{group items}\\ \text{elementary items}\end{array}\right\}$.

●●●

a group item; elementary items

181 A record is [a group item] [an elementary item] [not an item].

●●●

EITHER a group item OR an elementary item

A record is nearly always a group item, in that it consists of smaller items. However, the programmer may sometimes decide to define a record as an elementary item; that is, he may choose not to describe any of the smaller items that make up the record, and treat it as one big item.

182 Earlier we noted that descriptions of independent items are separate from record descriptions in the Data division. The reason for this should now be clearer to you, since an independent item, by definition, is any item that is [not a part of the record] [not a record].

●●●

not a part of a record AND not a record

183 Thus, there are three possible things that an item can be:

1. a record
2. a part of a record
3. an independent item

It is possible for an elementary item to fall into any one of these three categories. However, a group item can only be

[a record]
[a part of a record]
[an independent item].

●●●

EITHER a record OR a part of a record

184 Any group item can be made up of elementary items and/or other group items. For example, there might be an item called CATALOG-NUMBER which is made up of AVAILABILITY-CODE, SHIPPING-CODE, and WAREHOUSING-NUMBER. WAREHOUSING-NUMBER might be further subdivided into WAREHOUSE-LOCATION, MANUFACTURER, PRIORITY, and BIN-NUMBER. One or another of these subdivisions might be still further subdivided.

Sooner or later, however, we will arrive at the point where no item can be subdivided any further. That is, any group item can ultimately be reduced to a group of _____ items.

●●●

elementary

185 Perhaps this notion of items-within-items will be easier for you to see if we look at it from the opposite point of view. We can begin with a number of elementary items, and combine them into group items. The group items can then be combined into larger group items. When we have combined all of our items into one, all-inclusive group item, that item is called a _____.

●●●

record

186 By going one step further, and collecting an entire set of all similar records, we will create a

$\left\{\begin{array}{l}\text{block}\\ \text{file}\end{array}\right\}$.

●●●

file

187 In addition to data records, a file may include _____ records which contain information about the file.

●●●

label

Label records (often spoken of simply as "labels") are generally written at the beginning and end of a file. Labels are also written at the beginning and end of each physical volume, especially when a file occupies more than one volume; for instance, labels would be written at the beginning and end of each magnetic tape reel if the file occupied more than one reel.

188 When ten data records are written at one time by an output device, those ten records constitute a _____.

●●●

block

189 Which statement is correct?

$\left\{\begin{array}{l}\text{A record must contain more than one item.}\\\text{A block must contain more than one record.}\\\text{A group item must contain more than one elementary item.}\\\text{An elementary item must contain more than one independent item.}\end{array}\right\}$

●●●

A group item must contain more than one elementary item.

The fourth statement above is ridiculous, but the first two verge on being correct. However, a record may itself be an elementary item (although most records are group items); and a block may contain just one record. When it comes to blocks, there is a similar but conflicting term that you should watch out for: "blocked". Records are sometimes said to be "blocked" when two or more records are combined into one physical data unit. In COBOL, we do not use this term: instead, we have the term "block", which is used only as a noun, meaning a physical data unit that comprises one or more records.

190 *In your next reading assignment, which deals with file description entries ("FD entries"), you will encounter several of the terms we have just defined. You may, of course, look back at the definitions of terms whenever you need to. You will read many details about clauses of FD entries, but don't try to memorize these details. As you read about each clause, you should look for that clause in the sample FD entry, and in the FD entries of the sample Data divisions in the reference handbook. Try to explain the meaning of each clause to yourself. By the way, be sure to note that even the absence of a clause from an FD entry has some significance.*

Reading assignment: FILE DESCRIPTION ENTRY
- Level indicator
- File name
- RECORDING MODE clause
- BLOCK CONTAINS clause
- RECORD CONTAINS clause
- LABEL RECORDS clause
- DATA RECORD clause

●●●

191 A file description entry always gives the file name. We have previously studied file names (in lesson 4), because the file name also appears in a _____ entry in the _____ division.

●●●

SELECT; Environment

192 FD entries $\left\{\begin{matrix}\text{do}\\ \text{do not}\end{matrix}\right\}$ specify whether a file is to serve as input or as output.

●●●

do not

193 The level indicator and the file name are always required in a file description entry. Which clauses are always required?

●●●

LABEL RECORDS and DATA RECORD clauses

194 The LABEL RECORDS clause in the FD entry below signifies that there are

[no label records of any kind in the file]
[standard label records in the file, but no additional user labels]
[non-standard label records in the file].

```
FD  ACCOUNTS-RECEIVABLE
    LABEL RECORDS ARE OMITTED
    DATA RECORD IS COLLECTION-ACCOUNT.
```

●●●

EITHER no label records of any kind OR non-standard label records

If there are no label records (as in a card file), no further work is required of the programmer. If there are non-standard labels, the programmer doesn't get off quite so easily, since he must provide for the input or output, and processing, of these records. Even though the handling of non-standard labels is interesting, and often quite challenging, it would take us out of our way to discuss how it is done; suffice it to say that the programmer's life is simpler when label records are standard.

195 The file described in the entry below contains

[standard label records]
[non-standard label records]
[user label records].

```
FD  PRODUCT-USAGE-FILE
    RECORDING MODE V
    BLOCK CONTAINS 20 RECORDS
    LABEL RECORDS ARE AUDIT-DATA
    DATA RECORD IS PRODUCT-USAGE-RECORD.
```

●●●

BOTH standard label records AND user label records

Whenever a programmer-supplied name appears in the LABEL RECORDS clause, there are user labels in addition to standard labels in the file. The user labels provide information about the file beyond that given in the standard labels.

196 *Standard label records must meet the format standards specified by IBM for the System/360. These labels provide information used by the input-output control system (IOCS), and so are sometimes called "System standard" labels.*

Both user and non-standard label records are created and processed by the user. User (sometimes called "user standard") labels are those that meet certain minimum format standards specified by IBM; whereas non-standard (sometimes called "user non-standard") labels are those that do not follow IBM System/360 standards.

●●●

197 The DATA RECORDS clause [names] [describes] each kind of record in the file.

●●●

names ONLY

198 The three recording modes are ___.

●●●

V, F, and U

199 The letters V, F, and U stand for _____, respectively.

●●●

variable length, fixed length, and unspecified length

200 Block-length and record-length fields are found only in mode _____.

●●●

V

201 The length of a mode U record is "unspecified" in the sense that

{ there is no definite way of determining how many characters it contains
the programmer has been given no idea of how long the record might be
the length is not specified in a record-length or block-length field
the length might vary unpredictably anywhere from zero to infinity }.

●●●

the length is not specified in a record-length or block-length field

202 Variable length records are permitted in recording mode [V] [F] [U].

●●●

BOTH V and U

203 Which statement is true?

{ In recording mode V, there must be more than one record in each block.
In recording mode U, there must be more than one record in each block.
In recording mode V, there must be only one record in each block.
In recording mode U, there must be only one record in each block. }.

●●●

In recording mode U, there must be only one record in each block.

204 The names of the recording modes are a little deceptive. In particular, the mere fact that each block of a file contains one fixed-length record does not necessarily mean that the recording mode is F. Besides F, the recording mode for such a file might also be [V] [U].

●●●

EITHER V OR U

However, this choice is not available for files that are assigned to UNIT-RECORD devices. The recording mode for unit-record files (card files and printer files) can only be F.

205 In which mode or modes can there be one variable-length record per block?

●●●

either V or U

206 In which mode or modes can there be more than one variable-length record per block?

●●●

V only

207 True or false: Records written in mode V need not be variable-length.

●●●

True. (Fixed-length records may be written in mode V.)

208 True or false: Records written in mode F need not be fixed-length.

●●●

False. (Variable-length records cannot be written in mode F.)

209 *Turn to the case study program at the back of the reference handbook, and locate the FD entry for the BILLING-FILE. The next five frames refer to that entry.*

●●●

210 How many records are in each block of the BILLING-FILE?

●●●

five

211 How many different types of records are in this file?

●●●

one (BILLING-RECORD)

212 Did the programmer make an error in omitting the RECORD CONTAINS clause?

●●●

No. (It is permissible to omit this clause, for the compiler can determine how many characters the record contains from the pictures given in the record description.)

213 The recording mode of the BILLING-FILE is

$$\left\{\begin{array}{l}\text{mode V}\\ \text{mode F}\\ \text{not specified -- hence, mode U}\\ \text{unknown}\end{array}\right\}.$$

●●●

mode V (Whenever the RECORDING MODE clause is omitted, the mode is V.)

214 Does the BILLING-FILE therefore contain variable-length records?

●●●

No. Either fixed or variable-length records can be written in mode V.

This particular file happens to contain fixed-length records. You could not tell that from the FD entry, however. You will learn shortly how to recognize a variable-length record by its record description. (Of course, if the recording mode had been F, you would have known at once that record length was fixed.)

215 *Now locate the FD entry for the CUSTOMER-BILL-FILE in the case study program. The next three frames refer to that entry.*

●●●

216 How many records are in each block of the CUSTOMER-BILL-FILE?

●●●

one (The BLOCK CONTAINS clause is omitted when there is only one record per block.)

217 How many different types of records are in this file?

●●●

four (BILL-LINE-1, BILL-LINE-2, BILL-LINE-3, and BILL-LINE-4)

218 The recording mode of the CUSTOMER-BILL-FILE is F. Suppose that a record of the first type (BILL-LINE-1) contains 133 characters. From this you can conclude that

[all records of the first type must contain 133 characters]
[records of the other three types must also contain 133 characters]
[every record in the file contains exactly 133 characters].

●●●

ALL of these statements are correct. In recording mode F, all records in the file are the same length.

LESSON 6

219 *When you examined the sample Data divisions in the reference handbook, you saw that there are comparatively few kinds of entries that make up the division. First of all, there are the division and section header entries (DATA DIVISION, FILE SECTION, WORKING-STORAGE SECTION), whose meaning is obvious. Next, there are file description entries, which we have just finished discussing in some detail.*

All remaining entries in the division are item description entries. These may either be found in record descriptions, or they may be descriptions of independent items. In this lesson, we will discuss the concept of a "record description" and its relation to the structure of a record. (You will study specific details of item description entries in Lesson 7.)

Reading Assignment: RECORD DESCRIPTIONS

●●●

220 How many record descriptions will you find in a Data division?

$$\left\{\begin{array}{l}\text{One}\\ \text{One for each file}\\ \text{One for each type of record in each file}\end{array}\right\}$$

●●●

One for each type of record in each file

221 Suppose that a program is being written to process one input file and one output file. There are three types of input records, and one type of output record. Using "FD" to represent a file description entry, and "01" to represent a record description, the correct sequence of the File section for this program is

$$\left\{\begin{array}{l}\text{FD, FD, 01, 01, 01}\\ \text{FD, 01, 01, 01, FD, 01}\\ \text{01, 01, 01, 01, FD, FD}\end{array}\right\}.$$

●●●

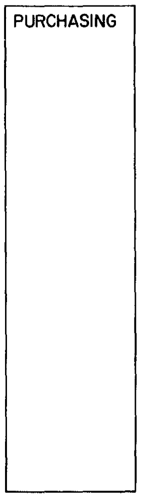
FD, 01, 01, 01, FD 01 (The record descriptions for each file must follow right after the file description entry for that file. The input file need not have been described first, so FD, 01, FD, 01, 01, 01 is another correct sequence.)

222 *A record description tells the structure of a record. It will help you a great deal in your study of COBOL if you are able to visualize that record structure. This is simple to do -- once you know how; and probably the simplest way to do it is by making a drawing of the record. We will proceed to make such a drawing, based on the sample record description given in the reference handbook under RECORD DESCRIPTIONS (your latest reading assignment). Turn to that sample record description.*

●●●

223 *We could draw various kinds of diagrams to represent the record. The one I will develop here is the one that seems to me to correspond most closely to the way the record description itself is organized.*

The first thing I will do is draw a box that represents the whole record. A record, you will remember, is the most inclusive data item. It is a storage area that will contain data when the program is executed. The name of this particular record is PURCHASING.



The level number of this record, like all records, is _____.

●●●

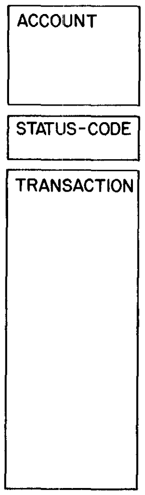
01

224 The record is now subdivided. This means that the same storage area that is known as PURCHASING is divided into smaller areas. In fact, certain parts of this storage area will be divided again and again, but the dividing is done one step at a time -- or in COBOL terms, one level at a time. The next level of this record is level 02. The three items at level 02 are _____.

