



Data Processing Systems Basic Concepts

Programmed Instruction

How To Use This Book

This is an instructional book that is more comparable to a series of lectures than it is to a “textbook.” (This book is *not* intended for use as a reference manual.) Study Chapter 1 first, then study Chapter 2, then Chapter 3, and so on. Do not skip sections or read chapters out of sequence.

To gain maximum benefit from this book, study *each chapter* in the following way:

Step A. Take pretest.

1. If you miss one or more pretest questions, continue by going to *Step B*.
2. If you answer *all* of the pretest questions correctly, skip the rest of the chapter and go to the pretest for the next chapter.

Step B. Read descriptive material.

Step C. Take post-test.

1. If you miss one or more post-test questions, continue by going to *Step D*.
2. If you answer *all* of the post-test questions correctly, skip the rest of the chapter and go to the pretest for the next chapter.

Step D. Read the programmed instruction sequence. After you finish, start at *Step A* for the next chapter.

Note: You may decide, after you have studied one or two chapters, to skip taking the pretest until after you have studied the material in a chapter. For example, you may first read the descriptive material and then take the post test. If you answer *all* post-test questions correctly, you can then skip to the next chapter. If you miss one or more post-test questions, however, read the programmed instruction sequence. Then you can take the pretest to assure yourself that you have learned the material.

First Edition (March 1973)

This publication is for IBM Internal Use Only. Comments concerning its contents can be sent to Systems Publications, Department 27T, Building 032-2, Boca Raton.

© Copyright International Business Machines Corporation 1973

CONTENTS

Introduction	v
Chapter 1. Basic Data Flow	1-1
Pretest 1	1-1
Descriptive Material for Chapter 1. Basic Data Flow	1-5
Post-Test 1	1-10
Programmed Instruction Sequence for Chapter 1. Basic Data Flow	1-13
Chapter 2. System Overview	2-1
Pretest 2	2-1
Descriptive Material for Chapter 2. System Overview	2-5
Post-Test 2	2-9
Programmed Instruction Sequence For Chapter 2. System Overview	2-13
Chapter 3. The Punched Card Medium	3-1
Pretest 3	3-1
Descriptive Material for Chapter 3. The Punched Card Medium	3-5
Post-Test 3	3-13
Programmed Instruction Sequence for Chapter 3. The Punched Card Medium	3-17
Chapter 4. Data Organization; Terminal and Decision Flowchart Symbols	4-1
Pretest 4	4-1
Descriptive Material For Chapter 4. Data Organization; Terminal and Decision Flowchart Symbols	4-3
Post-Test 4	4-11
Programmed Instruction Sequence for Chapter 4. Data Organization; Terminal and Decision Flowchart Symbols	4-13
Chapter 5. Job Overview: From Planning to Output Data; Programming Languages	5-1
Pretest 5	5-1
Descriptive Material for Chapter 5. Job Overview: From Planning to Output Data; Programming Lanugages.	5-5
Post-Test 5	5-11
Programmed Instruction Sequence for Chapter 5. Job Overview: From Planning to Output Data; Programming Languages	5-13
Chapter 6. Documentation	6-1
Pretest 6	6-1
Descriptive Material for Chapter 6. Documentation	6-3
Post-Test 6	6-9
Programmed Instruction Sequence for Chapter 6. Documentation	6-11

Chapter 7. Arranging Records	7-1
Pretest 7	7-1
Descriptive Material for Chapter 7. Arranging Records	7-3
Post-Test 7	7-9
Programmed Instruction Sequence for Chapter 7. Arranging Records	7-11
Chapter 8. File Organization and the Magnetic Tape I/O Medium	8-1
Pretest 8	8-1
Descriptive Material for Chapter 8. File Organization and the Magnetic Tape I/O Medium	8-5
Post-Test 8	8-17
Programmed Instruction Sequence for Chapter 8. File Organization and the Magnetic Tape I/O Medium	8-21
Chapter 9. File Organization and Direct Access Storage Devices (DASD)	9-1
Pretest 9	9-1
Descriptive Material for Chapter 9. File Organization and Direct Access Storage Devices (DASD)	9-5
Post-Test 9	9-15
Programmed Instruction Sequence for Chapter 9. File Organization and Direct Access Storage Devices (DASD)	9-19
Chapter 10. Output Printing; Levels of Control; Auditing	10-1
Pretest 10	10-1
Descriptive Material for Chapter 10. Output Printing; Levels of Control; Auditing	10-5
Post-Test 10	10-16
Programmed Instruction Sequence for Chapter 10. Output Printing; Level of Control; Auditing	10-19

Purpose of This Course

This course, which is designed to save you time in learning about basic data processing system concepts, is a general introduction to basic data processing terminology and concepts.

Intention of This Course

Upon completion of this course, you should be able to:

- Produce a flowcharted plan that can be followed to solve an elementary problem when given a description of that problem.
- Match the basic elements (CPU, main storage, I/O units) of a data processing system to descriptions of their basic functions in the system.
- Recognize relationships among the terms in data organization (such as field, data item, record, data file, sequential organization).
- Organize the basic operations that must be performed (from planning a solution to a problem to obtaining the solution by means of a data processing system) in the sequence in which such operations are performed.
- Match particular input and output devices (such as magnetic tape, punched card, and DASD) to file-organization schemes appropriate to those devices.
- Match the basic types of documents required in a data processing department to the points in the problem-solving process when these documents are produced.

How To Use This Book

Summary information about how to use this book is located inside the front cover.

Each of the ten chapters of this book is made up of:

1. A *pretest* that covers the material you are expected to learn as a result of your studying the chapter
2. *Descriptive material* that is a concise narrative covering the subject matter you are expected to learn in the chapter
3. A *post-test* that covers the material of the chapter in the same way as the pretest
4. A *programmed instruction (P.I.) sequence* that allows you to study, a step-at-a-time, the same subject matter that was presented in the descriptive material

You need not take a pretest before you read the descriptive material for any chapter. However, if you take and answer correctly all pretest questions for a chapter, you can feel confident in skipping the rest of that chapter. The primary purpose of each pretest is to save you time if you already know the material in a chapter. If you miss one or more pretest questions, you should read the descriptive material.

Alternately, if you wish, you can skip the pretest and start the chapter by reading the descriptive material. (The pretests are not meant to teach; they test what you are expected to do.)

The post-test, after the descriptive material in each chapter, tests what you are expected to be able to do in the same way as does the pretest in the same chapter. If you miss one or more of the post-test questions, read the programmed instruction sequence that follows the post-test. If you answer all post-test questions correctly, you can go directly to the next chapter without reading the programmed instruction sequence.

You will probably, after having read one or two chapters, decide on a procedure for study that best suits you. Several points, however, that you should note in order to gain maximum benefit from this book are:

1. Be sure to read the descriptive material in a chapter *before* you read the programmed instruction sequence for that same chapter.
2. Do not skip chapters or read later chapters before earlier chapters.
3. Do not attempt to use this book as a reference manual. It is designed to be an instruction manual only.

How to Read the Programmed Instruction (P.I.) Sequences

Suggestion: Do not read the following material until you are ready to read your first programmed instruction sequence.

1. The first thing you need is a cover card or piece of paper to mask out the correct answer to each frame. (A frame is the printed information between two sets of three dots, ● ● ●. The answer to the frame is below the lower set of dots.) Now, please obtain such a cover card and place it over this page so that you can read the following sentence. Next, move the card down the page until you uncover three dots on the left. Finally, move the card down to the next set of dots on this page.

● ● ●

2. As you move the card down the page, you uncover new material. Now move the card to the next set of dots.

● ● ●

3. Whenever you are asked a question, answer it to yourself or use a piece of scratch paper. Do not write in this book.

Question: "How do you uncover each frame in succession?" (When you have answered to yourself and are ready to check your answer, move the card down to the next set of dots and check your answer.)

● ● ●

Answer: Each frame is uncovered by moving the card from one set of dots to the next. (If you had the same idea, but your answer was worded differently, you were correct.)

4. Sometimes a frame contains a statement with a blank line in it. Your answer is the word, phrase, number, or symbol that, if put in place of the line, completes the answer. Remember, just think the answer, don't write in the book. Example: A blank line in a sentence stands for a _____ that completes the sentence.

● ● ●

word, phrase, number or symbol

5. A frame may contain more than one blank line. You should think of an answer for each blank and then move the card down to check yourself. On occasion, a frame requires no response at all. In this case, merely move on to the next frame.

● ● ●

6. When a frame contains two choices, your answer could be one of four possibilities: Item a, or item b, or BOTH items, or NEITHER item. (The answer EITHER, which you may have seen in similar texts, is NOT USED in this book.) Example:

The sum of six and five is:

- a. 30
- b. 1

• • •

Neither (the sum is eleven)

- 7. While reading the text you will be directed to a “preceding” or “following” figure. Examine the figure that is immediately above or below the text and then respond to the question in the frame.
• • •
- 8. When a frame asks you to “construct a flowchart. . .” make a drawing on scratch paper. You may simply sketch the flowchart. It should contain the properly shaped symbols but need not be precisely drawn.
• • •
- 9. A list of flowcharting symbols is found on the inside of the back cover of this text. You may refer to it whenever you sketch your flowcharts.
• • •
- 10. When you are asked to “match the lists” two lists will immediately follow. Match the items in the rightmost list with those in the leftmost list. For example, match the two lists:

a. At top left corner of frame	1. Three dots
b. Follows frame	2. Frame number

• • •
 - a. 2. (The frame number is to the left of a frame.)
 - b. 1. (Three dots follow the frame.)

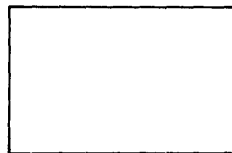
A summary of how to use this book is on the inside of the front cover. Please read that summary before proceeding.

PRETEST 1

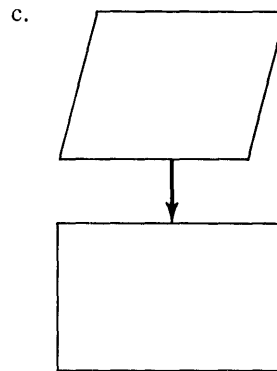
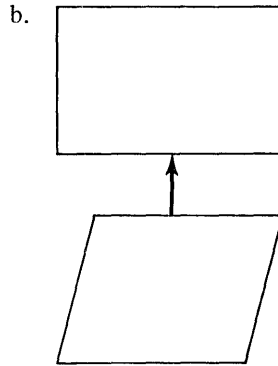
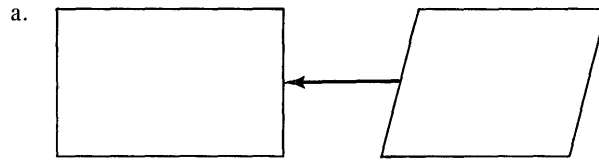
Write your answers on a separate sheet of paper. Do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. A process:
 - a. Is a series of operations that produces an end result
 - b. In order to be started, requires output
 - c. Is a mathematical ratio
 - d. Is what occurs when conditions change
 - e. I don't know
2. Another word that means basically the same thing as the word *data* is:
 - a. Medium
 - b. Information
 - c. Mathetics
 - d. None of the above
 - e. I don't know
3. Draw a flowchart that represents the basic data flow in a data processing system. Be sure to write the appropriate word(s) in the flowchart symbols. (If you don't know how to do this, go on to the next question.)
4. Write each of the following descriptions in the appropriate symbol shown below. (Do *not* draw a flowchart; merely write the description in the appropriate symbol. Note, you must draw three symbols, with a description in each.)
 - a. Read pay report.
 - b. Compute deductions.
 - c. Print paycheck.



5. The preferred direction of data flow is shown by:



- d. None of the above
- e. I don't know

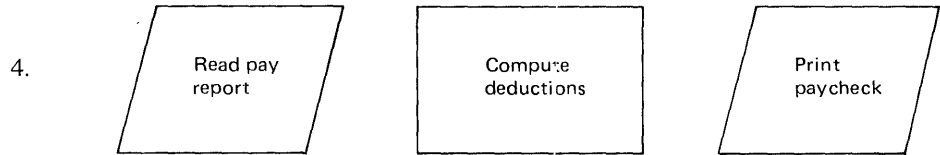
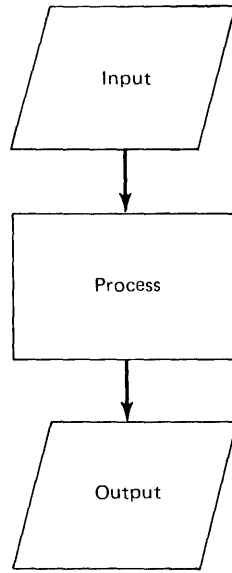
6. Flowchart the following problem statement:

A gas company bill is printed and mailed to a customer every three months. Billing amount is calculated by multiplying two and one-half cents by the number of 100-cubic-foot units of gas used. The meter report sheet is read to find the amount of 100-cubic-foot units of gas used by the customer.