●●●

ACCOUNT, STATUS-CODE, and TRANSACTION

225 *This drawing shows how the record has been subdivided so far. The three items shown below coincide with the original record.*

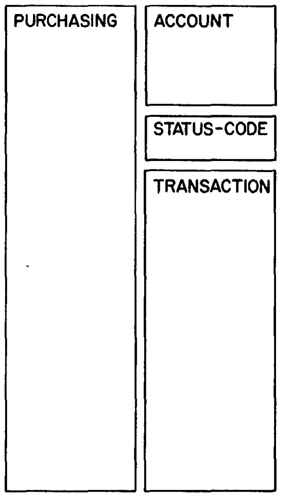


●●●

226 But we can still refer to the record as a whole by the name PURCHASING, so I will put the original box back into the drawing that we are developing.

The large box (on the left) and the three smaller boxes (on the right) represent

{the same, identical area in storage
two adjacent areas in storage
two separate storage areas of equal size}.



●●●

the same, identical area in storage

227 In a record description, an entry for a group item is followed by entries for the items that make it up. A group item comprises all the items described under it, until a level number equal to or less than the level number of the group item is encountered.

This means that the items that make up a group item

[must be described right after the group item]
[must have level numbers that are equal to or less than the group item]
[must have level numbers that are greater than the group item].

●●●

must be described right after the group item AND must have level numbers that are greater than the group item

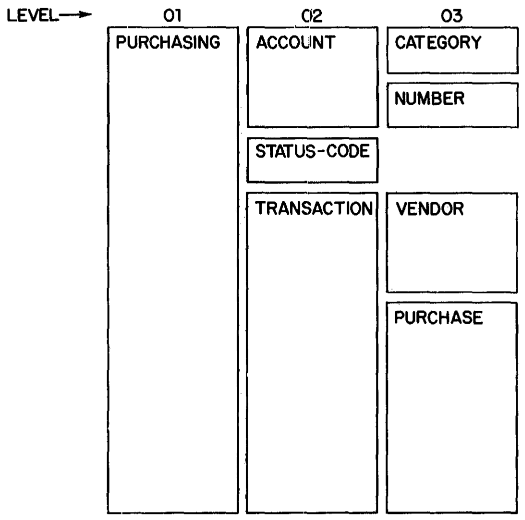
228 In the record description on which we are basing our drawing, a level 02 item which is further subdivided is followed by items with level number 03 or greater.

Which items defined at level 02 in this record description are further subdivided?

●●●

ACCOUNT and TRANSACTION are further subdivided; STATUS-CODE is not.

229 When level 03 is brought into our drawing, this is the result:



●●●

230 Since STATUS-CODE is not further subdivided, it is

$\left\{\begin{array}{l}\text{an independent item}\\ \text{a non-group item}\\ \text{an elementary item}\\ \text{a group item}\end{array}\right\}$.

●●●

an elementary item

231 You can tell which items are group items and which are elementary items by examining the level numbers that follow them in a record description. Group items are followed by items with higher level numbers. Elementary items are followed by items with equal or smaller level numbers.

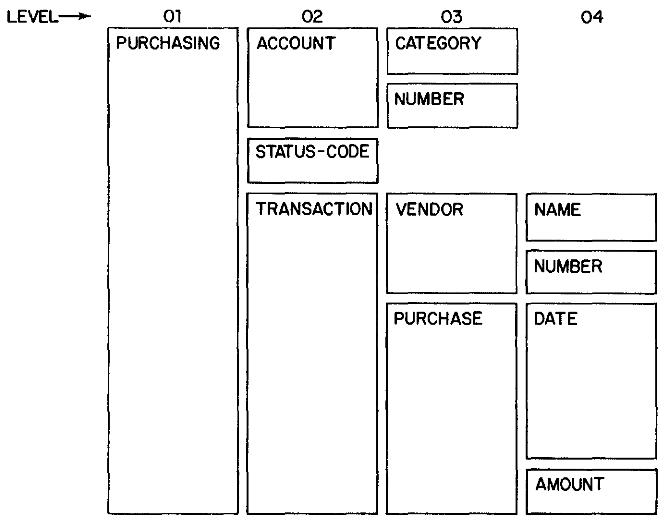
Which of the level 03 items in our sample record description are group items? Which are elementary items?

●●●

VENDOR and PURCHASE are group items; CATEGORY and NUMBER are elementary items.

You may have noticed that there is a level 04 item named NUMBER, and a level 03 item with the same name. Duplication of names is perfectly all right in COBOL, as long as the names are "qualified" when they are used in procedures, so the compiler will know precisely which item is being referred to. Qualification is done by naming the group item which the item with a duplicate name is part of; in the record description we are working with, the items would be qualified by calling them NUMBER OF ACCOUNT and NUMBER OF VENDOR, to make it clear which is which. You will study the rules for qualifying names in the next book of this series.

232 Here is our drawing with the level 04 items added.

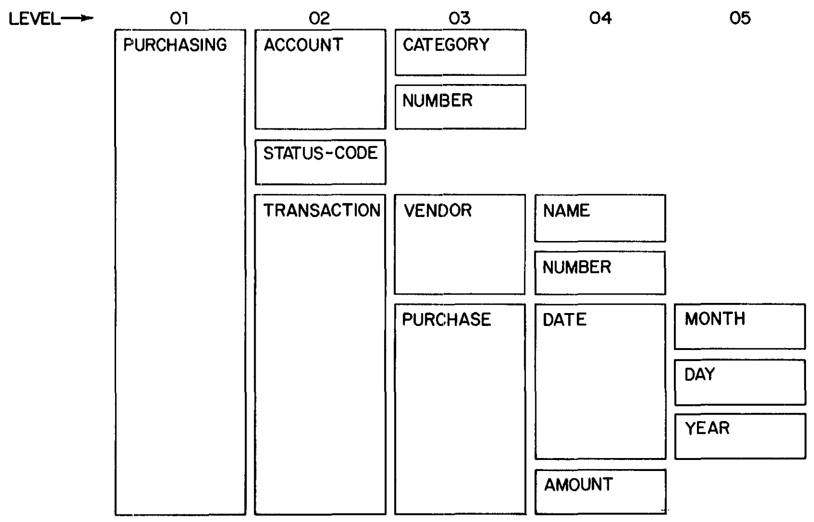


Which level 04 item is a group item? What items does it contain?

●●●

DATE is a group item, containing MONTH, DAY, and YEAR.

233 *Finally, we have a complete diagram of the structure of this record. In our diagram, the sizes of the boxes do not indicate the sizes of the items; actually, NAME contains 25 characters, while STATUS-CODE contains just 1 -- but the boxes for these two items are the same size. The reason for this is that we are not concerned about the sizes of items at this moment. Instead, we are concerned with the order in which the items appear in the record, and how the items are related to each other.*



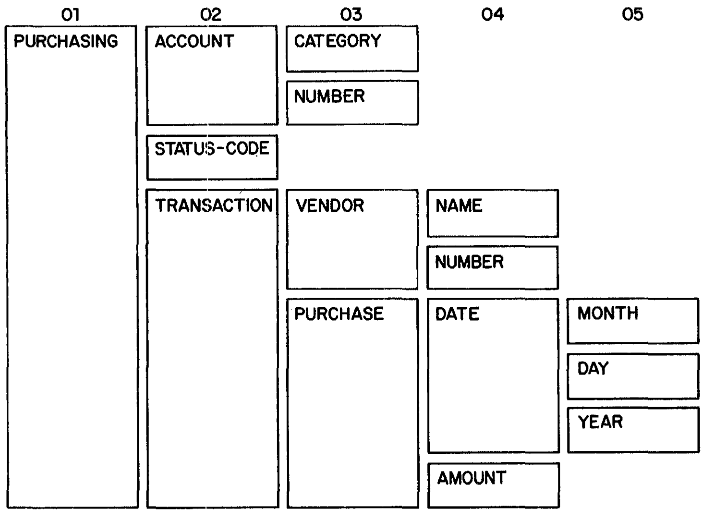
The diagram shows us the breakdown of a record into group and elementary items. Any item that is divided into smaller items at the next level is a group item. An item that is not divided into smaller items is an elementary item.

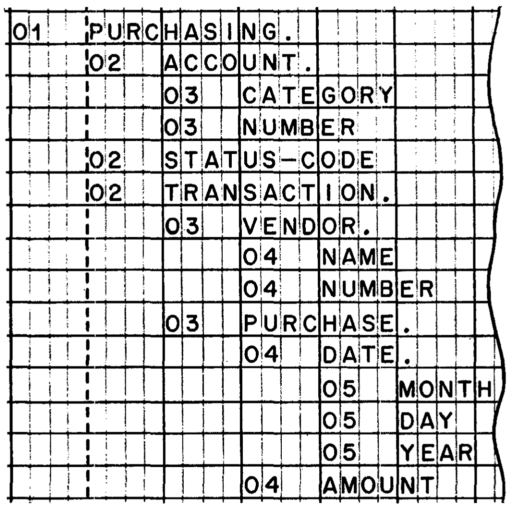
Pick out the group items and the elementary items in the record.

●●●

<u>Group items</u>	<u>Elementary items</u>
PURCHASING	CATEGORY
ACCOUNT	NUMBER
TRANSACTION	STATUS-CODE
VENDOR	NAME
PURCHASE	NUMBER
DATE	MONTH
	YEAR
	DAY
	AMOUNT

234 Now let's put our diagram side by side with the record description.





```
01     PURCHASING.
       02  ACCOUNT.
           03  CATEGORY
           03  NUMBER
       02  STATUS-CODE
       02  TRANSACTION.
           03  VENDOR.
               04  NAME
               04  NUMBER
           03  PURCHASE.
               04  DATE.
                   05  MONTH
                   05  DAY
                   05  YEAR
               04  AMOUNT
```

Which statement best describes the sequence in which the item description entries appear?

The level 01 entry appears first, followed by a level 02 entry, then entries for the items contained within the 02 item; then another 02 entry, and so on.
The level 01 entry appears first, followed by all of the level 02 entries, then all of the level 03 entries, 04 entries, and 05 entries.
The level 01 entry appears first, followed by a level 02 entry, then an 03 entry; and then a rhythmic cycle is repeated: 02, 03, 04; 03, 04, 05; and so on.

●●●

The level 01 entry appears first, followed by a level 02 entry, then entries for the items contained within the 02 item; then another 02 entry, and so on.

235 The item description entries are made

{ in any convenient sequence
in the sequence of the items in the record
in numerical order by level number } .

●●●

in the sequence of the items in the record

236 True or false: The higher the level number, the smaller the size of the item; for instance, a level 03 item is always smaller than a level 02 item.

●●●

False. The level number and size of an item are two completely separate considerations. Earlier we noted that there was a one-character level 02 item in our record, as well as a 25-character level 04 item.

237 The point made in the previous frame applies to items in general. But suppose we were dealing with a group item and the items that make it up. Then we would have a situation in which

[the group item must be larger than any item within it]
[the size of the group item is not related to the size of the items within it]
[the size of the group item is equal to the sum of the items within it].

●●●

the group item must be larger than any item within it AND the size of the group item is equal to the sum of the items within it

And, as we have already seen, the items that make up a group item must have higher level numbers than the group item. For example, in the record that we have been working with, DATE (level number 04) contains MONTH, DAY and YEAR (all level number 05); each of the level 05 items happens to contain 2 digits, so DATE contains 6 digits.