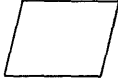

Answers are on the next page.

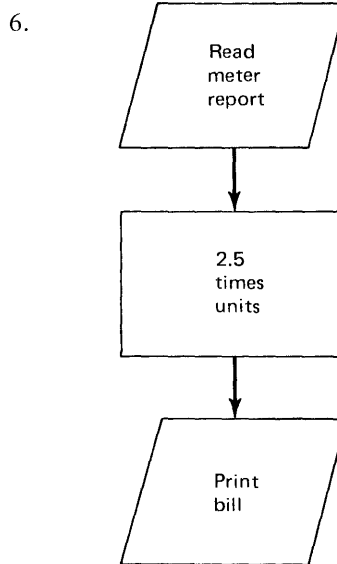
Answers to Pretest 1

- 1. a
- 2. b
- 3.



- 5. c

Note: The symbol  is used instead of the symbol  to represent input/output throughout this book.



Your wording can be different than that shown in this flowchart, but your answer should show the same basic operations.

DESCRIPTIVE MATERIAL FOR CHAPTER 1. BASIC DATA FLOW

Introduction

Today, we live in an ocean of information, information that is as much a part of our world as the air around us. How we interact with this information determines, in many respects, the degree of success that we achieve in living. We should, for example, purchase the family's groceries only after comparing various food prices. If we do not, our weekly food budget will be unfavorably affected. In making these comparisons, we use such information as prices (dollars and cents), weights (pounds and ounces), and volumes (quarts and pints).

Every year many of us spend many hours filling out our federal income tax returns. Whether we mail them early or just make the deadline, we have to interact with a good deal of information that is related to our income for the past year, especially if we elect to itemize our deductions.

Such examples of information-handling (comparison grocery shopping and income tax return preparation) are activities that many of us would not associate with computers. But computers, or data processing systems as we shall call them in this book, handle information in much the same way as you do when you compare grocery prices or compute your income tax. Data processing systems, then, are tools that man uses to help him solve problems. The variety of problems to which this tool can be applied is virtually without end.

We as individuals must make such decisions related to information. Similarly, today's business, governmental, and scientific enterprises require high-speed, accurate handling of vast amounts of information. In business, for example, hundreds of thousands of stock transactions occur in major stock exchanges every day. Each of these transactions requires computing of dollar amounts for brokerage commissions, cost of stock, and taxes. In other areas of the business world, literally millions of printed bills must be produced every month of the year in order to account for sales of services and goods.

Efficient operation of our state and federal governments demands accurate handling of an almost unbelievable amount of information that is related to tax structures, defense programs, and so on.

In scientific fields, men are dealing more and more with problems of the very small and the very large. Here, extensive manipulations of information are required to produce accurate, timely answers. In space travel, for example, extremely precise navigational information must be computed, and related decisions must be made in fractions of a second. The margin of error must be small indeed when astronauts' lives and costly space vehicles depend upon precise answers—answers that can be computed and made available for use only within a very limited time.

Without data processing systems, most of the activities just described would be next to impossible to carry out. They would be impossible at least on a scale dictated by the enormous magnitude of today's undertakings.

On the other hand, you should realize that information-handling is certainly not restricted to large concerns only. Perhaps a majority of the data processing machines in existence today are used by very much smaller organizations than those just mentioned. Primary and secondary school systems use data processing equipment for test grading, payroll-check preparation, and numerous other administrative functions. Many small businesses utilize data processing equipment for such applications as billing, sales analysis, accounts receivable, payroll, and so on.

The basic reasons for such widespread use of data processing equipment are related to the need for consistency in the production of very accurate, timely reports. The demands of competition in today's industrialized society do not allow a business to have inaccurate or leisurely billing procedures.

Whether they are used for small or for large undertakings, however, data processing systems have already freed many men from a host of tiresome tasks, tasks that are necessary in the handling of information. Imagine the human hours that would be required to calculate and then handprint, or even typewrite for that matter, the payroll checks for the thousands upon thousands of federal government employees, every month of the year! The bulk of such drudgery can be and is now performed by machines.

Data processing systems, then, are machines that help men to solve problems, problems similar to those just mentioned. Nevertheless, with all their capability for carrying out the solution to problems, such machines, you should realize, can do only what they are told to do. And, they must be told very accurately, in minute detail, exactly what steps they are to follow in their handling of any particular type of information. Also, the information they are to handle must be organized in very specific ways. In fact, the work involved in defining various problems and in specifying how such problems are to be solved by data processing systems has led to the development of many skills and professions. These skills and professions did not exist before the late 1950's. Systems designers, systems analysts, coders, programmers, data processing system installation planners. . . all of these job titles relate to tasks whose basic elements are even today little understood by a majority of the people in the world.

But what are the basic machines in the cluster of machines that is called a data processing system? How do people go about specifying a solution to a problem so that that problem can be solved by a data processing system? And what are the ways that information must be organized before it can be handled by a data processing system?

We hope that you will soon be able to answer such questions. You will not become an expert in the field of data processing as a result of your reading this book. But you will be exposed to ways of thinking about information and its handling by data processing systems, that form the basis of all data processing activities.

Data processing system is the term used in this book for a system of machines that handles information and produces meaningful results, provided that the system is carefully instructed how to do so. The term *computer* is frequently used to describe the same kind of system of machines. Another term that is used is *information processing system*. All of these terms are useful. But you should notice that a data processing system is a lot more than a group of machines. Such a system also includes:

1. The many instructions, which people have formulated, that tell the machines how to handle information
2. The information itself
3. Most importantly, both those people who specify how the machines are to handle information and those who operate the machines.

In this book, the activities of the programmer—the person who determines how a data processing system handles information for any particular problem—are stressed. We also look at the basic machines in a data processing system, and how information is organized so that these machines can handle it.

Basic Data Flow

A *process* is a series of operations that produces some end result. It is convenient to think of a process as being made up of three basic parts:

1. Getting input
2. Processing the input
3. Producing output

People set up processes in order to solve problems. The end result of a process is the solution or answer (which we call output) to the problem.

Steel, for example, is the end result (output) of the process of manufacturing steel. The problem of producing the steel is solved by getting iron ore (input) and processing that ore.

But in this book, we are interested in data processing. *Data* is information or factual material. For example, your age is data or information. The words you are reading and the time of day when you are reading are both examples of data.

Data processing, which is similar to other processes, is made up of three basic steps:

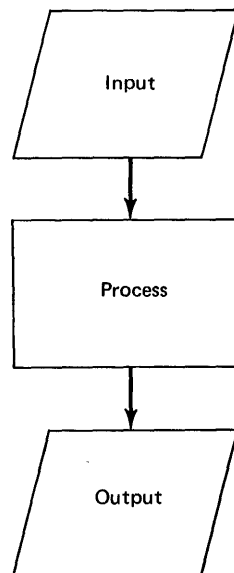
1. Getting input data
2. Performing a series of operations on that data (or processing the data)
3. Producing output data

In other words, data processing involves the solving of problems that require data for input and produce data as output.



People, who are called programmers, plan the steps that are required to solve such problems. (Sometimes, people who perform the tasks that are described in this book have other job titles such as: systems programmer, coder, systems analyst. Such job-title discriminations are not made in this book.) The problem in the form of a problem statement (or problem description), which is a definition of the problem, is given to the programmer. Frequently, the programmer must seek out further information about the problem. He forms a plan that can be used to solve the problem. Making such a plan may take days, weeks, months or longer, depending upon the complexity of the problem, how well the problem is defined, and the number of available programmers.

The programmer provides flowcharts of his plan. A flowchart is a document that shows the operations of a process and the order in which the operations should be performed. In other words, the flowchart represents the plan or sequence of operations that must be performed in order to produce an answer to the problem.

The basic plan that must be followed in the solution of any data processing problem is the same as the basic data flow in a data processing system (which is often called a computer). A simple flowchart that shows this basic data flow is:



Notice in this flowchart:

1. The direction of data flow is indicated by lined arrows (↓) and is from the top to the bottom of the flowchart. This is the preferred direction. When the top-to-bottom flow is not convenient, preferred direction is left-to-right.
2. The symbol  represents either input or output, depending upon the name or label written in that symbol.
3. The symbol  represents processing operations other than getting input or producing output.

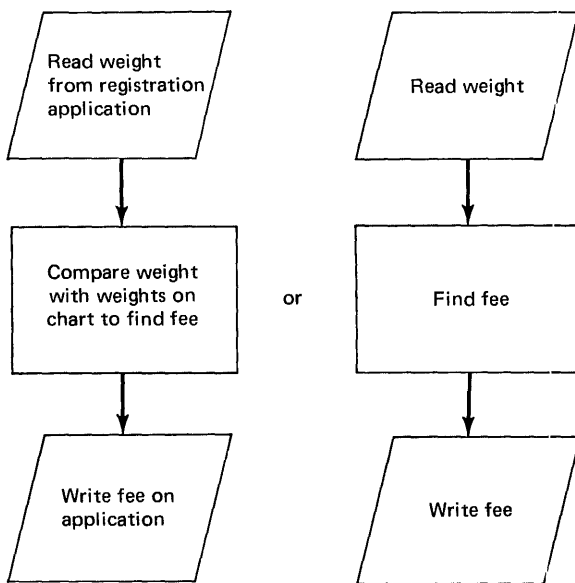
Flowcharts for simple problem statements can be constructed by use of these basic flowchart symbols. Consider, for example, the following problem statement:

The automobile registration fee is based on the weight of the automobile. The weight can be read from the registration application form. A chart, which contains weight ranges and corresponding fees, is used to find the fee for a given automobile weight. After the fee is determined, it is written on the registration application form.

Reorganizing this statement into input, process, and output steps results in:

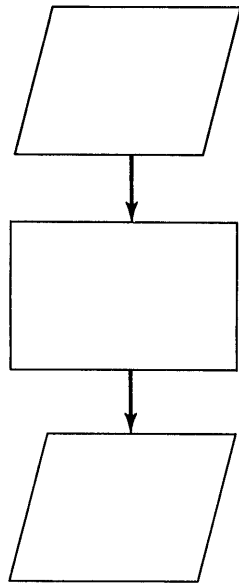
Input Read weight from registration application.
Process Find fee by comparing input weight with weight listed on chart.
Output Write fee on registration application.

A more easily understood plan of solution has been developed because of this simple reorganization. The plan can be documented in flowchart form, for example, as follows:



While the preceding flowchart on the left provides more information, you should notice that the operations and the sequence in which they are performed are shown by either flowchart.

Note that various symbols are used in producing flowcharts. Also, the following “pattern” will *not* be observable in all flowcharts:



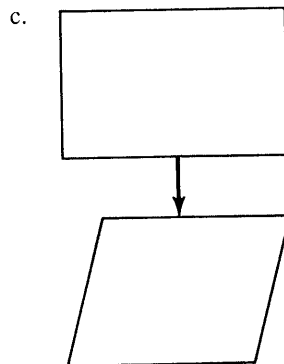
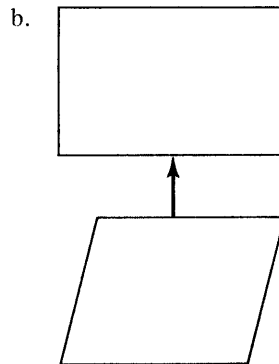
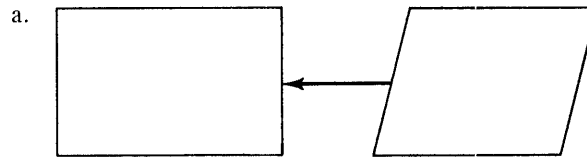
As a matter of fact, some flowcharts you will draw in this course will not contain the symbol , even though processing steps are represented in the flowchart. In other words, do not falsely assume that all flowcharts follow exactly the “pattern” described in this chapter. (The reason for sometimes not using the symbol, which is part of the basic data flow, is related to the type of operation that it is desirable to show in the flowchart being drawn. Later in this course it will become clear to you why the symbol for processing, , is not always needed.)

POST-TEST 1

Please write your answers on a separate piece of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

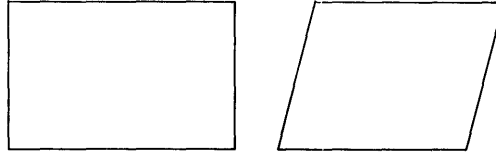
1. Data is another word that means basically the same thing as the word:
 - a. Information
 - b. Processing
 - c. A problem statement
 - d. All of the above
 - e. I don't know
2. A process is:
 - a. A step by step procedure that requires output and produces input
 - b. What occurs when conditions change
 - c. A series of operations that produces an end result
 - d. All of the above
 - e. I don't know
3. The preferred direction of data flow in a flowchart is:



- d. All of the above
- e. I don't know

4. Draw a flowchart of the basic data flow in a data processing system. Be sure to write the appropriate words in the flowchart symbols.
5. Write each of the following descriptions in the appropriate symbol shown below. (Do *not* draw a flowchart; merely write the description in the appropriate symbol. *Note:* You must draw *three* symbols, with a description in each.)
 - a. Subtract overhead from gross profit.
 - b. Read the report.
 - c. Print a chart.

Symbols



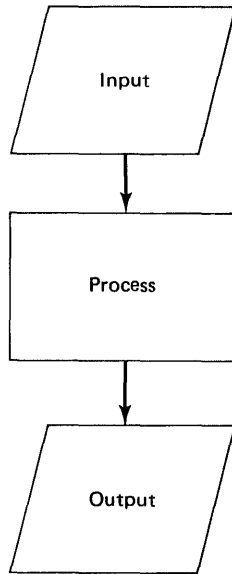
6. Flowchart the following problem statement (do not work out the mathematics for computing average age):

The average age of students is to be computed. A printed report that includes average age by class is to be produced. Students' ages are on the class roster sheet.

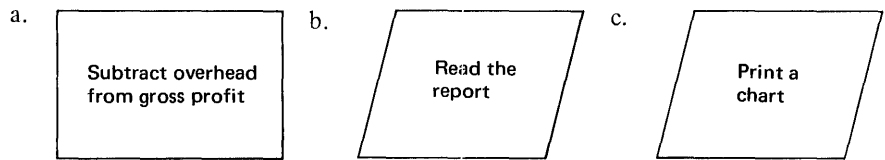
Answers are on the next page.

Answers to Post-Test 1

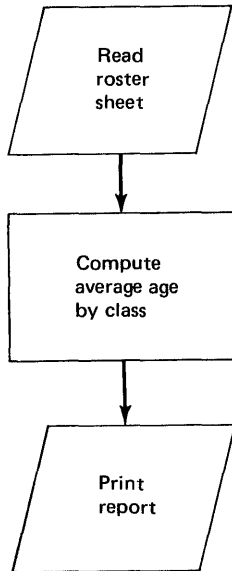
- 1. a
- 2. c
- 3. c
- 4.



- 5. (Note: This question does not ask for a flowchart.)



- 6.



Your wording can be different than that shown in this flowchart, but your answer should show the same basic operations.

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 1. BASIC DATA FLOW

Note: Before reading this programmed instruction sequence, you should have read the descriptive material for this chapter. Directions concerning how to use the programmed instruction sequences in this book are in the introduction.

1. A *process* is a series of operations that is developed to solve a problem. The first operation of the series is to get *input*, the last is to produce *output*. The simple process of solving the following problem can be performed only when the price of the merchandise is available:
“How much is the sales tax for a piece of merchandise, when the tax rate is 4%?”
Consequently, this process (solving for the sales tax) can be started only if:
 - a. Input is available.
 - b. Output is available.

● ● ●

 - a. Input is available
2. The end result of a process, such as a particular amount of sales tax for an item, is called:
 - a. Input
 - b. Output

● ● ●

 - b. Output
3. A process for solving a problem:
 - a. Is a series of operations that produces some end result
 - b. Requires an input and produces an output

● ● ●

Both
4. *Data* is information or factual material. A rate of pay, a social security number, and the time of day are all examples of data. Problem-solving processes performed by data processing systems produce output data. And, the processes carried out by a data processing system:
 - a. Are problem-solving processes
 - b. Require input data

● ● ●

Both
5. The end result of data processing is:
 - a. Output data
 - b. Information

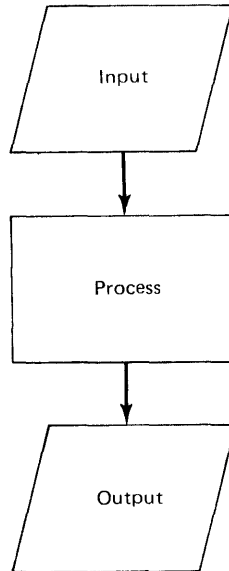
● ● ●

Both
6. Another word that means basically the same thing as the word data is _____

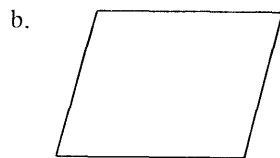
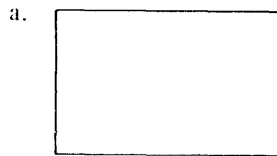
● ● ●

information

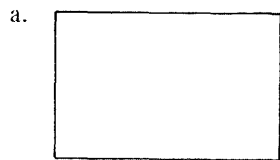
7. Problems to be solved by data processing systems are defined in problem statements, which are given to programmers. Programmers are people who plan the operations needed for the solving of problems by means of data processing systems. They show their plans in flowcharts, which are documents that specify (1) the operations required to solve a problem and (2) the order in which the operations are to be performed.
8. The basic flow of data in a data processing system can be represented by a simple flowchart:



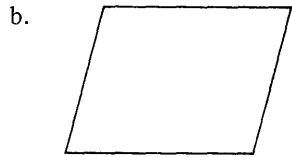
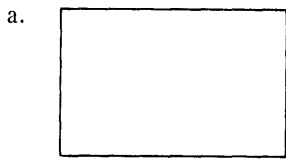
9. The operation (in the above flowchart) that does *not* represent getting input or producing output is symbolized by:



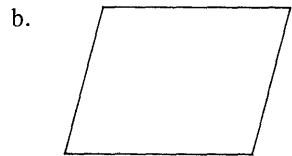
• • •



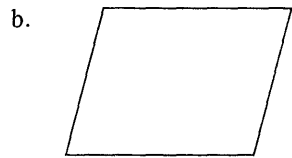
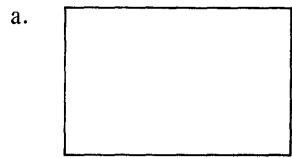
10. The operation of getting available input data is part of a process but is represented by the flowchart symbol:



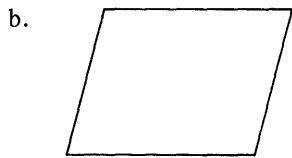
• • •



11. Production of the end result of a process can be represented by the symbol:



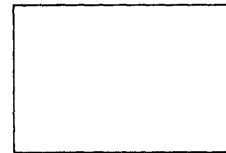
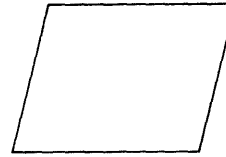
• • •



12. Draw three symbols and write the following descriptions in the symbols (the symbols on the right are those you should use as reference):

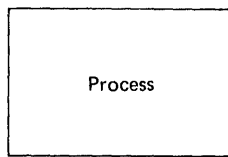
1. Process
2. Input data
3. Output data

Reference Symbols

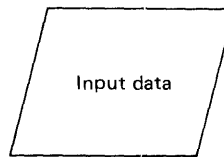


• • •

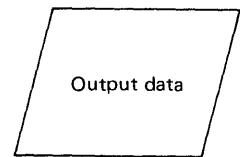
1.



2.



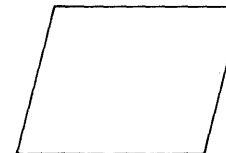
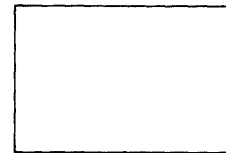
3.



13. Using the symbols on the right as reference, draw three symbols and write the following descriptions in the appropriate symbols:

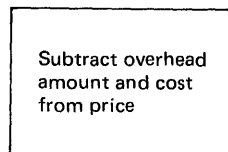
1. Subtract overhead amount and cost from price.
2. Read overhead amount and cost from accounting sheet.
3. Write price and profit on accounting sheet.

Reference Symbols

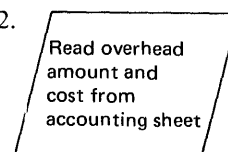


• • •

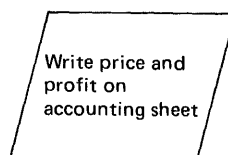
1.



2.



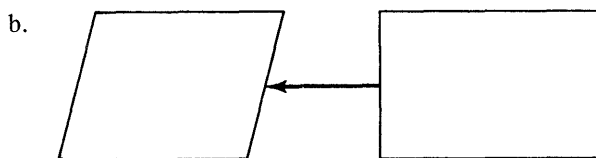
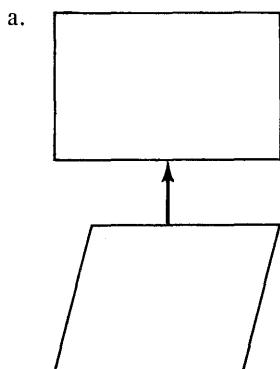
3.



14. The preferred directions of data flow in a flowchart are:

1. Top-to-bottom when possible
2. Left-to-right when top-to-bottom is not convenient

Direction of data flow is shown with lined arrows. Preferred data flow is shown by:

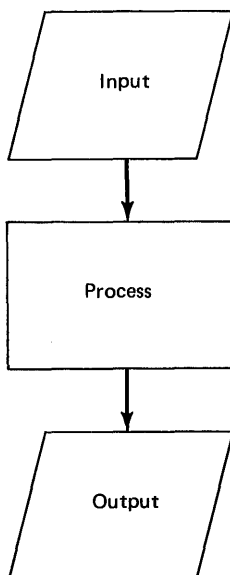


• • •

Neither

15. Draw, on a separate sheet of paper, a flowchart (with appropriate words) that represents the basic flow of data in a data processing system. (Your flowchart need not be precisely drawn; a rough sketch is sufficient for this and other flowcharts that you will be asked to draw.)

• • •



16. Rewrite (do not produce a flowchart) the following problem statement into input, processing, and output steps:

For the item sold, the sales price (as shown on the sales tag attached to the piece of merchandise) and the sales tax must be printed on the cash register tape. The sales tax can be found by multiplying the sales price by .04.

• • •

Your wording need not be identical to the following; you should have the operations organized, however, in the manner shown here.

Input Read sales price from sales tag.

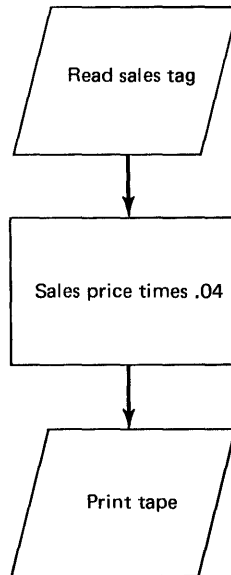
Process Multiply sales price by .04.

Output Print sales price and sales tax on the cash register tape.

17. Draw on a separate sheet of paper, a flowchart of the problem statement from the preceding frame. Use your reorganized statements to produce the flowchart.

• • •

The wording in your flowchart need not be identical to that shown here. Also, a sketch of the flowchart is sufficient; your drawing need not be precise.



Chapter 2. System Overview

For directions, refer to the summary on the inside of the front cover.

PRETEST 2

Write your answers on a separate sheet of paper. Do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. The machine portion of a data processing system carries out the plan that:
 - a. Is called a program and is made up of instructions
 - b. Must be loaded into main storage from an output device
 - c. Is executed by main storage after being stored in alphanumeric form
 - d. All of the above
 - e. I don't know
2. The machine portion of the system that controls execution of each operation in the solution of a problem is:
 - a. Main storage
 - b. Output device
 - c. CPU
 - d. All of the above
 - e. I don't know
3. Before input data can be processed, it must be:
 - a. Executed
 - b. On an input medium
 - c. Read from an output device
 - d. On an output medium
 - e. I don't know
4. Match the two lists (match the numbered items to the lettered items):

<ol style="list-style-type: none">a. CPUb. Input mediumc. Main storaged. I don't know how to match the two lists	<ol style="list-style-type: none">1. Can contain data that can be read by an input device2. Can execute instructions3. Can contain instructions that are ready to be executed
---	---
5. The old data contents of a main storage location are automatically erased:
 - a. During an operation to store new data into that same location
 - b. During an operation to fetch the old data from that same location
 - c. Both a. and b.
 - d. When data is stored in a different location
 - e. I don't know
6. Data can be most quickly accessed by the CPU from:
 - a. A high-speed input device
 - b. A high-speed output device
 - c. Main storage
 - d. All of the above
 - e. I don't know

7. Match the two lists:
- | | |
|--|---------------|
| a. a b c d e K | 1. Alphabetic |
| b. A B % 9 C 1000 | 2. Alphameric |
| c. 0 1 4 7 87 | 3. Numeric |
| d. I don't know how to match the two lists | |
8. After occurrence of a fetch operation for a specific main storage location:
- New data must be stored in that location before another fetch can occur for that location.
 - The location from which the data was fetched still contains the same data.
 - Fetching cannot again occur for that same location.
 - All of the above.
 - I don't know.