238 Some record descriptions contain entries that begin with level number 88. The reference handbook suggests that when you are analyzing the structure of the record, you can

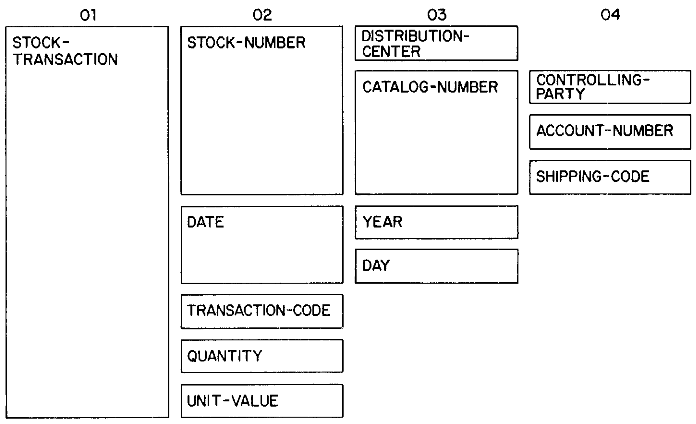
$$\left\{\begin{array}{l}\text{treat level 88 entries as if they were 01 entries}\\ \text{deal with level 88 entries just the same as other entries}\\ \text{ignore the level 88 entries}\end{array}\right\}.$$

●●●

ignore the level 88 entries

239 *Before we go on to look at the clauses that make up item description entries, I want you to try your hand at drawing a diagram to represent the structure of a record. Turn to the sample Data division in the reference handbook which is printed under the title, Another sample Data division, with entry types identified. Draw a diagram of the STOCK-TRANSACTION record. Make the same kind of diagram that you have seen in the preceding frames. Use scratch paper. Neatness doesn't count.*

●●●

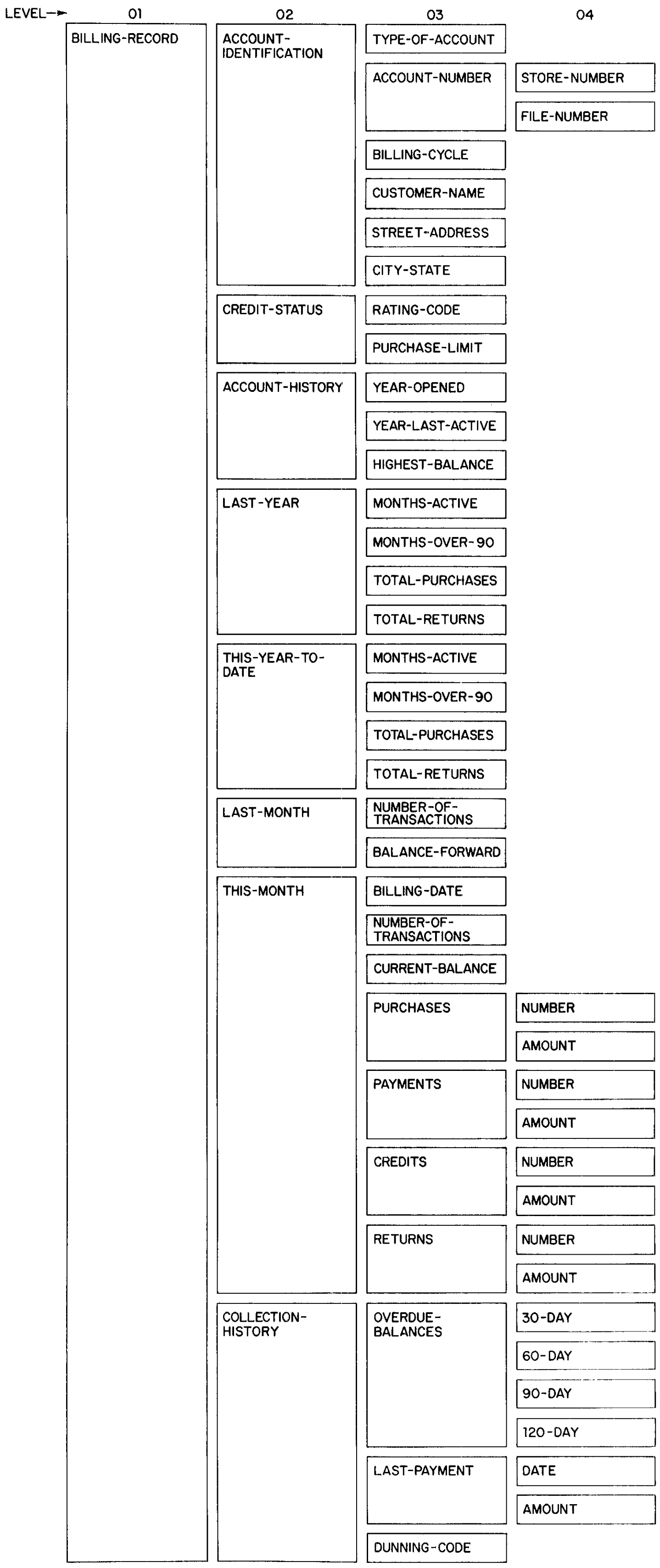


240 *Turn to the first page of the case study at the back of the reference handbook. There you will find another kind of diagram that is sometimes used to show the structure of a record. The horizontal format of the diagram is awkward in that we are forced to break off several times, but the sequence of items within the record is easy to see, as is the subdivision of items into smaller items.*

Although the levels of items are harder to see, some programmers prefer this sort of diagram because it corresponds more closely to the way the record would actually appear on tape or disk. On the next page in this textbook, the BILLING-RECORD has been diagrammed using the vertical format you have been taught. Compare the two kinds of diagrams.

Also, examine the record description of the BILLING-RECORD in the Data division of the case study program. It should be easy for you to pick out the item description entry for each of the items within the record structure. Simply keep in mind that every item description entry begins with a level number.

●●●


LEVEL
01
02
03
04
BILLING-RECORD
ACCOUNT-IDENTIFICATION
TYPE-OF-ACCOUNT
ACCOUNT-NUMBER
STORE-NUMBER
FILE-NUMBER
BILLING-CYCLE
CUSTOMER-NAME
STREET-ADDRESS
CITY-STATE
CREDIT-STATUS
RATING-CODE
PURCHASE-LIMIT
ACCOUNT-HISTORY
YEAR-OPENED
YEAR-LAST-ACTIVE
HIGHEST-BALANCE
LAST-YEAR
MONTHS-ACTIVE
MONTHS-OVER-90
TOTAL-PURCHASES
TOTAL-RETURNS
THIS-YEAR-TO-DATE
MONTHS-ACTIVE
MONTHS-OVER-90
TOTAL-PURCHASES
TOTAL-RETURNS
LAST-MONTH
NUMBER-OF-TRANSACTIONS
BALANCE-FORWARD
THIS-MONTH
BILLING-DATE
NUMBER-OF-TRANSACTIONS
CURRENT-BALANCE
PURCHASES
NUMBER
AMOUNT
PAYMENTS
NUMBER
AMOUNT
CREDITS
NUMBER
AMOUNT
RETURNS
NUMBER
AMOUNT
COLLECTION-HISTORY
OVERDUE-BALANCES
30-DAY
60-DAY
90-DAY
120-DAY
LAST-PAYMENT
DATE
AMOUNT
DUNNING-CODE

LESSON 7

241 *This lesson looks at item description entries in detail. Some item description entries are quite simple, consisting only of a level number and a programmer-supplied name. Others contain one or more clauses, in addition to a level number and a name. We will explore each clause in some depth.*

To keep you from being swamped by a great many facts, the reading assignment has been chopped up into a number of little pieces. You will first read about the level number and name. Then, in succeeding reading assignments, you will study the important clauses one at a time.

Reading assignment: ITEM DESCRIPTION ENTRIES
Level number
Name or FILLER

(Do not go on to the USAGE clause yet.)

●●●

242 Where would you look for the level number in an item description entry?

●●●

At the beginning; a level number is always the first element of the entry.

243 What level number signifies that the item is an independent item?

●●●

77

244 Which level number is associated with an entry that is not, strictly speaking, an item description? What type of entry is it?

●●●

88; condition-name entry

Condition names will be discussed in a later lesson.

245 Where would you look for the name of an item?

●●●

Just after the level number.

246 The word FILLER $\left\{\begin{matrix}\text{is}\\\text{is not}\end{matrix}\right\}$ a name.

●●●

is not

247 The word FILLER is most often used to indicate that a portion of a record contains no information. For instance, suppose that we are processing a file of punched cards, and that one of the card records contains only the current date plus an identifying code. The record description of the date card might look like this:

```
01  DATE-CARD.
    02  CARD-CODE           PICTURE X.
    02  CURRENT-DATE        PICTURE 9(6).
    02  FILLER              PICTURE X(73).
```

This record description indicates that a card code is the first character of the record, with the current date in the next six positions. Even though there is no more data, it is necessary to show that the record is 80 positions long; in order to "fill out" the record, the programmer has defined 73 additional positions and has called them _____.

●●●

FILLER

248 Let's take a slightly different case. Suppose we are processing a file of magnetic tape records: Each record contains 200 characters of data. The processing consists merely of splitting the file into two smaller files. The splitting is to be done on the basis of the control number in the first 12 positions of each record. If the control number falls within certain limits, the record is to be moved to one output area; otherwise, the record is to be moved to a different output area.

Examine the record description below and decide whether the word FILLER has been used properly in this case.

```
01  COMMODITY-RECORD.
    02  CONTROL-NUMBER      PICTURE 9(12).
    02  FILLER              PICTURE X(188).
```

The use of the word FILLER in this record description is

right, because there is no need for a procedure to refer separately to the data in the last 188 positions of the record.
wrong, because the word FILLER can only be used for blank positions, never for positions that contain data characters.
wrong, because there is no name by which a procedure can move all 200 characters of the record to an output area.

●●●

right, because there is no need for a procedure to refer separately to the data in the last 188 positions of the record.

The control number is the only item within the record that will be examined separately by the program. The only other processing required is to move the entire record to an output area, and for that purpose a procedure will use the name COMMODITY-RECORD.

249 *I think it may be useful to pursue this "file-splitting" problem, to explore other aspects of its solution, and especially to see what kinds of item description entries are involved.*

Suppose that the COMMODITY-RECORD is to be written in files called PLANT-1-PRODUCTS-FILE and PLANT-2-PRODUCTS-FILE. The whole record is moved to one file or the other with no changes being made to any of the items in the record. Here is what the file description entry and the record description might look like for one of these files:

```
FD  PLANT-1-PRODUCTS-FILE
    BLOCK CONTAINS 5 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS PLANT-1-PRODUCTS.

01  PLANT-1-PRODUCTS      PICTURE X(200).
```

The record description consists of a single item description entry that defines a storage area large enough to hold 200 characters. The PLANT-1-PRODUCTS record is not subdivided at all; in other words, here is an example of a record that is treated as an _____ item.

●●●

elementary

250 *For the PLANT-2-PRODUCTS-FILE, we could have a file description entry and a record description almost exactly like the one shown above. The record for that file could be called PLANT-2-PRODUCTS.*

You can visualize statements in the Procedure division of this program, saying MOVE COMMODITY-RECORD TO PLANT-1-PRODUCTS or MOVE COMMODITY-RECORD TO PLANT-2-PRODUCTS depending on the control number.

Suppose that records whose control numbers range from 309463552078 through 790084659302 are to be written in the first file, and all other records in the second file. These numbers are a little unwieldy, and the people in our firm don't go around with them on the tips of their tongues; it would mean more if we were to say that the first number is the control number for wheelbarrows, and that the second is the control number for pickup trucks.

●●●

251 *To sum it up, we would like to be able to write the following entry in the Procedure division:*

```
    IF CONTROL-NUMBER IS NOT LESS THAN
    WHEELBARROW,  AND CONTROL-NUMBER IS
    NOT GREATER THAN PICKUP-TRUCK,
        MOVE COMMODITY-RECORD TO
            PLANT-1-PRODUCTS,
        WRITE PLANT-1-PRODUCTS;
    OTHERWISE, MOVE COMMODITY-RECORD TO
            PLANT-2-PRODUCTS,
        WRITE PLANT-2-PRODUCTS.
```

The entry uses the name WHEELBARROW in place of 309463552078, and PICKUP-TRUCK in place of 790084659302. In order to use these names, they must be defined; the required item description entries would look like this:

```
77  WHEELBARROW,  PICTURE 9(12),
        VALUE '309463552078'.
77  PICKUP-TRUCK,  PICTURE 9(12),
        VALUE '790084659302'.
```

From their level numbers, you can identify these entries as descriptions of _____ items.

●●●

independent

These entries actually cause the values to be stored as constants. The procedural entry at the top of the page causes the control number of a record to be compared with the constant values. We will have much more to say about procedural entries later in this book.

252 *During our brief look at the hypothetical "file-splitting" problem, we have seen several item description entries. We will carry this just one more step further, by combining all of the entries into a division, as they would appear in a program.*

```
DATA DIVISION.
FILE SECTION.
FD  COMMODITY-FILE
    BLOCK CONTAINS 5 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS COMMODITY-RECORD.
01  COMMODITY-RECORD.
    02  CONTROL-NUMBER      PICTURE 9(12).
    02  FILLER              PICTURE X(188).
FD  PLANT-1-PRODUCTS-FILE
    BLOCK CONTAINS 5 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS PLANT-1-PRODUCTS.
01  PLANT-1-PRODUCTS        PICTURE X(200).
FD  PLANT-2-PRODUCTS-FILE
    BLOCK CONTAINS 5 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS PLANT-2-PRODUCTS.
01  PLANT-2-PRODUCTS        PICTURE X(200).
WORKING-STORAGE SECTION.
77  WHEELBARROW             PICTURE 9(12)
        VALUE '309463552078'.
77  PICKUP-TRUCK            PICTURE 9(12)
        VALUE '790084659302'.
```

Before you proceed to the next frame, take some time to look through the above division. Be sure that you can distinguish file description entries from item description entries. Be able to identify where each entry begins and ends. Recognize that the two sections of the division serve different purposes.

●●●

253 *Now we will discuss the clauses that are found in item description entries, beginning with the USAGE clause. We will concentrate on USAGE before going on to clauses such as PICTURE and VALUE, which you have seen in several item description entries. Although the USAGE clause has not appeared in the entries that you have seen, you will discover that every item has a particular usage -- even when the clause is omitted.*

The reading assignment contains a number of specialized terms with which you are expected to be familiar: bits, bytes, floating-point, packed decimal, binary, hexadecimal, etc. Don't get the impression that we are suddenly changing the rules of the game, and making COBOL "machine-oriented"; we have simply come to the point where the programmer must specify precisely how the data appears. Also, don't be concerned if you can't remember the exact definitions of these terms; as long as you have a general idea of their meaning, you will get along fine.