Answers are on the next page.

Answers to Pretest 2

1. a
2. c
3. b
4. a. 2
b. 1
c. 3
5. a
6. c
7. a. 1
b. 2
c. 3
8. b

DESCRIPTIVE MATERIAL FOR CHAPTER 2. SYSTEM OVERVIEW

So far we have considered two tasks that are performed by the programmer:

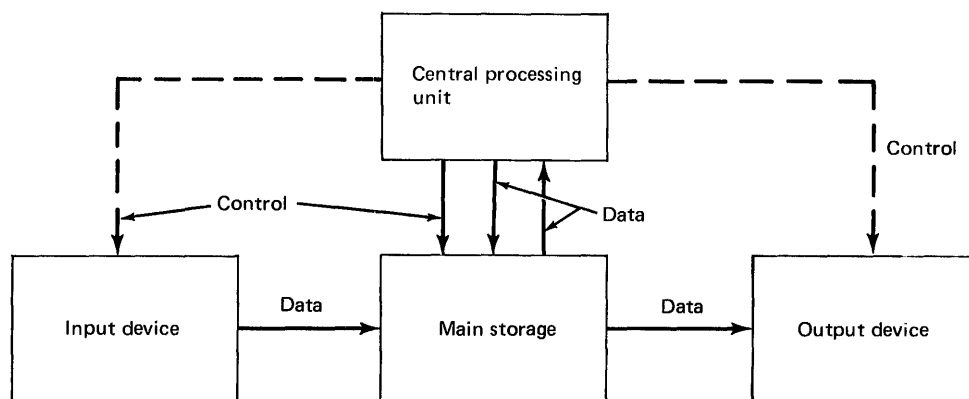
1. He analyzes the problem statement and produces a plan that can be followed to solve the problem.
2. He documents his plan. Any one or more of several documentation schemes can be used. Flowcharting, perhaps the most popular, is stressed in this book.

His next step is to produce a program. A program is made up of instructions. Each instruction represents an operation (or step) that must be performed in the solution of the problem. The overall program is, like the flowchart, the plan for the solution of the problem.

The program is produced by using the flowchart as a guide. Each flowchart step may call for the writing of one or more program instructions.

The program must be produced because it is the detailed plan that is put into the machine portion of a data processing system. Without such a plan, the data processing system can do very little toward the solving of any problem.

The basic parts of the machine portion of a data processing system are:



The central processing unit (CPU) controls overall system operation; the CPU executes (or carries out) operations that are specified by program instructions. Instructions, however, must be in main storage before they can be executed by the CPU.

A program loading operation is performed to store the program into main storage. First, the program is put in an input medium. An input medium can contain data in a form that can be read by an input device (an input machine).

The punched card is a medium with which you may be familiar. (It is described in a later chapter of this book.) Data in punched cards can be read by an input device called a card reader.

A program that is put in punched cards is data that can be read by a card reader. After such cards are placed in a card reader, a program loading operation can be performed:

1. To read the input medium (the cards) at the input device (the card reader)
2. To write the program (that was just read from cards) into main storage

Program loading may be controlled either manually or by another program that has already been loaded into main storage. The procedure depends upon the particular machines used and consequently is not described here.

The Central Processing Unit (CPU)

After a program is in main storage, it can direct system operation through the control provided by the CPU. For example, the CPU must execute a program instruction to start an input or output operation such as:

1. Reading input data from an input medium in an input device
2. Writing output data to an output medium in an output device

The input (or output) device not only reads data from (or writes data to) a medium, but it does so when the CPU calls for the operation.

Note: You will probably hear the following slang expressions frequently used:

- “Data read into main storage from a card reader.”
- “Data written from main storage to a card punch.”

It may be of some interest to know what is meant in each expression. Each, respectively, means:

- Data is read by a card reader and then written into main storage.
- Data is read from main storage and punched (written) into a card at the card punch.

Besides controlling input/output (I/O) operations, the CPU executes other instructions that specify what processing steps are to occur and when they are to occur. Instruction execution must occur to carry out the plan to solve the problem, which is also known as the processing of data.

In other words, instructions are executed in order to process data.

Main Storage

A main-storage location is similar to a mailbox. Each mailbox has its own address and can hold a certain amount of mail. Main-storage locations hold data—input data, output data, data being processed, programs (which are data).

With respect to the amount of data held, however, all main-storage locations are the same in any particular machine. If one location can contain only enough data to represent a two-digit number, then that is all that each other location can contain. In such a machine, data 86 75 41 could be held by three consecutive locations starting, say, at main-storage address 847:

Address of location	—————→	847	848	849
Contents of location	—————→	86	75	41

This is an example; the amount and type of data held by a main-storage location is, of course, dependent upon the actual machine, as is the amount of available locations.

Any main-storage location can be accessed directly by the CPU. Accessed directly means that the CPU, by executing an instruction, can get to (or directly address) one main-storage location just about as quickly as it can get to any other main-storage location. A main-storage location is accessed in order to either read data from or write data into that same location.

Reading is also called fetching. When data is fetched from a main-storage location, the same data remains at that same location after the read operation.

The operation of writing (or storing) data proceeds somewhat differently than reading. Any data that is in a location before a store operation for that same location is erased when the store operation occurs.

Note: The data is erased only in the location in which the new data is being stored. Properly written programs provide for maintaining of needed data in another location when that data is in any danger of being erased as a result of a store operation.

A separate instruction does not erase the location before the store operation. The erasing occurs automatically, as part of the store operation, just before the new data is stored. Fetching contrasted with storing:

	<i>Before Operation</i>	<i>After Fetching</i>	<i>After Storing New Data (86)</i>
Contents of Address 947	71	71	86

Input data, read from an input device, must be stored in main storage before that data can be processed. Any data to be sent to an output medium must also first be in main storage.

Main storage locations can be accessed by the CPU more quickly than any other data source. Data sources, other than main storage, are input media (plural of medium) in input devices. The data from these sources must be put into main storage before that data can be accessed by the CPU. Main storage, then, is a place where:

1. Data must be put before that data can be processed (by the program)
2. The program must be put before that program can be executed (by the CPU) to process the data

The size of main storage in a data processing system is dependent upon the characteristics of the system. Some systems, for example, have about 4,000 addressable main-storage locations. Other systems may have as much as 512,000 addressable locations. Each location can hold a certain amount of data.

Regardless of the size of a main storage, it is important here only to notice what storing and fetching operations do with respect to any specific location. Suppose, for example, that we show a small part of a main storage in the following way:

		Column										
		0	1	2	3	4	5	6	7	8	9	
Row												
	Location 00	0	V	T	A	B	L	E	T	H	A	T
		1	I	S	N	O	T	F	O	U	N	D
		2	0	0	0	0	0	0	0	0	0	0
		3	4	7	6	1	X	Y	Z	W	0	0
		4	9	9	9	9	8	7	6	1	4	3
		5	A	B	C	D	0	0	0	0	0	0
		6	E	F	G	H	0	0	0	0	0	0
	Location 38	7	I	J	K	L	0	0	0	0	0	0
		8	M	N	O	P	0	0	0	0	0	0
		9	Q	R	S	T	0	0	0	0	0	0

Figure 1. Section of a Main Storage

Note: There is no need for you to “learn” the addressing method that is described in the next couple of paragraphs. The important point is the effect that storing and fetching operations have on the contents of main storage locations.

The column and row (see Figure 1) contain numbers that are used for addressing. The address of any location is a column number and a row number, in that order. For example, the address 00 (column 0, row 0) is used to address the location that contains the data V (upper left square in the figure). The location just to the right of the V contains a T, and its address is 1 (column 1) 0 (row 0). The location with an address of 01 (column 0, row 1), on the other hand, contains the data I.

Now, by use of the preceding chart, we can examine more closely what happens during storing and fetching operations. Suppose that a fetch (read) operation is to be performed for location 38 (column 3, row 8). The data read from that location is a P (see Figure 1). After this fetch operation is completed, what is the content of location 38? It still contains the P. (What was done to the P that was fetched is not important here. The important point is that even though the P was fetched, that P data is still available in location 38.) Also, no other location is disturbed by the fetch operation for location 38; the figure still looks the same.

Now, suppose that a store operation to store the new data F is performed, again for location 38. This time, however, the P in location 38 is erased during the store operation. No other main-storage location is disturbed. At conclusion of this store operation, location 38 contains an F.

Note: Other names are used to designate a variety of storages. Main storage is sometimes called prime storage. Additional storage, that is used like an input/output device, is frequently called auxiliary storage. Many other types of storages exist. But the purpose of all of them is to hold data. Main storage is the place where data is put before it can be processed by the program; it also, as already described, is the place where a program must be put before that program can be executed by the CPU.

Data Classifications

Three classes of data are:

1. *Alphabetic:* A b D x Z G H i
2. *Numeric:* 0 99 1841 20000 00 4
3. *Alphameric:* A 0 b 99 D 1841 X # S ¢ %

Alphameric data is made up of:

1. Alphabetic data
2. Numeric data
3. Special-character data (such as # ¢ %)

The word alphameric is derived from the words ALPHAbetic and nuMERIC.

Pure numeric, or pure alphabetic, or alphameric data can be stored into main storage in most data processing systems. Input data, from an input medium, may be pure numeric, pure alphabetic, or alphameric information. The form and organizations of such data are described in more detail in later chapters of this book.

POST-TEST 2

Please write your answers on a separate piece of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. The plan that is carried out by the machine portion of a data processing system:
 - a. Must be loaded into main storage from an output device
 - b. Is stored in alphameric form and is executed by main storage
 - c. Is called a program and is made up of instructions
 - d. All of the above
 - e. I don't know
2. Match the two lists (match the numbered items to the lettered items):

a. Alphameric	1. A B c e D X
b. Numeric	2. 00 98 1000 4996
c. Alphabetic	3. A 9 c Q # 10

 - d. I don't know how to match the two lists
3. Input data to be processed:
 - a. Must first be put on an input medium
 - b. Must be read at an output device
 - c. Must be executed
 - d. Must be on an output medium
 - e. I don't know
4. Match the two lists (match the numbered items to the lettered items):

a. Contains data that can be read by an input device	1. Main storage
b. Contains instructions that are ready for execution	2. Input medium
c. Executes instructions	3. CPU

 - d. I don't know how to match the two lists
5. Erasing of the data contents of a main storage location occurs:
 - a. During a fetch operation, to the same location, for the data
 - b. During a store operation, to the same location, for storing of new data
 - c. Both a and b
 - d. When data is stored in a different location
 - e. I don't know
6. The CPU can access data most quickly from:
 - a. Main storage
 - b. An input device
 - c. An output device
 - d. All of the above
 - e. I don't know
7. After data is fetched from a main storage location, that location:
 - a. Contains the data that was just fetched
 - b. Is not available to the CPU for another fetch operation
 - c. Must accept new data, by means of a store operation, before it can be read again
 - d. All of the above
 - e. I don't know

8. Each operation to be carried out by a data processing system is started by instruction execution in:
- a. An input device
 - b. Main storage
 - c. The CPU
 - d. All of the above
 - e. I don't know

Answers are on the next page.

Answers to Post-Test 2

1. c
2. a. 3
b. 2
c. 1
3. a
4. a. 2
b. 1
c. 3
5. b
6. a
7. a
8. c

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 2. SYSTEM OVERVIEW

1. After the programmer has a description of a problem to be solved, he must plan a solution to that problem. The problem statements (problem descriptions) in this book are not very complex. Usually, however, problems to be solved by a data processing system *are* complex and require a lot of planning on the part of the programmer. The programmer keeps a record of his planning by drawing a flowchart. A flowchart is a series of symbols connected by data flow lines.
The flowchart symbols and the way they are connected to each other by the data flow lines:
 - a. Represent the steps to be performed in the solution of the problem
 - b. Usually represent only input and output but not processing operations



• • •

 - a. Represent the steps to be performed in the solution of the problem
2. Input, output, and processing operations are represented in a flowchart by:
 - a. Asterisks
 - b. Flowchart symbols




• • •

 - b. Flowchart symbols
3. The sequence in which operations should be performed (such as: input, then process, then output) is shown in a flowchart by:
 - a. The data flow lines and arrows that connect the symbols together
 - b. The words in the flowchart symbols

• • •

 - a. The data flow lines and arrows that connect the symbols together
4. Each specific processing operation is defined by:
 - a. The words written in the  symbol
 - b. The words written in the  symbol

• • •

 - a. The words written in the  symbol
5. While the  and  symbols represent the general operations (process *or* input or output) to be performed:
 - a. The actual words that are written in a symbol indicate the specific operation to be performed.
 - b. The data flow lines and arrows indicate the sequence in which operations are to be performed.

• • •

Both
6. A particular flowchart represents:
 - a. The plan for the solution of a specific problem
 - b. The plan for the solution of any data processing problem

• • •

 - a. The plan for the solution of a specific problem

7. After flowcharting his planned solution to the problem, the programmer writes a program. A program is made up of instructions, each of which represents an operation that must be performed in the solution of the problem.
Because the flowchart is used as a guide for writing the program:
- The instructions that the programmer writes represent the plan for the solution of the problem.
 - The program represents the plan for the solution of the problem.
- • •
- Both
8. But why produce two almost identical plans (the flowchart and the program) for the solution of the problem? Because the machine portion of a data processing system carries out the operations called for by program instructions. A flowchart cannot direct machine operations; it is a guide for writing the instructions. Therefore:
- The program is exactly the same as the flowchart.
 - The program is somewhat different than the flowchart.
- • •
- b. The program is somewhat different than the flowchart.
9. A plan (for solving a problem) that is carried out by the machines in a data processing system is called a _____ and is made up of _____.
- • •
- program
instructions
10. The problem that is to be solved is described in a *problem statement*. The overall plan, developed by the programmer, that is used by the machines in a data processing system to give a solution to the problem is called:
- The problem statement
 - The program
- • •
- b. The program
11. Each logical step in the plan for the solution of a particular problem is specified by:
- The problem statement
 - An instruction in the program
- • •
- b. An instruction in the program
12. Match the two lists:
- | | |
|----------------------|--|
| 1. Problem statement | a. Is the plan for the solution of a problem |
| 2. Program | b. Describes the problem to be solved |
- • •
- | | |
|----------------------|--|
| 1. Problem statement | b. Describes the problem to be solved |
| 2. Program | a. Is the plan for the solution of a problem |

13. Both the flowchart and the program represent the plan for the solution of the problem. The symbols, words in the symbols, and data-flow lines in the flowchart make up the plan for the solution to a problem. In the program, the plan is represented by:

- a. The program instructions and the order in which they are written
- b. The reorganized problem statement

• • •

- a. The program instructions and the order in which they are written

14. The various machine operations in a data processing system are controlled by a machine called the central processing unit (CPU). Each operation to be performed must be specified to the CPU by an instruction.

The CPU, then, causes the machines in the system to carry out the plan that is specified by:

- a. The program
- b. The problem statement

• • •

- a. The program

15. The program is executed under control of the machine portion of the system that is called the _____.

• • •

central processing unit (CPU)

16. The three steps of “(1) getting input data, (2) processing that data, and (3) producing output data” are all controlled by the CPU as it is directed by the:

- a. Main storage
- b. Program

• • •

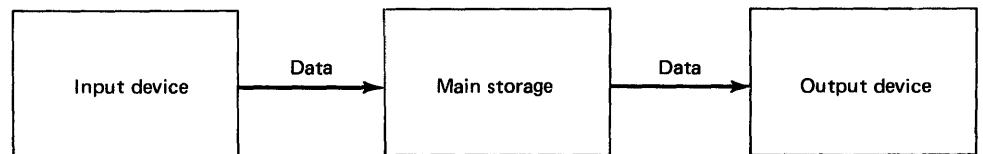
- b. Program

17. Each portion of a program that is executed by the CPU is called _____.

• • •

an instruction

18. Before a program can direct system operation by means of the CPU, that program must be stored into main storage.



A program is data. The above diagram indicates that a program can be written into main storage from:

- a. An input device
- b. An output device

• • •

- a. An input device

19. An input device reads the data in an input medium. A program to be executed by the CPU must:
- a. First be put in an input medium
 - b. Be read at an input device and written into main storage

• • •

Both

20. A program can be carried out or executed by the CPU only after that program is in the section of a data processing system called _____.

• • •

main storage

21. Match the two lists:

- | | |
|-----------------|---|
| 1. Input medium | a. Can execute instructions |
| 2. Main storage | b. Can contain data that can be read by an input device |
| 3. CPU | c. Can contain instructions that are ready to be executed |

• • •

- | | |
|-----------------|---|
| 1. Input medium | b. Can contain data that can be read by an input device |
| 2. Main storage | c. Can contain instructions that are ready to be executed |
| 3. CPU | a. Can execute instructions |

22. The operations of putting data into and then getting data out of main storage are somewhat similar to writing words on and then reading them from a piece of paper. Putting data *into* main storage is called:

- a. Writing
- b. Reading

• • •

- a. Writing

23. Getting data *out of* main storage is called:

- a. Writing
- b. Reading

• • •

- b. Reading

24. The operation of reading data is frequently called *fetching*, while the operation of writing data is often called *storing*.

Input data is read at an input device and then (stored into/fetched into) _____ main storage.

• • •

stored into

25. After input data has been stored into main storage it can be (fetched/stored) _____ from main storage and sent to an output device.

• • •

fetched

26. Match the two lists (*two* items in the right list match *each* item in the left list):

- | | |
|----------------------------------|----------|
| a. Put data into main storage. | 1. Store |
| | 2. Fetch |
| b. Get data out of main storage. | 3. Read |
| | 4. Write |

• • •

- | | |
|----------------------------------|----------|
| a. Put data into main storage. | 1. Store |
| | 4. Write |
| b. Get data out of main storage. | 2. Fetch |
| | 3. Read |

27. *Rule:* Data in a main storage location is erased as the result of a store operation that stores new data in the same location.

Assume the following:

<i>Main storage location address</i>	<i>Contents of location</i>
1080	AB

A write (or store) operation to store 45 into location 1080 is performed as a result of instruction execution in the CPU. The contents of location 1080 after the store operation are (see rule):

- a. 45
- b. A4B5

• • •

- a. 45

28. Assume that the following represents a section of main storage:

4	7	8	← Location with address 10
F	9	1	
A	D	4	← Location with address 22

During a store (write) operation to any location, the data in that location is erased before the new data is stored. For example, storing the new data 6 into the location with address 10 (see preceding diagram) first causes the data 8, in that location, to be erased.

Storing the new data 2 into the location with address 22:

- a. First causes the old data 4 to be erased.
- b. First causes the old data 9 to be erased.

• • •

- a. First causes the old data 4 to be erased.

29. Given the following section of main storage:

A	B	D	
0	4	7	← Location with address 21
Z	K	9	

Assume that a store operation, to write the data 5 into the location at address 21, is performed (refer to the above diagram).

- a. What is the content of the location at address 21 after this store operation?
- b. What is the content of the location just to the left of the location with address 21 after this store operation?

• • •

- a. 5
- b. 4 (This location was not changed; only the data 7 in the location stored into was erased as a result of the store operation.)

30. Erasing of the data content of a main storage location occurs:

- a. During a store operation, to the same location, for storing of new data
- b. When data is stored in a different location

• • •

- a. During a store operation, to the same location, for storing of new data

31. *Rule:* After a fetch operation, the data in the addressed location is the same as before the fetch. The operation of getting data out of a main storage location is called reading or fetching.

Assume the following data:

<i>Main storage address</i>	<i>Contents</i>
1080	47

After a fetch operation at location 1080, that location contains (see rule):

- a. Blank (or all zeros)
- b. 47

• • •

- b. 47

32. An instruction is executed by the CPU to perform the process step of getting data out of a main storage location. Such an operation is called _____.

• • •

reading (or fetching)

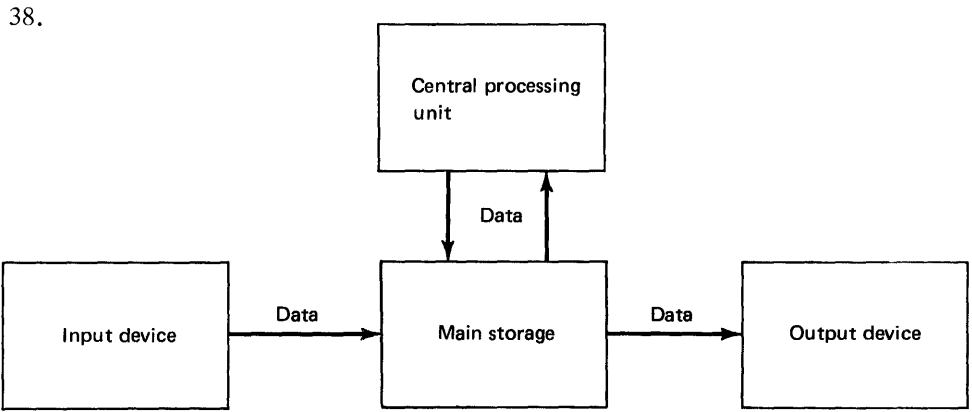
33. After a location of main storage has been read, that same location contains:

- a. Blank (or all zeros)
- b. The data that was just read

• • •

- b. The data that was just read

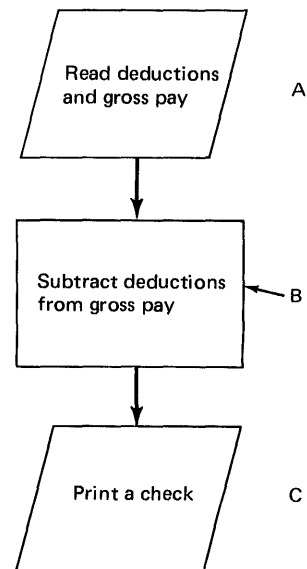
34. After data is fetched from a main storage location:
- a. New data must be stored in that location before another fetch can occur for that location.
 - b. That location contains the data that was just fetched.
- • •
- b. That location contains the data that was just fetched.
35. Erasing of the data contents of a main storage location occurs:
- a. During a fetch operation, to the same location, for the data in that location
 - b. During a store operation, to the same location, for storing new data into that location
- • •
- b. During a store operation, to the same location, for storing new data into that location
36. Erasing of the data contents of main storage location occurs automatically:
- a. During a store operation to that location
 - b. During a fetch operation to that location
- • •
- a. During a store operation to that location
37. Reading and writing data from/into main storage is performed by instruction execution. The CPU executes an instruction to:
- a. Store data into main storage
 - b. Fetch data from main storage
- • •
- Both



- The preceding diagram indicates that data read from an input device:
- a. Can be sent directly to an output device
 - b. Must be stored in main storage before it can be sent to an output device
- • •
- b. Must be stored in main storage before it can be sent to an output device

39. As a general rule, all operations are initiated by instruction execution. Before output data can be written to an output device:
- That data is read from main storage.
 - The CPU must execute an instruction that starts the process of writing to the output device.
- • •
- Both
40. The operation of reading from an input device:
- Is caused by the CPU executing an instruction that specifies such an operation
 - Results in storing input data into main storage
- • •
- Both
41. Input data is read at an input device, stored into main storage, and then processed. Finally the resulting data is sent to an output device. All of these operations can occur only as a result of execution of instructions by the CPU. Match the two lists:
- | | |
|-----------------------------|--------------|
| 1. Instructions are _____ . | a. executed |
| 2. Data is _____ . | b. processed |
- • •
- executed
 - processed
42. Before data from an input device can be processed, it must be stored in _____
- _____.
- • •
- main storage
43. Program instructions are executed by:
- The CPU
 - Main storage
- • •
- The CPU
44. Data is processed by execution of:
- The CPU
 - Instructions
- • •
- Instructions
45. When the CPU actually carries out the operation specified by a program instruction, that instruction is said to be:
- Executed
 - Flowcharted
- • •
- Executed

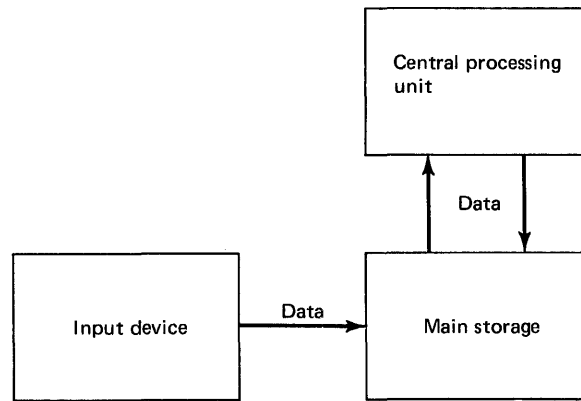
46. The action of handling data by means of instruction execution is indicated by saying that the data is:
- a. Executed
 - b. Processed
- • •
- b. (Instructions are *executed*; data is *processed*.)
47. Before it can be used for output, data must be:
- a. In main storage
 - b. In the CPU
- • •
- a. In main storage
48. Match the two lists:
- | | |
|-----------------|---------------------------------------|
| 1. Main storage | a. Executes instructions |
| 2. CPU | b. Holds data that is to be processed |
- • •
- | | |
|-----------------|---------------------------------------|
| 1. Main storage | b. Holds data that is to be processed |
| 2. CPU | a. Executes instructions |
49. Data to be processed by instruction execution must first be put into main storage. Processing that data by instruction execution is under control of:
- a. The CPU
 - b. Main storage
- • •
- a. The CPU
50. The operations performed in step B can be carried out only after deductions and gross pay are:



- a. In an output device
 - b. In main storage
- • •
- b. In main storage

51. Input data to be processed comes from an input device. Before input data can be read at an input device, that data must be:
- a. On an input medium
 - b. On an output medium
- • •
- a. On an input medium
52. Data to be read by an input device must be on an _____.
- • •
- input medium
53. Input data to be stored into main storage:
- a. Must be read from an input medium that is in an input device
 - b. Is created by the CPU
- • •
- a. Must be read from an input medium that is in an input device
54. Instructions that are used to write output data are:
- a. Executed by the CPU
 - b. Executed after input data has been processed
- • •
- Both
55. During instruction execution, the CPU can always directly access the storage location for which an address is provided. (*Directly access* means go directly to the location for either a store or a read operation.) Main storage locations are specified by addresses in program instructions.
- The CPU has just read data out of main storage location 4096. The CPU then:
- a. Can read main-storage location 8198 directly
 - b. Must read main storage locations 4097 through 8197, one at a time and in order, before it can read the data from location 8198
- • •
- a. Can read main-storage location 8198 directly

56.