Reading assignment: ITEM DESCRIPTION ENTRIES (continued)
USAGE clause
What the usage words indicate

(Do not go on to the PICTURE clause yet.)

●●●

254 The reserved word USAGE $\left\{\begin{matrix}\text{must appear}\\ \text{need not appear}\end{matrix}\right\}$ in a USAGE clause.

●●●

need not appear

255 Since the word USAGE may be missing, you must look for one of the five reserved words that specify the usage of the item. Those five words are _____.

●●●

DISPLAY, COMPUTATIONAL, COMPUTATIONAL-1, COMPUTATIONAL-2, and COMPUTATIONAL-3

256 Which one of the five usage words may be omitted?

●●●

DISPLAY

257 Match the list of terms below. (Two or more of the terms on the right may apply to a usage word.)

COMPUTATIONAL-1	internal decimal
COMPUTATIONAL-2	external decimal
DISPLAY	packed decimal
COMPUTATIONAL-3	binary-coded decimal
COMPUTATIONAL	binary
	floating-point
	EBCDIC
	BCD

●●●

COMPUTATIONAL-1 and COMPUTATIONAL-2 = floating-point
DISPLAY = external decimal, binary-coded decimal, EBCDIC, BCD
COMPUTATIONAL-3 = internal decimal, packed decimal
COMPUTATIONAL = binary

258 *The basic idea should be clear to you: In System/360 COBOL, we can process data in five different formats. Thus, we can take advantage of the flexibility of data representation that is built into the System/360. It is not our purpose in this course, though, to discuss the reasons for having different data codes, nor to explain which codes are best for which purposes -- those are "system design" considerations, not COBOL considerations.*

The question that we want to tackle is, how can you figure out what the usage of an item is by reading the item description entry? Sometimes the answer is simple; for example, if the entry says COMPUTATIONAL-3, there is no doubt at all -- the data is in packed decimal. But what if there is no usage word in the entry, as in the entry below? What is the data code in that case? We will explore this next.

```
       03  AMOUNT-PAID, PICTURE S9(6)V99.
```

●●●

259 There are two key facts to keep in mind. The first is that the usage specified for a group item

{ has no bearing on the usage of the items in the group
applies to the first item in the group, but not to the rest
applies to all of the items in the group }.

●●●

applies to all of the items in the group

260 The second fact is that when no usage at all is specified for an elementary item or for a group it is part of,

{ the item has no usage
the usage can be any one of the five possibilities
the usage is assumed to be DISPLAY } .

●●●

the usage is assumed to be DISPLAY

261 So, if no usage is specified for an elementary item, the first thing to do is to see if usage has been specified for a group item that this item is part of. What is the usage of AMOUNT-PAID in the illustration below?

```
       02  PAYMENT, COMPUTATIONAL-3.
           03  AMOUNT-DUE, PICTURE S9(6)V99.
           03  AMOUNT-PAID, PICTURE S9(6)V99.
```

●●●

COMPUTATIONAL-3, because that is the usage of the group item that AMOUNT-PAID is part of; AMOUNT-DUE has the same usage.

262 Examine the item description entry below, and then select the most accurate statement regarding the usage of the item.

```
77  GROUP-TOTAL    PICTURE S9(6)V99.
```

{ Since no usage has been specified for this item, it must be DISPLAY.
To determine its usage, you must see the entry for the group item it is part of.
This item has whatever usage is specified for the record it is part of.
There is no way of determining its usage without a large crystal ball. }

●●●

Since no usage has been specified for this item, it must be DISPLAY.

The level number informs you that this is an independent item, and therefore not part of a group item. What we have here, then, is a self-contained item description entry; there is absolutely no point in looking at the descriptions of any other items to find out the usage of an independent item.

263 The record description below seems to indicate conflicting usages. No usage is specified for the group item, nor for most of the items within the group, yet one of the items within the group is COMPUTATIONAL.

There is really no conflict. It is permissible for items within a group to have varying usages. Avoid the misimpression that the usage of a group item must be DISPLAY unless otherwise specified; in fact, the usage of a group item is the same as the combined usages of the elementary items it contains. Thus, the usage of STOCK-TRANSFER in the example below is

{ COMPUTATIONAL, since that is the only specified usage
DISPLAY, since that is the only specified usage
part DISPLAY and part COMPUTATIONAL } .

```
01  STOCK-TRANSFER.
    02  STOCK-NUMBER    PICTURE X(7).
    02  DESCRIPTION     PICTURE X(15).
    02  UNITS-OF-STOCK  PICTURE S9(8)
            COMPUTATIONAL.
```

●●●

part DISPLAY and part COMPUTATIONAL

There would be a conflict only if one usage were specified for the group item, and a different usage were specified for the elementary items within the group -- for example, if the word DISPLAY had actually been written in the level 01 entry above. Such a conflict, incidentally, would be diagnosed as a programmer error by the COBOL compiler.

264 When you are trying to figure out the usage of an elementary item whose description does not contain a USAGE clause, you first look for a USAGE clause in the description of a group item that contains the elementary item. If you don't find a USAGE clause in those item description entries, you can conclude that the usage is DISPLAY.

It will be useful to keep in mind what you have already learned about the structure of records, in particular the idea that items can be subdivided several times. Or, from another point of view, that an item can be part of several group items. In the example below (which you may recall seeing before), MONTH is part of DATE, which is part of PURCHASE, which is part of TRANSACTION, which is part of PURCHASING. In other words, the elementary item MONTH is part of four group items.

```
01  PURCHASING.
    02  ACCOUNT.
        03  CATEGORY        PICTURE XX.
        03  NUMBER          PICTURE 9(6).
    02  STATUS-CODE         PICTURE X.
    02  TRANSACTION.
        03  VENDOR.
            04  NAME        PICTURE X(25).
            04  NUMBER      PICTURE 9(6).
        03  PURCHASE.
            04  DATE.
                05  MONTH   PICTURE 99.
                05  DAY     PICTURE 99.
                05  YEAR    PICTURE 99.
            04  AMOUNT      PICTURE 9(4)V99.
```

There is no USAGE clause in the item description entry for MONTH, so we must look for a USAGE clause in the description entries for all of the group items of which MONTH is a part. Since we find no USAGE clause in any of these entries, we know that the usage of MONTH must be _____.

●●●

DISPLAY

265 *Next, the PICTURE clause. You have seen PICTURE clauses in every record description, though not in every item description entry. PICTURE clauses are easier to identify than USAGE clauses, since they always contain the reserved word PICTURE. Also, PICTURE clauses cannot optionally be omitted, as USAGE clauses sometimes can; the rules are cut and dried -- there must be a PICTURE clause actually written in certain cases, and in all other cases there must not be a PICTURE clause.*

Learn which item description entries must contain pictures, and what you can tell about an item from its picture. Spend a little extra time with the six common picture characters that are explained in the reference handbook. Also note -- but don't try to memorize -- the relationship of picture and usage.

Reading assignment: ITEM DESCRIPTION ENTRIES (continued)
PICTURE clause
How to identify an item from its picture
What some common picture characters mean
How picture and usage are related

(Do not read about the VALUE clause yet.)

●●●

266 Group items $\left\{\begin{array}{l}\text{always}\\ \text{sometimes}\\ \text{never}\end{array}\right\}$ have pictures.

●●●

never

267 Pictures are required in the item description entries of

$\left\{\begin{array}{l}\text{all elementary items}\\ \text{all elementary items except those that are only one}\\ \quad\text{character long}\\ \text{all elementary items except internal floating-point items}\end{array}\right\}$

●●●

all elementary items except internal floating-point items (that is, all except COMPUTATIONAL-1 and COMPUTATIONAL-2 items)

268 From a picture, you can tell what category an item falls into (alphanumeric, alphabetic, numeric, etc.), how many characters the item contains, and what kinds of characters they are.

From the picture in this entry, determine the category of the item, as well as the number and type of characters.

```
       03  SHIPPING-ROUTE  PICTURE XXXX.
```

●●●

alphanumeric; 4 characters; may be letters, digits, special characters, or spaces

269 The usage of SHIPPING-ROUTE (see entry in previous frame)

can be determined only by looking at the entries for the group items that contain this item
is most likely to be COMPUTATIONAL because shipping routes are generally numbers
must be DISPLAY because alphanumeric items can only have DISPLAY usage
must be COMPUTATIONAL-3 (packed decimal) because the picture characters are packed together
.

●●●

must be DISPLAY because alphanumeric items can only have DISPLAY usage

Translated from COBOLese, this means: the only data code in which alphanumeric characters can be stored in the System/360 is BCD (or if you prefer, EBCDIC or external decimal). All other ways of representing data in System/360 (binary, packed decimal, and internal floating-point) can be used only to store numeric data.

270 The fact that SHIPPING-ROUTE is alphanumeric means that the item may contain

[a number, such as 1234]
[letters, such as QWTG]
[a combination of characters, such as EC25]

●●●

a number, such as 1234 OR letters, such as QWTG OR a combination of characters, such as EC25

271 *The preceding frame makes the point that "alphanumeric" is a catch-all category, and that the data stored in alphanumeric items may be numeric, alphabetic, or mixed. This fact sometimes leads beginning programmers to jump to the conclusion that they can define all data items as alphanumeric. Why should they bother with As and 9s and all those other symbols, when Xs can stand for any characters imaginable?*

Briefly, the answer is that an item's picture affects the way the item can be processed. For instance, an item can be involved in an arithmetic operation only if it is a numeric item. Also, only numeric data can be moved into a report item; that is, only numeric data can be edited (punctuated). Similarly, data cannot be moved from an alphanumeric item into a numeric item. All of these examples show how important it is for the picture to be an accurate description of a data item; in particular, how important it is to distinguish between "any old characters" (represented by Xs) and "digits" (represented by 9s).

Rules of the sort that have just been mentioned are discussed in later books of this series. In this book, we are concerned with what pictures look like and what they mean, not with the whys and wherefores of composing pictures.

●●●

272 Which if these might be the picture of a numeric item:

[99,999.99] [99V99] [9(6)] [99099099] [+9999] [S9(8)].

●●●

99V99 OR 9(6) OR S9(8) (All the rest are pictures of report items.)

273 If an item's picture is 9999V999, how many digits does the item contain? How many decimal places?

●●●

seven digits; three decimal places

The actual value of the number in storage might be 1234567. The item in storage does not contain an actual decimal point character, but because of the V in its picture, that value would be treated as if it were 1234.567.

274 Suppose that the actual number stored in an item is 3975. As what value would this number be treated if the item's picture were:

1. 9(4)
2. P(4)9(4)
3. 999V9
4. 9999PPV
5. V9(4)

●●●

1. 3975 (No assumed decimal point or assumed zeros are indicated, so the value is treated as a whole number; the result is the same as if the picture were 9999V).
2. .00003975 (This picture might also have been VP(4)9(4), but the V is not necessary since its presence is implied by the Ps. The Ps say, in effect, "the assumed decimal point is located four positions in front of the first actual digit"; thus, there are four assumed zeros.
3. 397.5
4. 397500 (This picture might also have been written as 9999PP, without the V.)
5. .3975

275 Although group items do not have pictures, you can figure out what a group item looks like by examining the pictures of the elementary items that make it up. From the entries below, you can determine that ACCOUNT-NUMBER is a _____ -digit item made up of STORE-NUMBER, which is a _____ -digit item, and FILE-NUMBER, which is a _____ -digit item.

```
       03  ACCOUNT-NUMBER.
           04  STORE-NUMBER       PICTURE 999.
           04  FILE-NUMBER       PICTURE 9(4).
```

●●●

7; 3; 4

276 According to the picture in the entry below, the number of digits that will be stored in BALANCE is

$$\left\{\begin{array}{c}6\\7\\8\end{array}\right\}.$$

```
    02  BALANCE, PICTURE S9(4)V99,
                COMPUTATIONAL.
```

●●●

6 (Each 9 represents a digit; neither S nor V represents a character in storage.)

277 The item illustrated in the previous frame has COMPUTATIONAL usage; therefore, the data code in which the digits will be stored is binary. Instead of merely saying that BALANCE contains six digits, it would be more accurate to say that BALANCE contains

$$\left\{\begin{array}{l}\text{six binary digits}\\\text{the binary equivalent of six decimal digits}\\\text{six digits in binary-coded decimal (BCD) format}\end{array}\right\}.$$

●●●

the binary equivalent of six decimal digits

When usage is COMPUTATIONAL, the data is stored in "true binary" form, not in binary-coded decimal (BCD) -- which, after all, is what we call DISPLAY usage in COBOL. It takes far more than six binary digits -- in fact, it takes twenty binary digits -- to store the equivalent of six decimal digits.

For the most part, COBOL programmers are not concerned with data codes, or with bits and bytes in storage. Instead they "think decimal" -- even when they are dealing with packed decimal or binary data items -- because the 9s in a picture represent decimal digits.

278 *We have spent a generous amount of time studying the USAGE and PICTURE clauses. The remaining clauses, VALUE, OCCURS, and REDEFINES probably deserve equal time, but it would take us a little afield -- into topics such as data tables -- if we were to explore them thoroughly; therefore, we will limit ourselves to a brief look at the final three clauses.*

Your objective is to learn how each clause appears, and what it means; not to learn the applications for which a programmer would use it -- except in the most general sense.

Reading assignment: ITEM DESCRIPTION ENTRIES (continued)
VALUE clause
OCCURS clause
REDEFINES clause

●●●

279 *One use of the VALUE clause is in level 88 (condition-name) entries. We will skip over this use for the time being, until a later lesson in this book.*

Another use of the VALUE clause is to assign initial values to constants or work areas in storage. Entries that use the VALUE clause in this way are not allowed in the File section; you would look for them in the _____ section of the Data division.

●●●

Working-Storage

This is the reason behind COBOL's ban on VALUE clauses for items in the File section: Items described in the File section constitute input or output records. The values of such items are not expected to be constant; they will vary with each record that is processed. For instance, each time an input record is read, a value enters each of the items in the record. There is therefore no good reason to assign initial values to those items; to do that would only mislead some programmers into thinking that the initial values would remain constant.

280 In a VALUE clause, the word VALUE (or VALUE IS) is followed by _____.

●●●

a literal

281 Even though both examples of VALUE clauses in the reference handbook show numbers as literals, any valid literal may appear in a VALUE clause. The example below uses letters and spaces for the literal value.

As you might have expected, the literal in the VALUE clause must be consistent with the picture of the item. In the example above, the picture tells us this is an alphabetic item containing 15 characters; hence, the literal must be non-numeric, and must contain no more than 15 letters and/or spaces.

If an item happened to be COMPUTATIONAL, and its picture was S9(7)V99, you can guess that a _____ literal would have to be used in the VALUE clause.

●●●

numeric

282 Another acceptable format of the VALUE clause makes use of reserved words such as ZERO and SPACE. Here is an example:

```
77  TRANSACTION-COUNT, PICTURE 999,
        VALUE ZERO.
```

Earlier, you learned that ZERO is a

$$\left\{\begin{array}{l}\text{literal constant}\\ \text{reserved constant}\\ \text{numeric constant}\\ \text{figurative constant}\end{array}\right\}.$$

●●●

figurative constant

283 In an OCCURS clause, the word OCCURS is followed by a _____ literal.

●●●

numeric (more specifically, the numeric literal must be an unsigned whole number)

284 The OCCURS clause specifies how many times an item appears, in sequence, in storage. The entry below describes a situation in which an item appears in storage 12 times in a row. That is, there are 12 items, each one named MONTH, and each one containing (how many?) _____ letters and/or spaces.

```
       02  MONTH, PICTURE A(9),
               OCCURS 12 TIMES.
```

●●●

9

285 When an item's name is followed by the word REDEFINES, an area of storage is not reserved for the item; rather, a second name is assigned to a previously defined item.

Examine these entries, and answer the questions below.

```
           03  PURCHASE, PICTURE 9(4)V99.
           03  REFUND REDEFINES PURCHASE,
                  PICTURE 9(4)V99.
```

1. For which item will an area of storage be reserved?
2. For which item will a new area of storage not be reserved?
3. By what two names will the originally reserved area of storage be known?

●●●

1. An area of storage will be reserved for PURCHASE.
2. A new area of storage will not be reserved for REFUND.
3. The area of storage originally reserved for PURCHASE will be known both as PURCHASE and as REFUND.

This is an example of an item that has a dual purpose in a record. Sometimes the item contains the "purchase" amount, and sometimes the "refund" amount. We would suppose that a code somewhere in the record indicates whether the transaction is a purchase or a refund; based on that code, the program procedures would do something either with PURCHASE or with REFUND. Note that, in this case, it is just a matter of having two names for an item, so the appropriate name can be used in procedures.

LESSON 8

286 *At last we have arrived at the final division, the Procedure division. To some people the Procedure division represents "the program", for it consists of procedures which the computer is to follow in processing the data. The entries in this division are very similar to English, so they will be easier for you to comprehend at once than some of the Data division entries were; also, there is very little in the way of special COBOL terminology, as compared with the multitude of special terms that popped up in your study of the Data division.*

To be sure, there are a few such special terms, and the first one that you will come across is "procedure". In this reading assignment, you will examine a sample Procedure division, and then you will learn exactly what we mean when we use the word "procedure" in talking about COBOL. At the same time, you will learn what we mean in COBOL by "sentences" and "statements".

Reading assignment: PROCEDURE DIVISION ENTRIES
 Sample Procedure division
 PROCEDURES

●●●

287 The six header entries that appear in the sample Procedure division in the reference handbook are _____. (Refer to the handbook, you were not expected to remember this, of course.)

●●●

PROCEDURE DIVISION.
BEGINNING-OF-JOB.
DETAIL-PROCESSING.
READ-NEXT-CARD.
TOTAL-ROUTINE.
END-OF-JOB.

288 You will recall that the structural units of COBOL programs are divisions, sections, paragraphs, and entries.

The first of the six header entries you have just looked at is a _____ header entry. The other five are _____ header entries.

●●●

division; paragraph

289 Each paragraph in the Procedure division is called a

$\left\{\begin{array}{l}\text{sentence}\\ \text{procedure}\\ \text{statement}\end{array}\right\}$.

●●●

procedure

290 In addition to paragraphs, the word "procedure" applies to _____ in the Procedure division.

●●●

sections

291 A "sentence" is any Procedure division entry that is not a _____ entry.

●●●

header

292 Like all entries, a sentence must be terminated by a _____.

●●●

period

293 A procedure must contain

$\left\{\begin{array}{l}\text{one sentence, or less}\\ \text{one or more sentences}\\ \text{more than one sentence}\end{array}\right\}$.

●●●

one or more sentences

294 Since a procedure might contain just one sentence, the shortest imaginable procedure would contain $\left\{\begin{array}{c}1\\ 2\\ 3\end{array}\right\}$ entries.

●●●

2 (a header entry in addition to a sentence)

295 The entry below calls for three actions: READ... CLOSE... and GO TO... The COBOL term used for a specification of action, such as READ OLD-MASTER, is

$$\left\{\begin{array}{l}\text{phrase}\\ \text{clause}\\ \text{statement}\\ \text{sentence}\end{array}\right\}$$

```
    READ OLD-MASTER; AT END, CLOSE
        OLD-MASTER, GO TO FINISH-NEW-MASTER.
```

●●●

statement

296 The complete procedure entry illustrated in the preceding frame is called a _____.

●●●

sentence

297 *Each statement begins with a procedural word, like ADD, MOVE, etc. Most procedural words are verbs, though one of the most important words -- IF -- is technically a conjunction. The reference handbook lists the seventeen most commonly used procedural words in alphabetical order, but for study purposes we will look at just a few words at a time. To begin with, you will read about four input-output verbs. Your objective is to learn enough about each verb to be able to read and comprehend statements using that verb. Keep in mind that you are not expected to be able to write original statements yet, so don't try to memorize the format rules.*

Reading assignment: PROCEDURAL WORDS
OPEN
READ
WRITE
CLOSE

(Be sure to read the summaries of these verbs in this sequence.)

●●●

298 Before a record may be written in a file, it is necessary to

$\left\{\begin{matrix}\text{READ}\\\text{OPEN}\\\text{CLOSE}\end{matrix}\right\}$ the file.

●●●

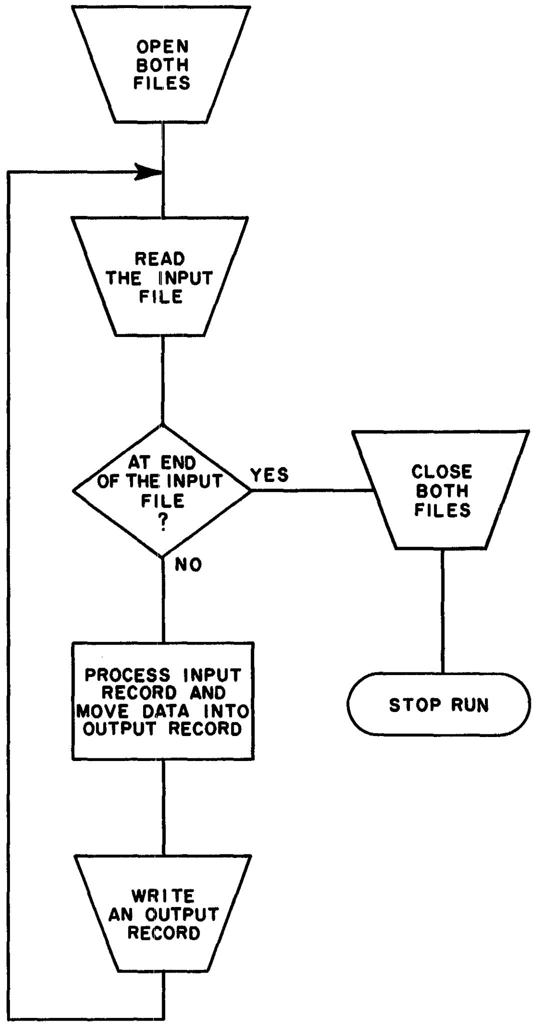
OPEN

299 A CLOSE statement is required after processing is completed for [input files] [output files].

●●●

input files AND output files

300 The flowchart below shows the overall logic of processing sequential files. In this example, there is one input file and one output file. For simplicity's sake, an output record is written for every input record.



This processing logic applies to sequential files that are stored on [magnetic tape] [magnetic disk] [punched cards] [printed forms].

●●●

ALL of these storage media

Printed forms could serve only for the output file, of course, while the other media could serve for both input and output.

301 The flowchart you just looked at contains a decision block testing whether the end of the input file has been reached. This AT END test is a clause of the

$\left\{\begin{array}{l}\text{READ}\\ \text{WRITE}\\ \text{OPEN}\\ \text{CLOSE}\end{array}\right\}$ statement.

●●●

READ

302 Which input-output statement is generally acted on at the beginning of a run? At the end of a run?

●●●

Beginning: OPEN; end: CLOSE

303 OPEN and CLOSE statements are acted on once for each $\left\{\begin{array}{l}\text{file}\\ \text{record}\end{array}\right\}$, whereas READ and WRITE statements are acted on once for each $\left\{\begin{array}{l}\text{file}\\ \text{record}\end{array}\right\}$.

●●●

file; record

304 *Turn to the Procedure division of the case study program in the reference handbook. The next six frames all refer to the case study program.*

●●●

305 What is the name of the procedure in which the files are opened?

●●●

START-PROCESSING

306 What is the name of the input file? What is the name of the output file?

●●●

BILLING-FILE; CUSTOMER-BILL-FILE

When these files were named in the Environment division and described in the Data division, no indication was given as to which file would be used for input, and which for output. The OPEN statement is where this is specified. This arrangement permits the same Environment and Data division entries to be used in other programs, where for instance, the input file of this program might be the output file.

307 Find the READ statement. What procedure is the computer told to go to when the end of the input file is reached?

●●●

END-OF-RUN

308 Find the procedure named END-OF-RUN. What does the first sentence of that procedure say?

●●●

CLOSE BILLING-FILE, CUSTOMER-BILL-FILE.

309 Both the OPEN and CLOSE statement gives the names of two files; however, the OPEN statement tells which file is _____ and which is _____, while the CLOSE statement does not.

●●●

INPUT, OUTPUT

310 How many WRITE statements are there?

●●●

four

311 *Now let's go back to read about a couple of other procedural words that deal with input and output. Determine the difference between these new words and READ and WRITE.*

Reading assignment: PROCEDURAL WORDS
ACCEPT
DISPLAY

●●●

312 ACCEPT and DISPLAY are used for reading and writing _____, not for _____.

●●●

low-volume data; (not) files of data

313 What is the maximum number of characters that can be read by an ACCEPT statement, and written by a DISPLAY statement?

●●●

maximum for ACCEPT: 80; maximum for DISPLAY; 120

314 ACCEPT and DISPLAY statements are used to receive information typed in on the console typewriter keyboard, and to type out information on the typewriter. For this use, the number of characters that can be read or written cannot exceed _____.

●●●

72

315 *Now that you have seen the verbs that fall into the "input-output" category, let's look at the "data movement" category. This category has just one basic verb: MOVE. You will find that this verb can do three main things, though not necessarily all three at one time.*

Reading assignment: PROCEDURAL WORDS
MOVE

●●●

316 A MOVE statement can _____, _____, and _____ data.

●●●

move, convert, edit

317 Most people find a MOVE statement easy to understand, but let's take a moment for you to double-check your understanding of it. In the sentence below, what would you say is the name of the "source field" -- the item from which data is to be moved? What is the name of the "receiving field" -- the item to which the data is to be moved?

```
    MOVE CATALOG-NUMBER TO CONTROL-ITEM.
```

●●●

CATALOG-NUMBER is the source field; CONTROL-ITEM is the receiving field.