The preceding diagram shows that the CPU can access data from main storage more quickly than from an input device because:

- a. Data from the input device must go to main storage before it can go to the CPU.
- b. There is no direct path from the input device to the CPU.

• • •

Both

57. Data available from a medium in an input device:

- a. Is stored in main storage automatically
- b. Is read at the input device and stored into main storage under program-instruction control

• • •

- b. Is read at the input device and stored into main storage under program-instruction control

58. Input data to be processed must first be read at an input device and then stored into main storage. After the input data is in main storage, it is:

- a. Processed by the CPU (as directed by the program)
- b. Sent back to the input device for reformatting

• • •

- a. Processed by the CPU (as directed by the program)

59. The CPU can get to (or access) data that is in main storage more quickly than it can access data in an input device because:

- a. Data on an input medium in an input device is normally not accessible to the system.
- b. Data in an input device must first be put in main storage before the CPU can access that data for processing.

• • •

- b. Data in an input device must first be put in main storage before the CPU can access that data for processing.

60. Three classes of data are:

1. Alphabetic
2. Numeric
3. Alphameric

Alphameric data is made up of numeric, alphabetic, and special-character (such as \$ # %) data.

Match the following:

1. Alphabetic
2. Numeric
3. Alphameric

- a. # A 9 4000 % G
- b. 0 1 495 1000
- c. A B C e X f g

• • •

1. Alphabetic
2. Numeric
3. Alphameric

- c. A B C e X f g
- b. 0 1 495 1000
- a. # A 9 4000 % G

61. Alphameric data can be stored in main storage. Instruction execution in the CPU can cause storing of:

- a. Numeric data into main storage
- b. Alphabetic data into main storage

• • •

Both

62. Data made up of numeric, alphabetic, and special-character data is called _____.

• • •

Alphameric (or alphameric data)

Chapter 3. The Punched Card Medium

For directions, refer to the summary on the inside of the front cover.

PRETEST 3

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions



1. Printing for field names, column numbers, and punch positions is usually on the side of a punched card called:
 - a. The row side
 - b. The column side
 - c. The face
 - d. All of the above
 - e. I don't know
2. A data item read from a punched card is identified by the machines in a data processing system according to:
 - a. The printing on the card
 - b. The specific card columns from which the data item is read
 - c. The row from which the data item is read
 - d. All of the above
 - e. I don't know
3. The punched card:
 - a. Can be an input/output medium
 - b. Can be an input medium only
 - c. Can be an output medium only
 - d. All of the above
 - e. I don't know

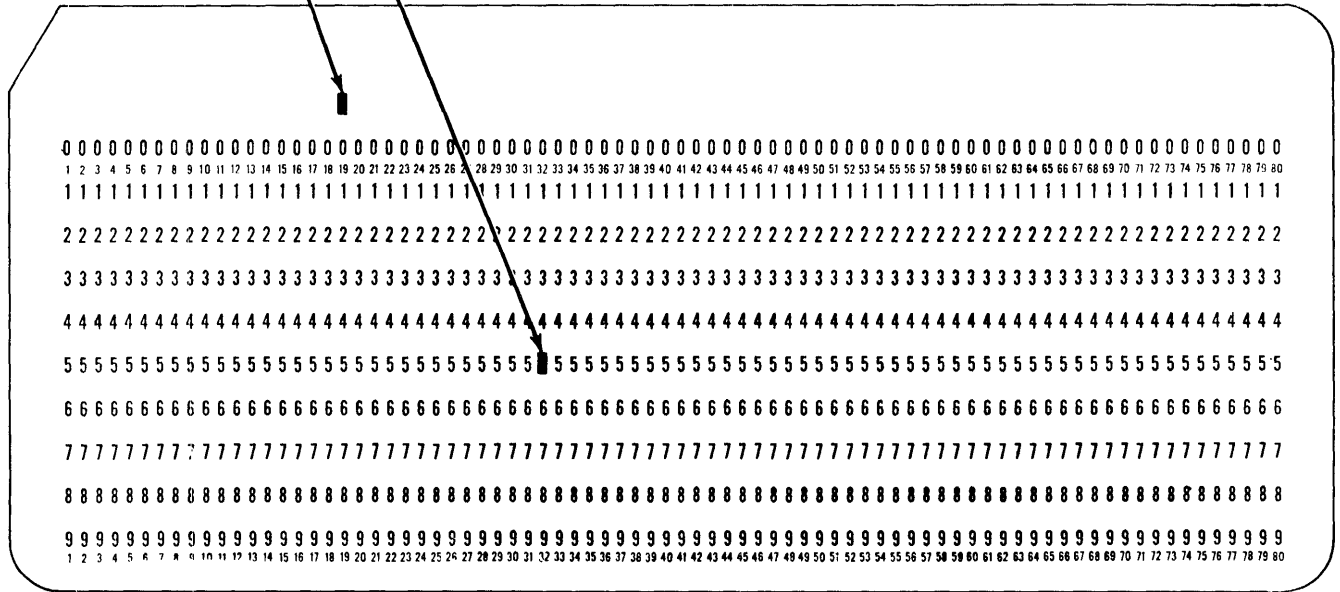
Answers to Pretest 3

1. c
2. b
3. a
4. A
5. B
6. Yes
7. No
8. 70 (in columns 79 and 80)
9. 12 and 1 (or 12-1)
10. 11 and 5 (or 11-5)
11. 8 (in column 78)
12. 4 (in column 57)

DESCRIPTIVE MATERIAL FOR CHAPTER 3. THE PUNCHED CARD MEDIUM

Input media can contain data in machine-readable form. An input device (machine) called a card reader is used to read input data from punched cards.

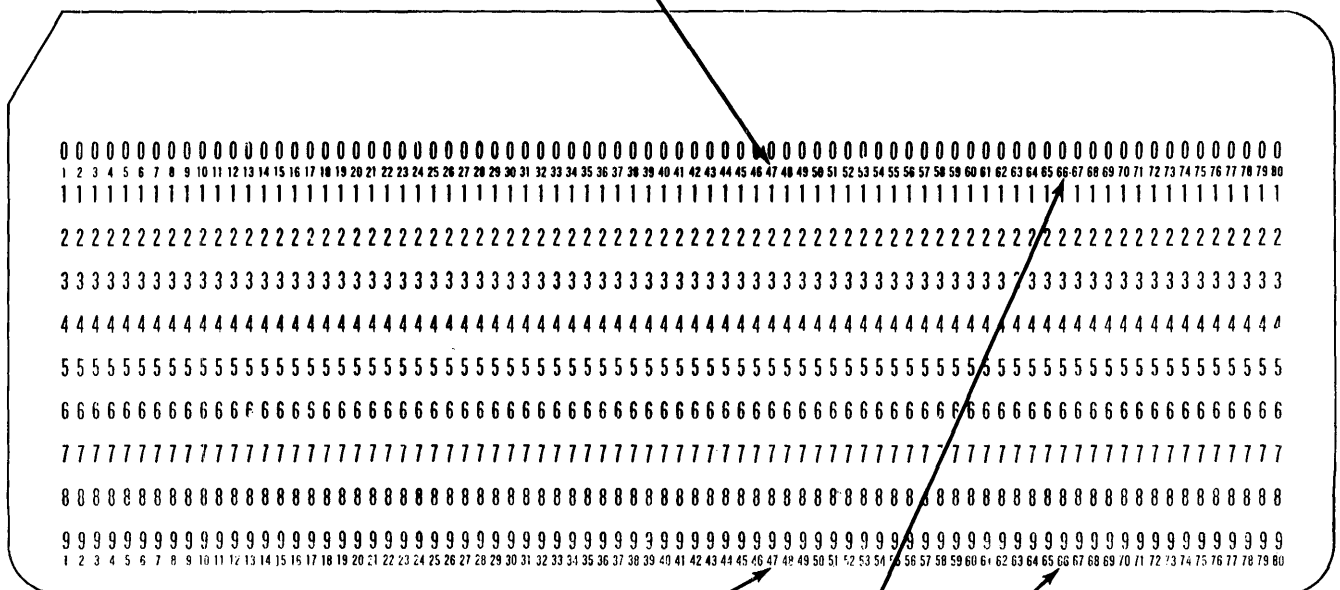
Data is represented in the punched card medium by small, rectangular punched holes, such as  and 



This data representation is machine readable (also called machine sensible) to a card reader (an input device). The punched holes can be put into as many as 80 vertical columns in a single card. (Some cards have fewer than 80 columns but the only card described in this book is the 80-column card.)

Frequently, column locations are specified by two rows of numbers printed on the face of the card. For example, column location 47 is