318 When the COBOL compiler generates machine language instructions for a MOVE statement, it refers to the descriptions of the items to be moved as given in the Data division. The item descriptions will indicate to the compiler how many characters are to be moved, what the usage of each item is, etc.

In reference to the MOVE sentence printed in the previous frame, suppose that CATALOG-NUMBER is a binary item (that is, its USAGE IS COMPUTATIONAL). And suppose that the usage of CONTROL-ITEM is DISPLAY. In this case, the compiler will generate the instructions required not only to move CATALOG-NUMBER to CONTROL-ITEM, but also to convert the data from (which item?) _____ to external decimal (BCD) code.

●●●

CATALOG-NUMBER

319 A MOVE statement would cause data to be edited if the _____ of the receiving item identified it as a report item.

●●●

picture

320 *"Arithmetic" is a larger category: five verbs. An interesting thing about this group of verbs is that one of them can replace the other four. It would be possible, then, to use just one arithmetic verb for all arithmetic statements; however, you will have a look at the other four verbs as well.*

Reading assignment: PROCEDURAL WORDS
- COMPUTE
- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE

●●●

321 Which verb can be used for all arithmetic operations?

●●●

COMPUTE

322 Whereas other statements use words like ADD, SUBTRACT, etc., to specify arithmetic operations, COMPUTE statements use _____ to specify operations.

●●●

arithmetic symbols

323 Notice that a COMPUTE statement not only computes a value, but also edits that value if the result is to be put into a report item. In the example below, into which item will be result be put? How could you tell if it is a report item?

```
    COMPUTE RATE = DISTANCE / TIME.
```

●●●

The result will be put into RATE (the item named after the verb). To find out if RATE is a report item, you would have to know what its picture is.

324 A DIVIDE statement might have been used to do the calculation specified in the COMPUTE statement above. In order to obtain an edited result, the DIVIDE statement would have to contain a _____ clause.

●●●

GIVING

325 Is this DIVIDE statement equivalent to the COMPUTE statement illustrated above?

```
    DIVIDE DISTANCE INTO TIME GIVING RATE.
```

●●●

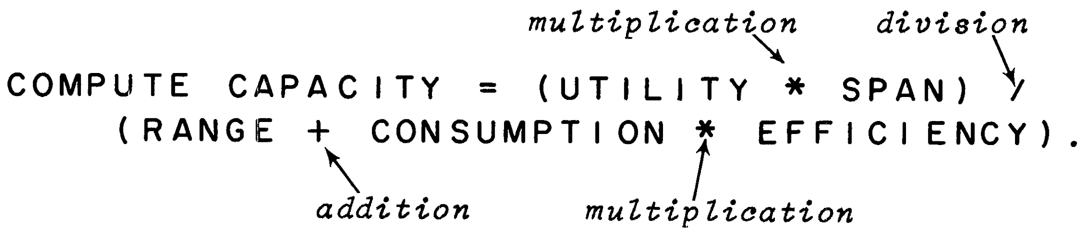
No. The correct equivalent would be DIVIDE TIME INTO DISTANCE GIVING RATE.

In the COMPUTE statement, the formula DISTANCE/TIME would be read "distance divided by time"; the meanings of arithmetic symbols are given in your reference handbook under the topic SYMBOLS.

326 A significant difference between COMPUTE and other verbs is that a COMPUTE statement can call for more than one kind of arithmetic operation. Name the operations that are called for in this statement.

```
    COMPUTE CAPACITY = (UTILITY * SPAN) /
        (RANGE + CONSUMPTION * EFFICIENCY).
```

●●●


multiplication
division
COMPUTE CAPACITY = (UTILITY * SPAN) /
(RANGE + CONSUMPTION * EFFICIENCY).
addition
multiplication

327 ADD, SUBTRACT, MULTIPLY, and DIVIDE statements perform only the type of arithmetic operation specified by the verb. Furthermore, MULTIPLY and DIVIDE statements operate on only two numbers; they multiply one number by another, or divide one number into another.

ADD and SUBTRACT, however, can operate on

$$\left\{\begin{array}{l}\text{only one number}\\ \text{only two numbers}\\ \text{more than two numbers}\end{array}\right\}.$$

●●●

more than two numbers

328 An operation that can be done with a COMPUTE statement, but cannot be done by any of the other arithemtic verbs is _____.

●●●

exponentiation

Exponentiation is the raising of a number to a certain power. To be sure, you can square a number (raise it to the second power) using a MULTIPLY statement -- multiplying the number by itself. But higher powers would require several MULTIPLY statements; and if the exponent varied, MULTIPLY statements would be hard to use.

329 *Another important category of procedural words is "sequence control". These words enable the programmer to control the sequence in which other statements or procedures will be acted on by the computer. There is IF, which permits different actions on the basis of a test-condition; GO TO and PERFORM, which cause branching; and STOP, which delays or halts the run.*

You should get a general idea of the function of IF by reading the summary in the reference handbook; however, we will postpone our discussion of IF until the next lesson, when we will study it in detail. We will discuss GO TO, PERFORM, and STOP at this time. Make sure that you are able to describe the difference between GO TO and PERFORM. And learn what the two types of STOP statements are, and how you would tell them apart.

Reading assignment: PROCEDURAL WORDS
IF
GO TO
PERFORM
STOP

●●●

330 A GO TO statement causes a branch to a $\left\{\begin{array}{l}\text{statement}\\ \text{sentence}\\ \text{procedure}\end{array}\right\}$.

●●●

procedure

331 A GO TO statement contains the name of the procedure to which a branch is desired. You can see that this name must be the same as the name given in the _____ entry of the procedure.

●●●

header

332 How does a GO TO differ from a PERFORM?

●●●

A PERFORM causes a branch to a procedure or series of procedures, just as a GO TO does. But after the procedure or procedures are acted on, PERFORM causes a return branch to the statement after the PERFORM statement.

333 *The flow of control through procedures, and further study of GO TO and PERFORM, will be taken up in the next lesson.*

●●●

334 There are two kinds of STOP statements. One stops the execution of the program permanently, the other temporarily. A permanent stop is indicated when the verb STOP is followed by _____.

●●●

the word RUN

335 If the stop is temporary, that is, if execution is to be resumed after the computer operator takes some corrective steps, the verb STOP is followed by _____.

●●●

a literal

336 Here is an example of STOP followed by a literal. What will happen to the literal when this stop is executed during the running of the object program?

```
    STOP 'HALT 725  SEE RUN BOOK'.
```

●●●

The literal will be typed out on the console typewriter.

337 The literal in the previous example is meant to be a

[message to the computer]
[comment in the source program listing]
[message to the computer operator].

●●●

message to the computer operator

338 *The final category of procedural words is "program comments". This is another one-word category, and deserves only brief study.*

Reading assignment: PROCEDURAL WORDS
NOTE

●●●

339 NOTE entries are used for program documentation only. The words written in a NOTE statement are

[converted into machine language statements]
[loaded into core storage as constants]
[printed in the source program listing].

●●●

printed in the source program listing ONLY

340 A NOTE

$\left\{\begin{array}{l}\text{has no size limit}\\ \text{cannot exceed the width of the source program listing form}\\ \text{can be as long as 120 characters, like all non-numeric literals}\end{array}\right\}$.

●●●

has no size limit

LESSON 9

341 *The brief summary of the procedural word IF, which you read in the previous lesson, indicated in a general way, that IF causes a condition to be tested, and causes alternate paths of action to be taken, depending on whether the description of the condition is true or false. In this lesson, we will first study the various kinds of conditions that can be tested in IF statements. There are five kinds of conditions (called "test-conditions" in COBOL jargon); you will study two of these now, the other three a little later.*

Reading assignment: TEST-CONDITIONS
 Relation test
 Sign test

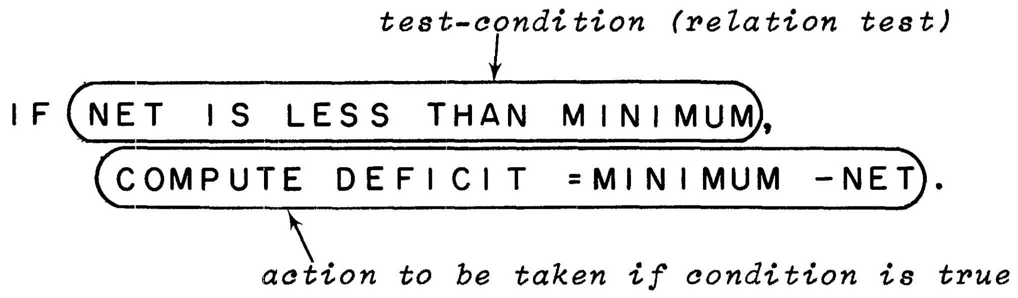
(Do not read about the condition-name test yet.)

●●●

342 In this IF sentence, pick out the test-condition and the statement to be acted on if the test-condition is true. Also, identify the test-condition.

```
       IF NET IS LESS THAN MINIMUM,
           COMPUTE DEFICIT = MINIMUM - NET.
```

●●●


test-condition (relation test)
IF NET IS LESS THAN MINIMUM,
COMPUTE DEFICIT =MINIMUM -NET.
action to be taken if condition is true

343 A relation test causes two values to be _____.

●●●

compared

344 The reason for comparing the two values in a relation test is to determine whether the <u>stated</u> relationship is true or false. In the first example below, the stated relationship is true if the values of the two data items are exactly equal, and false if either value is higher or lower than the other.

```
    IF PURCHASER = CONTROLLING-PARTY,
        MOVE 'NO CHARGE' TO EXPLANATION.
```

In the next example, the stated relationship is true if APPLICANT-AGE is [equal to] [less than] [greater than] 16; and false if APPLICANT-AGE is [equal to] [less than] [greater than]16.

```
    IF APPLICANT-AGE IS NOT GREATER THAN 16,
        MOVE APPLICATION-RECORD TO
        EXCEPTION, WRITE EXCEPTION.
```

●●●

true if EITHER equal to OR less than 16; false ONLY if greater than 16

When we take up "flow of control" later in this lesson, we will look closely at what happens when the test-condition is true or false.

345 In the sign test, ZERO is considered to be [positive] [negative].

●●●

NEITHER positive NOR negative

346 This means that an item whose value is zero

{must not have an operational sign
can have a sign, but it is ignored
has a special ± (plus-or-minus) sign}

●●●

can have a sign, but it is ignored

347 The reference handbook points out that the sign test is another way of stating a relation test that compares a number with zero. What sign test would be equivalent to the relation test below?

```
    IF TOTAL-DUE > 0, GO TO WRITE-BILL.
```

●●●

IF TOTAL-DUE IS POSITIVE, GO TO WRITE-BILL.

348 *Both the relation test and the sign test are easily comprehended. Next you will learn about three other tests, whose meaning -- at least the meaning of two of the three -- is not always obvious.*

Reading assignment: TEST-CONDITIONS (continued)
Condition-name test
Overflow test
Class test

●●●

349 The class test is used to determine whether or not an item is _____ or _____.

●●●

alphabetic, numeric

350 *A possible use for class tests is to check the validity of certain data items. For example, it might be desired to determine whether an item that is supposed to contain numeric information actually contains digits. (Such an item would have a picture identifying it as a numeric item, but there is no automatic checking to verify that data put into an item during the running of a program corresponds to the item's picture.)*

●●●

351 In a condition-name test, the test-condition consists solely of a _____.

●●●

programmer-supplied name

352 The _____ test resembles the condition-name test.

●●●

overflow

353 Since a condition-name test is another way of testing whether a data item is equal to a literal, any condition-name test could be replaced by a

$$\left\{\begin{array}{l}\text{relation test}\\ \text{sign test}\\ \text{class test}\\ \text{overflow test}\end{array}\right\}$$

●●●

relation test

354 The reason a programmer might use a condition-name test instead of a relation test is to make the program more readable. If he has done his job right, the name of the condition will explain the meaning of the condition. For instance:

```
    IF APPLICANT-WILL-TRAVEL,
        GO TO OPENINGS-ON-THE-ROAD.
```

The condition name, APPLICANT-WILL-TRAVEL, actually represents a specific value of a specific item. To find out what value, and what item, you would look in the _____ division.

●●●

Data

355 A condition name is defined in a level number _____ entry in the Data division.

●●●

88

356 Suppose the following entry appeared in the Data division:

```
    88  APPLICANT-WILL-TRAVEL   VALUE 'G'.
```

This entry gives the condition name, and the value represented by the condition name, but does not tell the name of the item. Where would you look for the item description entry to which the 88-entry applies?

●●●

Above the 88-entry. Level-88 entries follow immediately after the description of the entry to which they apply; there may, however, be more than one level-88 entry for an item.

357 Here we see the condition-name entry in context.

```
    02  ATTITUDE-TOWARD-TRAVEL  PICTURE X.
    88  APPLICANT-WILL-NOT-TRAVEL VALUE 'A'.
    88  APPLICANT-WILL-TRAVEL   VALUE 'G'.
```

Which one of the following relation tests is the equivalent of IF APPLICANT-WILL-TRAVEL...?

```
(1)  IF ATTITUDE-TOWARD-TRAVEL
         IS EQUAL TO APPLICANT-WILL-TRAVEL.
```

```
(2)  IF ATTITUDE-TOWARD-TRAVEL = 'G'.
```

●●●

(2)

358 Whenever a programmer wants to test whether the value of a data item is equal to a literal, he has the choice of using either a relation test or a condition-name test. His decision as to which to use would be based in part on the readability of the relation test. A relation test such as IF TEST-SCORE = 100... tells just as much as IF PERFECT-TEST-SCORE..., so the programmer would probably use the relation test.

On the other hand, if he had a choice of writing IF MARITAL-STATUS = 7... or IF DIVORCED..., the programmer would certainly write IF DIVORCED... In this instance, then, the

preferred test is the $\left\{\begin{matrix}\text{relation test}\\ \text{condition-name test}\end{matrix}\right\}$.

●●●

condition-name test

359 No similar choice exists when it comes to the overflow test. The overflow test is the only way of testing whether the 12-punch has been sensed in the carriage tape. (The punch in channel 12 of the carriage tape indicates when the last normal printing line of the form has been reached, in order to leave a margin of blank paper at the bottom of each form.)

This means that an overflow test will be found in most programs where there is printed output. The logic of the IF sentence containing the overflow test will nearly always be similar to the sample sentence given in the reference handbook, but of course, the programmer-supplied names will undoubtedly be different.

The programmer-supplied name that represents the form-overflow condition is defined in an APPLY entry, which will be found in the I-O-Control paragraph of the _____ division.

●●●

Environment

360 *Our discussion of test-conditions leads naturally to closer study of the flow of control through COBOL procedures. After all, the reason for having test-conditions is to permit control to flow along alternate procedural paths.*

The COBOL compiler, as you know, will cause the COBOL procedural statements to be translated into actual machine language instructions. When we talk about flow of control, we are "playing computer", so to speak, and acting as though the COBOL statements had already been translated and are now being executed by the computer. In order to trace the flow of control, you must know such things as where the starting point is, what sequence is normally followed, and what statements cause deviations from that sequence.

Reading assignment: FLOW OF CONTROL
Starting point
Sequence
Branching

(Don't study the flow of control through IF statements yet.)

●●●

361 *"Declarative" procedures are used to alter the usual actions performed by the input-output control system. One example of the use of declaratives is in the processing of user label records, that is, labels that provide file information over-and-above that provided by the standard label records.*

In this book, our only concern with declaratives is their effect on the place at which execution of a COBOL program begins. What is the starting point of program execution when there are declaratives? When there are no declaratives?

●●●

When there are declaratives, control starts at the first procedure after the declaratives. When there are no declaratives, control starts at the first procedure of the Procedure division.

362 How can you tell whether a program contains any declarative procedures? How can you tell where the declaratives end and the regular procedures begin?

●●●

If a program contains declaratives, the line after the division header will state: DECLARATIVES. And the line after the last declarative procedure will state: END DECLARATIVES.

363 Control normally flows from one statement to the next

$$\left\{\begin{array}{l}\text{in the order in which they are written in the program}\\ \text{in alphabetical order according to the first letters of}\\ \quad\text{the verbs}\\ \text{in order by verb categories, input-output being done first}\end{array}\right\}.$$

●●●

in the order in which they are written in the program

364 When control comes to the end of a procedure, it normally

$$\left\{\begin{array}{l}\text{stops temporarily, until an operator presses the Start key}\\ \text{waits for the next command from the supervisor program}\\ \text{goes right on to the next procedure in sequence}\end{array}\right\}.$$

●●●

goes right on to the next procedure in sequence

365 *The procedural words which can change the normal flow of control are the familiar quartet of "sequence control" words: GO TO, IF, PERFORM, and STOP.*

A GO TO statement causes control to branch unconditionally to the _____ of a procedure.

●●●

first statement

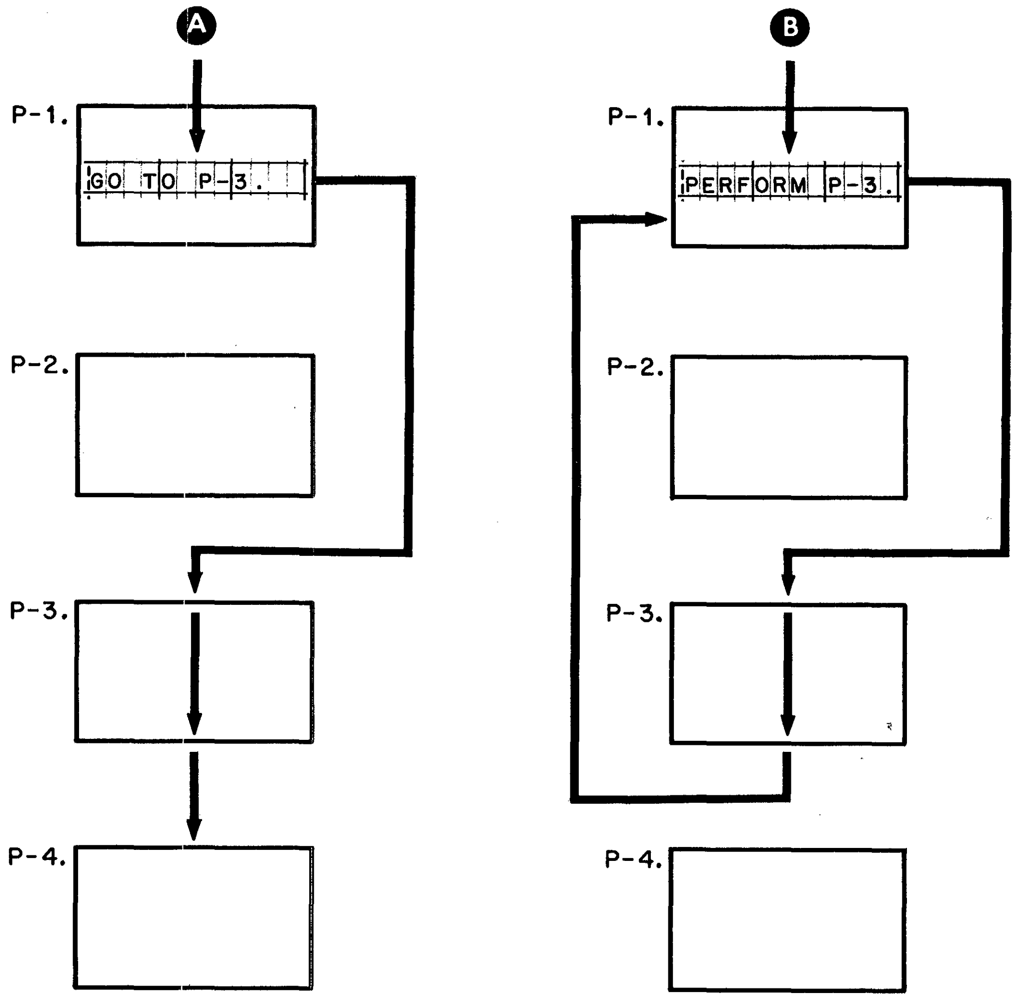
366 After a GO TO has caused control to branch to the beginning of a procedure,

$$\left\{\begin{array}{l}\text{control immediately branches back to the statement after}\\ \quad\text{the GO TO}\\ \text{the normal flow of control is resumed}\\ \text{control flows through that procedure, and then returns}\\ \quad\text{to the GO TO}\end{array}\right\}.$$

●●●

the normal flow of control is resumed

367 These drawings graphically illustrate the difference between GO TO (drawing A) and PERFORM (drawing B). Each drawing consists of four boxes that represent procedures. In each case, the flow of control begins at procedure P-1; part way through the first procedure a sequence control statement is acted on. The flow arrows remind us that a branch is caused by [GO TO] [PERFORM].



●●●

BOTH GO TO AND PERFORM

368 In both cases, control flows through procedure P-3, but at that point the difference occurs. In drawing A, control then flows _____, whereas in drawing B, control flows _____.

●●●

(A) to the next procedure in sequence, P-4
(B) back to procedure P-1, where it will next act on the statement following the PERFORM statement

369 *GO TO and PERFORM statements cause control to flow to the beginning of a procedure, so the programmer must think in terms of procedures when using these statements. IF sentences, on the other hand, cause control either to flow through or to jump over certain statements. Keep this difference in mind as you examine the logic diagrams for IF statements, in the reference handbook.*

Reading assignment: FLOW OF CONTROL (continued)
Flow of control through an IF sentence that does not contain ELSE or OTHERWISE
Flow of control through an IF sentence that contains ELSE or OTHERWISE

●●●

370 The logic diagrams you have just looked at ought to help you in determining what processing is accomplished in an IF sentence. The first step indicated in both diagrams is that the "data is evaluated", to find out whether the description of the data condition is true or false.

Of the five test-conditions that can be used in IF sentences, one test-condition does not, strictly speaking, describe a data condition. Which one?

●●●

overflow test

The logic diagrams apply to the overflow test, too, except that the words "data is evaluated" are not appropriate. For that test, it would be more appropriate to say something like "status of electronic indicator in printer is tested". However, I am sure we can get by with the diagrams as they stand; and might say that the "status of an indicator" constitutes a piece of information, and that it is therefore "data" -- in the larger sense of the word.

371 It is also important to emphasize the difference between a "data condition", which is a matter of fact, and a "description of a data condition", which might be true or false at any given time. It is certainly not possible to tell whether a test-condition, such as STOCK-LEVEL IS LESS THAN MINIMUM-BALANCE, is true or false merely by looking at the COBOL entry. It is necessary to look at the data itself; and if we find that the value of STOCK-LEVEL is 355, and that the value of MINIMUM-BALANCE is 352, we can then say that the test-condition is $\left\{\begin{matrix}\text{true}\\\text{false}\end{matrix}\right\}$ at this time.

●●●

false

372 When an IF sentence does not contain ELSE or OTHERWISE, control jumps to the next $\left\{\begin{matrix}\text{statement}\\ \text{sentence}\end{matrix}\right\}$ if the test-condition is false.

●●●

sentence

373 The end of a sentence is identified by a _____.

●●●

period

374 It should be clear from the logic diagrams how important the distinction between a "sentence" and a "statement" now becomes. Previously, we have dealt with "unconditional" actions, where the distinction was not so important. For example, the first sentence below calls for exactly the same actions as the two sentences below it.

```
ADD PENALTY TO AMOUNT-DUE, MOVE
    AMOUNT-DUE TO PAYABLE.

ADD PENALTY TO AMOUNT-DUE.
MOVE AMOUNT-DUE TO PAYABLE.
```

But where the word IF is involved, the exact location of periods is vital. Thus, the two sets of entries below do not mean the same thing. Can you explain the difference between them?

```
IF SALES < QUOTA, MOVE 'BELOW' TO MEMO.
ADD SALES TO YEAR-TO-DATE-SALES.

IF SALES < QUOTA, MOVE 'BELOW' TO MEMO,
ADD SALES TO YEAR-TO-DATE-SALES.
```

●●●

In the first set of entries, SALES will be added to YEAR-TO-DATE-SALES whether or not the value of SALES is less than the value of QUOTA. In the second set, SALES will be added to YEAR-TO-DATE-SALES only if the value of SALES is less than QUOTA.

375 What is wrong with this reasoning:

The two sets of entries below accomplish exactly the same purpose. In each one, the EMPLOYEE-PLAN procedure will be performed if the value of AGE is less than 65; and RETIREE-PLAN will be performed if AGE is equal to or greater than 65.