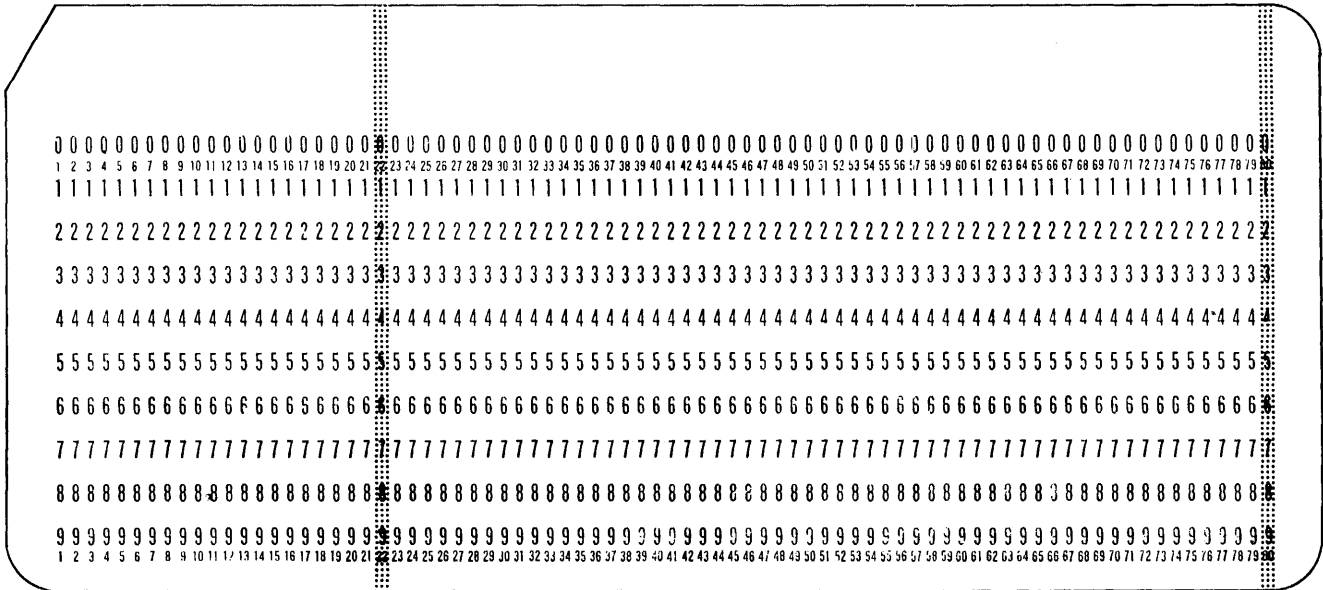
shown here



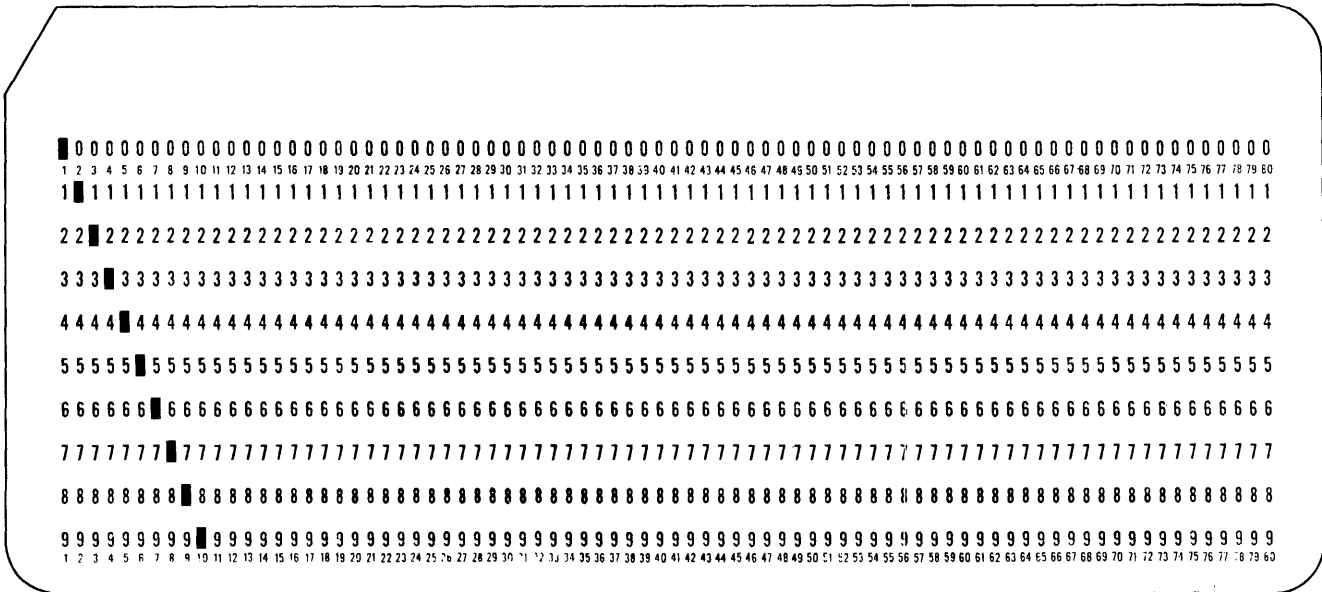
and here . . . column 66, here

and here

Each column is a narrow strip between the bottom and the top of the card.
 For example: column 22 and column 80.

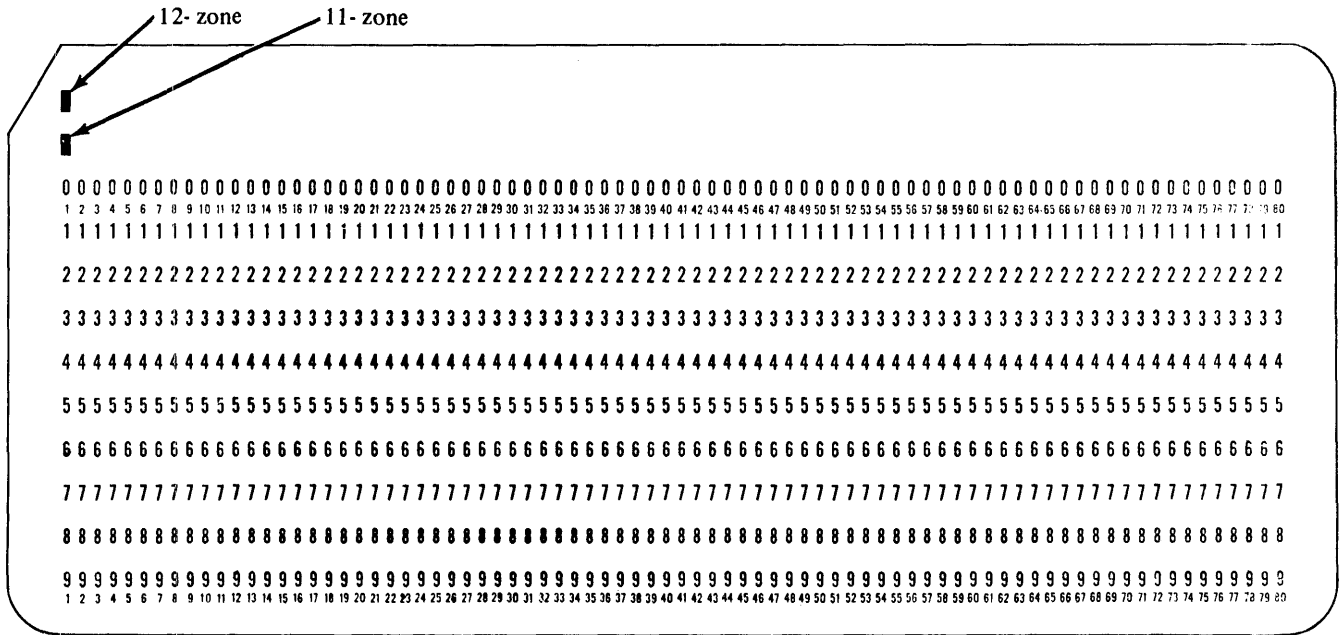


The rectangular holes are punched only at specific places in the columns. In some cards, as in the following figure, the locations of the digit punching positions are shown by the printed digits 0 through 9:

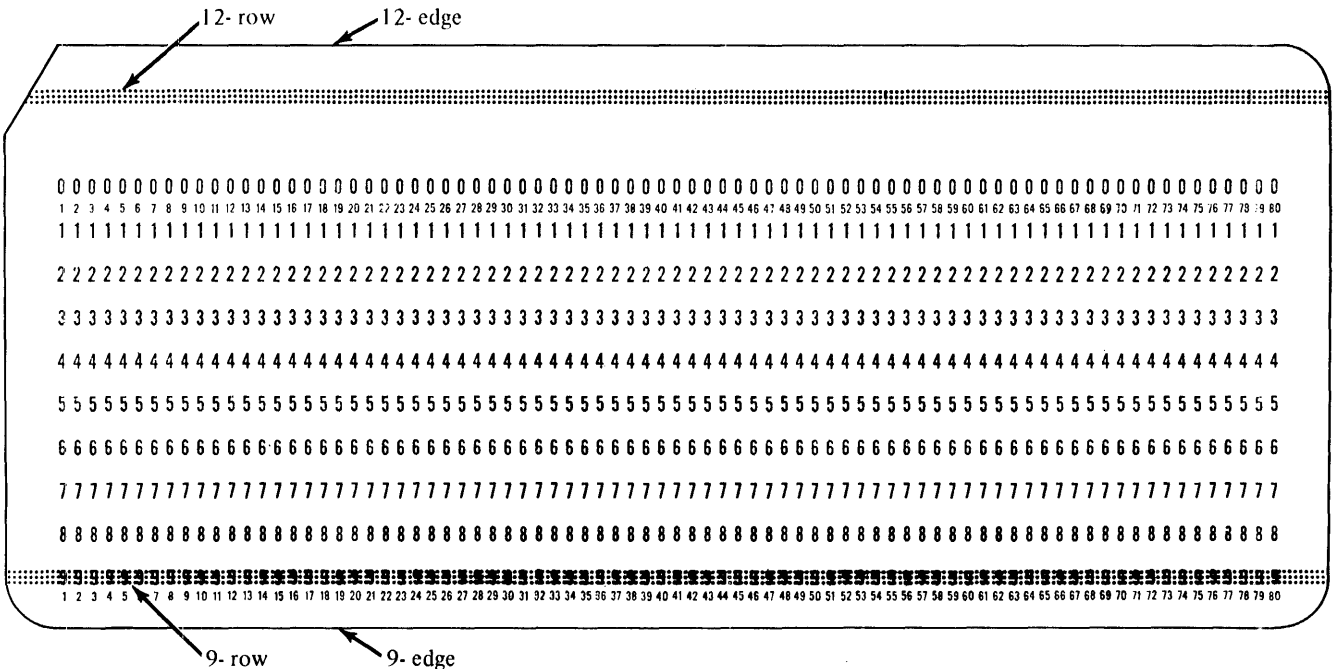


The preceding figure shows the punched hole representation of the digits 0123456789 in columns 1 through 10 respectively. This is the way that numeric data is usually represented: one digit punch per column. Notice that the 0's, 1's, and so forth are printed in rows across the face of the card so that each column contains printing for all the digits (0 through 9). *Note:* Not all cards contain such digit printing.

Two other punching positions, for which there is usually no printing on the card, are the 12-zone and the 11-zone positions. These positions, just like the digit positions, are in each and every column. The 12-zone punching position is just above each 11-zone position. These positions are shown for column 1:



When you look at the face of a card (the side of the card that usually contains printed matter), the 9 row is at the bottom and the 12 row is at the top. The top edge of the card is called the 12 edge and the bottom the 9 edge:



Zone punched holes used in combination with digit punched holes represent letters and special characters. For such data, the 0 digit-punch position is redefined to be a zone-punch position.

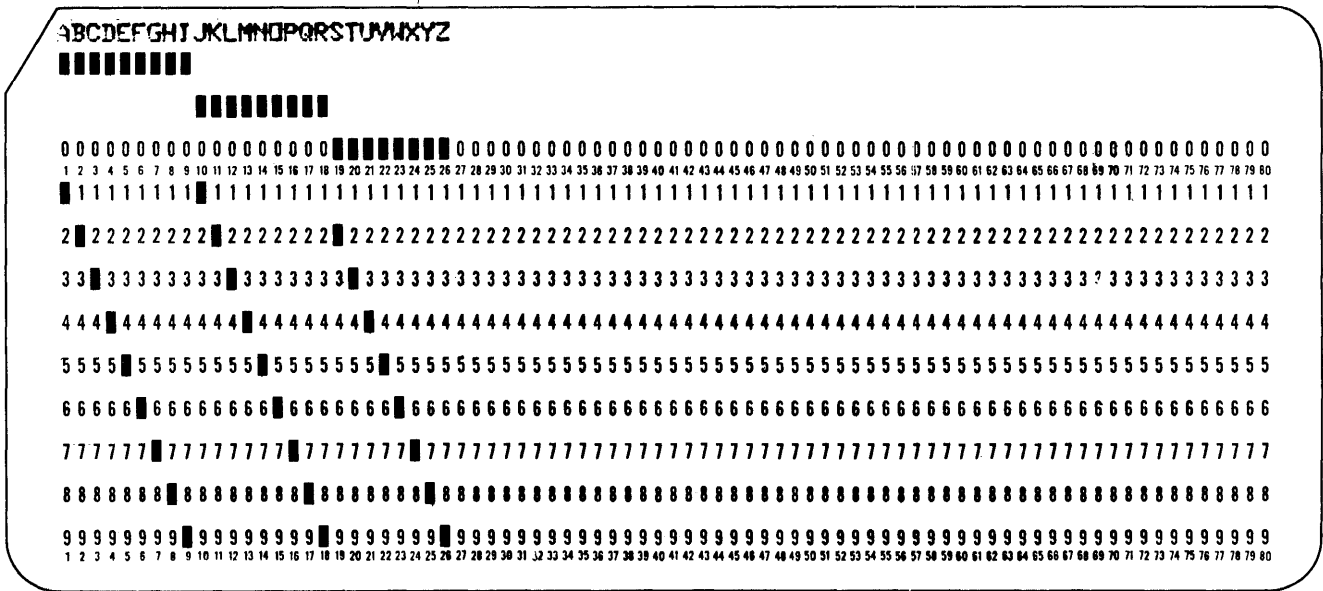
Punched-Hole Patterns for Letters

Skip this section and go to the next heading, "Data Organization," if you are not interested in reading about punched-hole patterns for letters. The following is presented for the convenience of those who may have interest in such information. You will not be tested on this information.