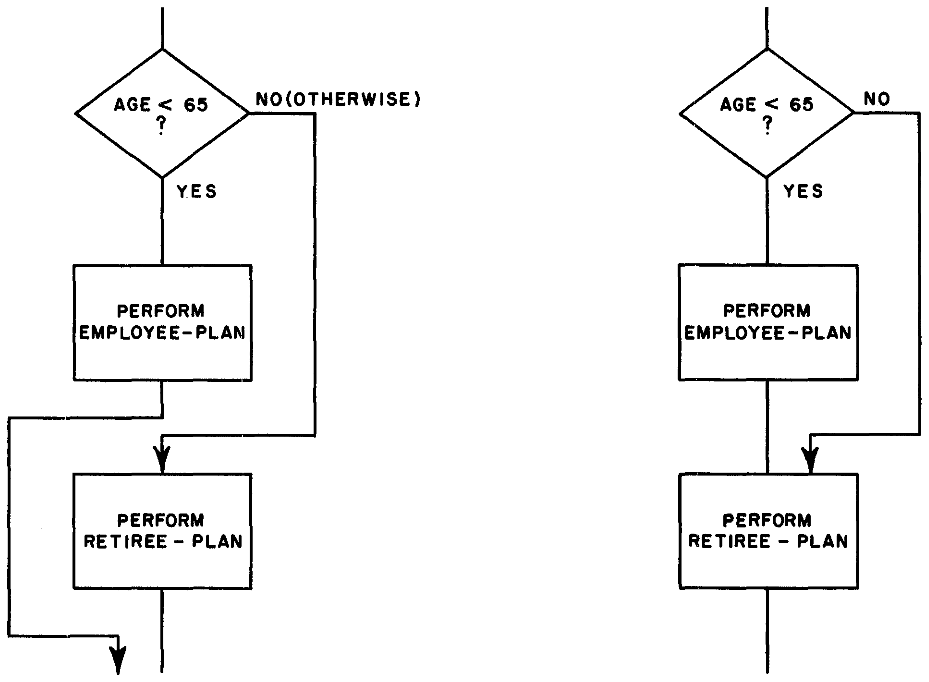
```
    IF AGE < 65, PERFORM EMPLOYEE-PLAN;
    OTHERWISE, PERFORM RETIREE-PLAN.
```

```
    IF AGE < 65, PERFORM EMPLOYEE-PLAN.
    PERFORM RETIREE-PLAN.
```

●●●

In the second set, RETIREE-PLAN will be performed no matter what the value of AGE is.

376 *The program-flowchart excerpts below correspond to the two sets of COBOL entries that you studied in the previous frame. They illustrate the difference between the two sets of entries. The difference is precisely the same as the difference between the two logic diagrams in the reference handbook; you might wish to compare those general logic diagrams with these specific application flowcharts.*



●●●

377 *The flowcharts in the preceding frame show how a programmer thinks about the verb PERFORM. He treats PERFORM as if it were a process done "in line" within a procedure. Of course, the programmer is well aware -- just as you are aware -- that PERFORM actually causes control to branch off to some other procedure, and then causes control to come right back. This linkage to and from the other procedure is completely taken care of by the COBOL compiler, which permits the programmer to take a simplified view of things.*

But, when you are analyzing a program that someone else has written, to see just what the program does, you must locate the procedure that is performed and examine the processing done in that procedure. We are going to do just that next; we will trace the flow of control through a portion of the sample Procedure division in the reference handbook.

Turn to the sample Procedure division -- the one you looked at when you began detailed study of the division, not the case study program. Locate the IF sentence in the READ-NEXT-CARD paragraph.

●●●

378 *This frame, and the next seven frames, refer to the sample Procedure division.*

The IF statement states that TOTAL-ROUTINE is to be performed IF _____.

●●●

NUMBER OF COMMODITY IS NOT EQUAL TO OLD-NUMBER

379 TOTAL-ROUTINE is the programmer-supplied name of a _____.

●●●

procedure (paragraph)

380 TOTAL-ROUTINE is the next procedure in sequence following READ-NEXT-CARD.

A procedure to be performed $\left\{\begin{matrix}\text{must follow}\\ \text{need not follow}\end{matrix}\right\}$ the procedure that contains the PERFORM statement.

●●●

need not follow

A little while ago, you studied a drawing that showed a PERFORM statement in procedure P-1. The procedure to be performed did not follow P-1; instead, it was procedure P-3, two procedures away. And it would have been perfectly all right for a PERFORM statement in procedure P-4 to say, "PERFORM P-1".

381 TOTAL-ROUTINE consists of some MOVE statements and a WRITE statement. It does not contain a GO TO statement. How will control be returned to the statement following the PERFORM statement in the READ-NEXT-CARD procedure?

●●●

Instructions to cause control to return will be generated by the COBOL compiler.

382 To what statement will control return after TOTAL-ROUTINE has been performed?

●●●

GO TO DETAIL-PROCESSING.

383 DETAIL-PROCESSING is the name of a _____.

●●●

procedure

384 Thus, immediately after control returns from TOTAL-ROUTINE, it is sent off the DETAIL-PROCESSING. This would seem to suggest, why not add a last statement to TOTAL-ROUTINE that says, "GO TO DETAIL-PROCESSING?" The main reason that this was not done is because the TOTAL-ROUTINE is to be performed at another point in the program, and at that point, control is <u>not</u> to go to DETAIL-PROCESSING.

Find the other "PERFORM TOTAL-ROUTINE" statement in the program. To what statement will control return following the execution of <u>that</u> PERFORM?

●●●

The other "PERFORM TOTAL-ROUTINE" statement is in the END-OF-JOB procedure. After it is executed, control will go to the statement that reads, "CLOSE PURCHASING-FILE, PURCHASE-REPORT-FILE".

385 Let's continue to trace the flow of control from the CLOSE statement. After the CLOSE statement is executed, control goes to a DISPLAY statement, and then to a STOP statement.

1. Each of these COBOL statements also happens to be a sentence. Does this fact alter the flow of control in any way?

2. Exactly <u>what</u> will happen when the DISPLAY statement is executed?

3. Where will control go following the execution of the STOP statement?

●●●

1. No, not in this case; but as you have seen earlier, the distinction between statements and sentences is crucial when you are dealing with IF sentences.

2. The words PURCHASE REPORT FINISHED will be typed on the console typewriter.

3. Control will not go anywhere, as far as this program is concerned. STOP RUN means the job is finished. *(Incidentally, this does not mean that the computer stops running; control is simply turned over to the operating system control program, which will probably load the next program and process the next job.)*

LESSON 10

386 *This final lesson will give you a chance to find out whether you have learned what you were expected to learn from the other lessons, which is to read a COBOL program with a high degree of comprehension. You can think of this lesson as a "self test", inasmuch as you will be applying what you have previously learned, rather than learning new information.*

Although there are no reading assignments in this lesson, you should feel free to look up information in the reference handbook whenever you need to. We have covered a lot of ground in nine lessons, and you were repeatedly urged not to memorize details, so it would hardly be fair to expect you to rememoer any but the most important facts! (You may be surprised to find how many of the details have "stuck with you".) Anyway, the answers are what count, not whether you have memorized them, or must look them up.

●●●

387 *Most of our work in this lesson will be done with the case study program. Turn to the Procedure division of the case study program, and locate the third procedure.*

The third procedure is $\left\{\begin{array}{l}\text{READ-AND-CHECK-RECORD}\\ \text{LINE-1-PROCEDURE}\end{array}\right\}$.

●●●

LINE-1-PROCEDURE

388 The first sentence of that procedure is

$\left\{\begin{array}{l}\text{LINE-1-PROCEDURE.}\\ \text{MOVE SPACES TO BILL-LINE-1.}\end{array}\right\}$

●●●

MOVE SPACES TO BILL-LINE-1.

389 If "LINE-1-PROCEDURE." is not a sentence, what is it?

●●●

It is a paragraph header entry (procedure header entry).

390 "MOVE SPACES TO BILL-LINE-1." is the first sentence of its procedure. What is the first statement of that procedure?

●●●

The sentence contains just one statement, "MOVE SPACES TO BILL-LINE-1". *(The sentence is terminated by a period, while the statement is not; so the period is the only difference between the first sentence and the first statement, in this case.)*

391 SPACES is one of a few reserved words that are called _____.

●●●

figurative constants

392 BILL-LINE-1 is a programmer-supplied name of a data item. In which division would you look to find the description of the item?

●●●

Data division

393 Find the item description entry for BILL-LINE-1. The level number of that entry tells that BILL-LINE-1 is the name of a _____.

●●●

record

394 The level-01 entry for BILL-LINE-1, plus the string of level-02 entries that follow it, together make up a _____.

●●●

record description

395 BILL-LINE-1 is a record, and it is also $\left\{\begin{array}{l}\text{a group item}\\\text{an elementary item}\\\text{an independent item}\end{array}\right\}$.

●●●

a group item

396 How many elementary items are there in the BILL-LINE-1 record?

●●●

thirteen -- Every level-02 item in this record is an elementary item.

397 There are no USAGE clauses in any of the entries in the BILL-LINE-1 record description. Do the items in this record have a usage? If so, what is their usage?

●●●

Yes -- all data items have a usage, since "usage" is the COBOL term that means the data code in which data will be stored in an item.

The usage of the items in this record is DISPLAY; when no usage is specified, it is assumed to be DISPLAY.

398 Seven of the items in this record are called FILLER. What is "FILLER"?

●●●

FILLER is a reserved word that can be used in place of a name when the item is not going to contain any information or is not going to be processed.

399 The picture of the item named CREDITS is $\left\{\begin{array}{l}\text{PICTURE}\\ \text{PICTURE \$\$,\$\$\$.\$\$}\\ \text{\$\$,\$\$\$.\$\$}\\ \text{\$\$,\$\$\$.\$\$.}\end{array}\right\}$.

●●●

$$,$$$.$$ (The final period is not part of the picture; it is the period that terminates the entry.)

400 The picture identifies CREDITS as a _____ item.

●●●

report (that is, an item used to store edited data)

401 What is the name of the file in which BILL-LINE-1 is a record?

●●●

CUSTOMER-BILL-FILE

402 What does "FD" stand for?

●●●

File Description

403 The FD entry for CUSTOMER-BILL-FILE contains the following clause: "RECORDING MODE F". What does "recording mode" mean? What does "F" stand for?

●●●

"Recording mode" is the COBOL term for the format (layout) of records. "F" stands for "Fixed length".

404 How many different kinds of data records are there in CUSTOMER-BILL-FILE?

●●●

four -- BILL-LINE-1, BILL-LINE-2, BILL-LINE-2, and BILL-LINE-4

405 Where would you look to find out what device CUSTOMER-BILL-FILE is assigned to?

●●●

In the Environment division

406 Locate the Environment division. CUSTOMER-BILL-FILE is assigned to device number _____.

●●●

1403 (the IBM 1403 Printer)

407 The Environment division also indicates that the object program will be compiled and executed on $\left\{\begin{array}{l}\text{the same}\\\text{different}\end{array}\right\}$ computer models.

●●●

different (Source computer is IBM-360 G50; object computer is IBM-360 F30.)

408 The name of this COBOL program is $\left\{\begin{array}{l}\text{BILLFILE}\\\text{CUSTBILL}\\\text{BILLING}\\\text{not specified}\end{array}\right\}$.

●●●

BILLING (See the Program-Id paragraph of the Identification division.)

409 The Data division is the $\left\{\begin{array}{l}\text{first}\\\text{second}\\\text{third}\\\text{fourth}\end{array}\right\}$ division of the program.

●●●

third

410 Turn to the Data division once again. Where would you look to find the descriptions of independent items?

●●●

In the Working-Storage section

411 Locate the Working-Storage section. Of the three items described in the section, the [first] [second] [third] is an independent item.

●●●

ALL THREE are independent items -- all have level number 77.

412 How many characters does the item named SKIP-TO-CARRIAGE-CHANNEL-1 contain?

●●●

one (Its picture is X.)

413 Just above the Working-Storage section header is the record description of the BILL-LINE-4 record. How many characters does that record contain?

●●●

133 (40 + 22 + 71)

414 Does this mean that each of the other three records in the CUSTOMER-BILL-FILE also contains 133 characters?

●●●

Yes, it does -- in this instance. You previously observed that the recording mode of the file is F, which means that all of the records in the file have the same length.

415 Turn to the Procedure division again. Where will the flow of control through the procedures of this program begin?

●●●

At the START-PROCESSING procedure (There are no declaratives in this program.)

416 In a few words, what is the function of the OPEN statement in the START-PROCESSING procedure? Also, what does the OPEN statement tell you about BILLING-FILE and CUSTOMER-BILL-FILE?

●●●

The OPEN statement makes the input and output files ready for reading and writing. It also tells us that BILLING-FILE is the input file, and that CUSTOMER-BILL-FILE is the output file.

417 Where will control flow when it comes to the end of the START-PROCESSING procedure?

●●●

To the next procedure in sequence: READ-AND-CHECK-RECORD

418 How many statements are in the first sentence of the READ-AND-CHECK-RECORD procedure?

●●●

two: READ BILLING-FILE and GO TO END-OF-RUN

419 BILLING-FILE is the name of a _____, and END-OF-RUN is the name of a _____.

●●●

file; procedure

420 Under what condition will control branch to END-OF-RUN?

●●●

When the end of the file has been reached; that is, when all of the data records have been processed.

(AT END is a clause of the READ statement, and is mentioned in the summary of the READ verb, under PROCEDURAL WORDS in the reference handbook.)

421 What actions are taken when control branches to the END-OF-RUN procedure?

●●●

The input and output files are closed, and the run is stopped.

R29-0205-0

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