The three zone-punch positions (12, 11, and 0) are used with the digit punches to represent capital letters as follows:

<i>Punch Pattern</i>	<i>Uppercase Letter</i>	<i>Punch Pattern</i>	<i>Uppercase Letter</i>	<i>Punch Pattern</i>	<i>Uppercase Letter</i>
12-1	A	11-1	J		
12-2	B	11-2	K	0-2	S
12-3	C	11-3	L	0-3	T
12-4	D	11-4	M	0-4	U
12-5	E	11-5	N	0-5	V
12-6	F	11-6	O	0-6	W
12-7	G	11-7	P	0-7	X
12-8	H	11-8	Q	0-8	Y
12-9	I	11-9	R	0-9	Z

These patterns, shown in a punched card, are:



Lowercase letters and special characters can be similarly represented. In order to distinguish between uppercase and lowercase letters, other zone combinations are necessary. In one punched-code scheme, for example, the lowercase "a" is represented by the punched-hole pattern 12-0-1, the letter "j" by 12-11-1, the letter "s" by 11-0-1, and so forth. In other words, the digit punch for these letters is the same as their uppercase digit punch counterpart. Only the zones have been changed to denote the difference (uppercase versus lowercase).

Assigning of other punch positions as "zones" (such as the 9 digit to the 9 zone) is done to provide for special-character representations. Alternately, the 12, 11, and 0 zones are used in combination with each other and with digit punches, as in the case of the lowercase letters, to specify certain special characters.

Such punched-hole patterns are not shown in this book. That information can be looked up, when needed, from reference literature available at data processing system installations.

A field, then, is the location of data. A data item is the actual data in a field. Only one data item is shown punched in the preceding card.

A field can be made up of one or more columns. The maximum field size in an 80-column card is 80 columns, though it is unusual to have a field of such size in the card medium. Notice that fields are defined locations, according to the data processing problem to be solved. For example, the following payroll application card has a defined field in columns 27 and 28, the department field:

EMPLOYEE NAME	SEX	SOCIAL SECURITY NUMBER		EMPLOYEE NO.		RATE	DATE			HOURS			EARNINGS			DEDUCTIONS		NET EARNINGS	STATUS	
		DEPT. CLOCK		MO. DAY YEAR			REGU-LAR	OVER-TIME	TOTAL	REGULAR	OVERTIME	GROSS	OASI	WITH-TAX						
00000000000000000000		000000000000		000000		00000	00	00	00	00	00	00	00	00	00	00	00	00	00	00
11111111111111111111	M	111111111111		111111		11111	11	11	11	11	11	11	11	11	11	11	11	11	11	11
22222222222222222222	F	222222222222		222222		22222	22	22	22	22	22	22	22	22	22	22	22	22	22	22
33333333333333333333		333333333333		333333		33333	33	33	33	33	33	33	33	33	33	33	33	33	33	33
44444444444444444444		444444444444		444444		44444	44	44	44	44	44	44	44	44	44	44	44	44	44	44
55555555555555555555		555555555555		555555		55555	55	55	55	55	55	55	55	55	55	55	55	55	55	55
66666666666666666666		666666666666		666666		66666	66	66	66	66	66	66	66	66	66	66	66	66	66	66
77777777777777777777		777777777777		777777		77777	77	77	77	77	77	77	77	77	77	77	77	77	77	77
88888888888888888888		888888888888		888888		88888	88	88	88	88	88	88	88	88	88	88	88	88	88	88
99999999999999999999		999999999999		999999		99999	99	99	99	99	99	99	99	99	99	99	99	99	99	99

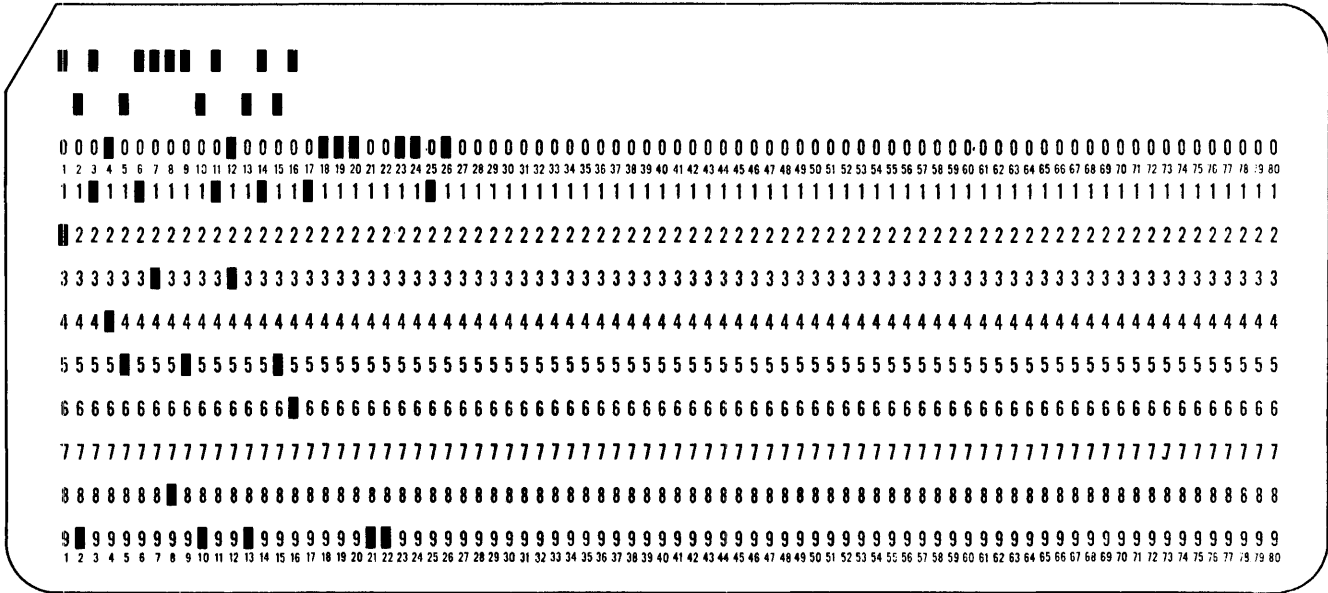
Columns 24 and 25 do not make up a field in the preceding card. They are part of the field defined (by the printing on the card) as the social security number field.

Input data items can be processed properly by a data processing system only if they are in the proper fields. The CPU handles data items, which are read from punched cards by a card reader, according to the locations of the data items in the card, not according to printing on the cards.

Suppose, for example, that the data items from the following card are read by a card reader and stored in main storage:

EMPLOYEE NAME	SEX	SOCIAL SECURITY NUMBER		EMPLOYEE NO.		RATE	DATE			HOURS			EARNINGS			DEDUCTIONS		NET EARNINGS	STATUS
DEPT. CLOCK		MO. DAY YEAR			REGU-LAR		OVER-TIME	TOTAL	REGULAR	OVERTIME	GROSS	OASI	WITH-TAX						
00000000000000000000		000000000000		000000		00000	00	00	00	00	00	00	00	00	00	00	00	00	00
11111111111111111111	M	111111111111		111111		11111	11	11	11	11	11	11	11	11	11	11	11	11	11
22222222222222222222	F	222222222222		222222		22222	22	22	22	22	22	22	22	22	22	22	22	22	22
33333333333333333333		333333333333		333333		33333	33	33	33	33	33	33	33	33	33	33	33	33	33
44444444444444444444		444444444444		444444		44444	44	44	44	44	44	44	44	44	44	44	44	44	44
55555555555555555555		555555555555		555555		55555	55	55	55	55	55	55	55	55	55	55	55	55	55
66666666666666666666		666666666666		666666		66666	66	66	66	66	66	66	66	66	66	66	66	66	66
77777777777777777777		777777777777		777777		77777	77	77	77	77	77	77	77	77	77	77	77	77	77
88888888888888888888		888888888888		888888		88888	88	88	88	88	88	88	88	88	88	88	88	88	88
99999999999999999999		999999999999		999999		99999	99	99	99	99	99	99	99	99	99	99	99	99	99

If, instead, the following card is read, the same data is stored in the same location in main storage because the same data is punched into the same columns as in the preceding card even though there is no printing of fields on the following card:



Regardless of the card, assume that the three data items (the name, sex, and social security number for a specific employee) are read from either of the preceding two cards and stored into main storage in the following locations:

<i>Field from card</i>	<i>Card columns</i>	<i>Stored into Main Storage locations (at addresses)</i>
Employee name	1-16	801-816
Sex	17	817
Social security number	18-26	818-826

The program being used processes the data item in main storage locations 818-826 as a social security number. The location of data, then, is of primary importance to the program in a data processing system.

Two devices that punch cards with data are:

1. *Keypunch.* Original (or source) documents, such as typewritten pieces of paper, are given to a keypunch operator. (A keypunch is a machine that is operated in a manner similar to a typewriter, except that cards are punched instead of pages typed.) The keypunch operator then keypunches the data from the original documents into cards.
2. *Card punch.* A card punch is an output device. Output data can be punched into cards that are in a card punch. The CPU causes, under program instruction control, reading of the output data from main storage and writing (punching) of that data into the cards in the card punch. (*Note:* Unfortunately, a keypunch is sometimes called a card punch. In this book, however, the distinction that was just made will be maintained.)

Cards punched by either of the two preceding methods can be read by a card reader to provide input data to a data processing system. The punched card, because it can be used to provide input data or accept output data is an input/output medium.

POST-TEST 3

Please write your answers on a separate piece of paper. Please do not guess. Specify the “I don’t know” answer when appropriate.

Questions

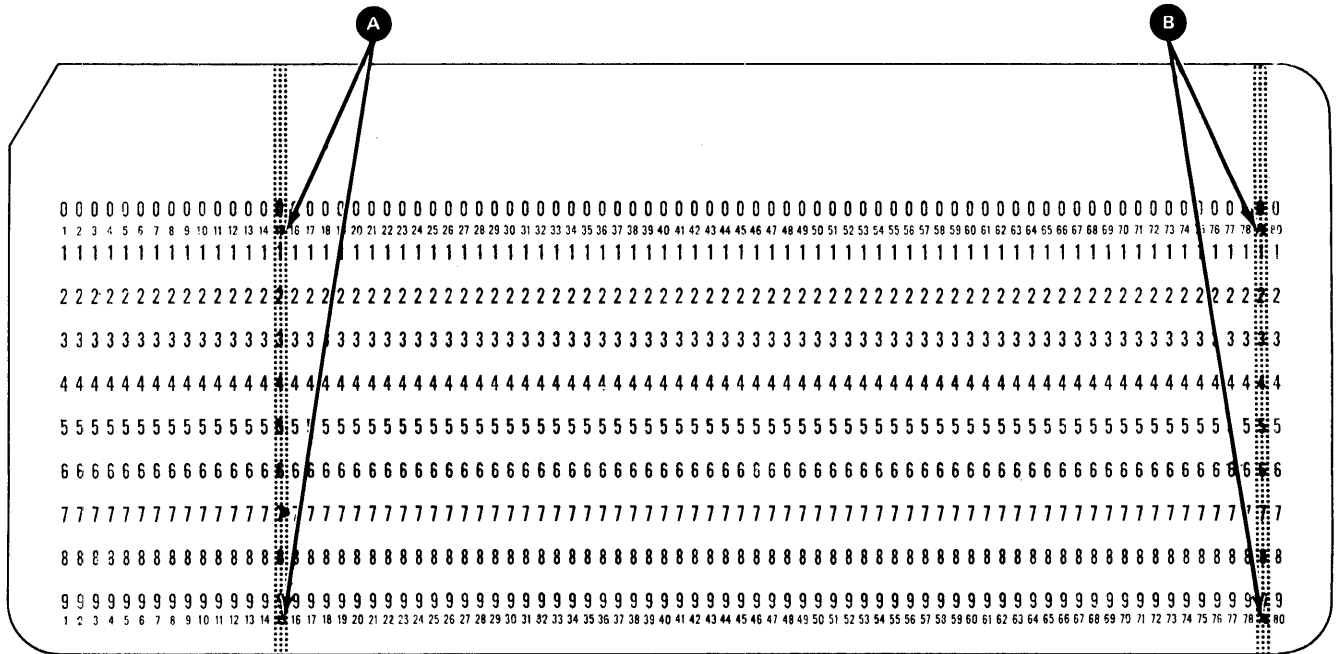
1. Usually, the side of a punched card that contains printing for field designation, card column numbers, and punch positions is called:
 - a. The face
 - b. The column side
 - c. The row side
 - d. All of the above
 - e. I don’t know
2. The machines in a data processing system identify a data item read from a punched card by:
 - a. Zone punches
 - b. The printing on the card
 - c. The specific card columns from which the data item is read
 - d. All of the above
 - e. I don’t know
3. The punched card is:
 - a. An input medium only
 - b. An output medium only
 - c. An input/output medium
 - d. All of the above
 - e. I don’t know

Answers to Post-Test 3

1. a
2. c
3. c
4. B
5. A
6. No
7. Yes
8. 12312
9. 11-3
10. 12-5
11. 1
12. 3

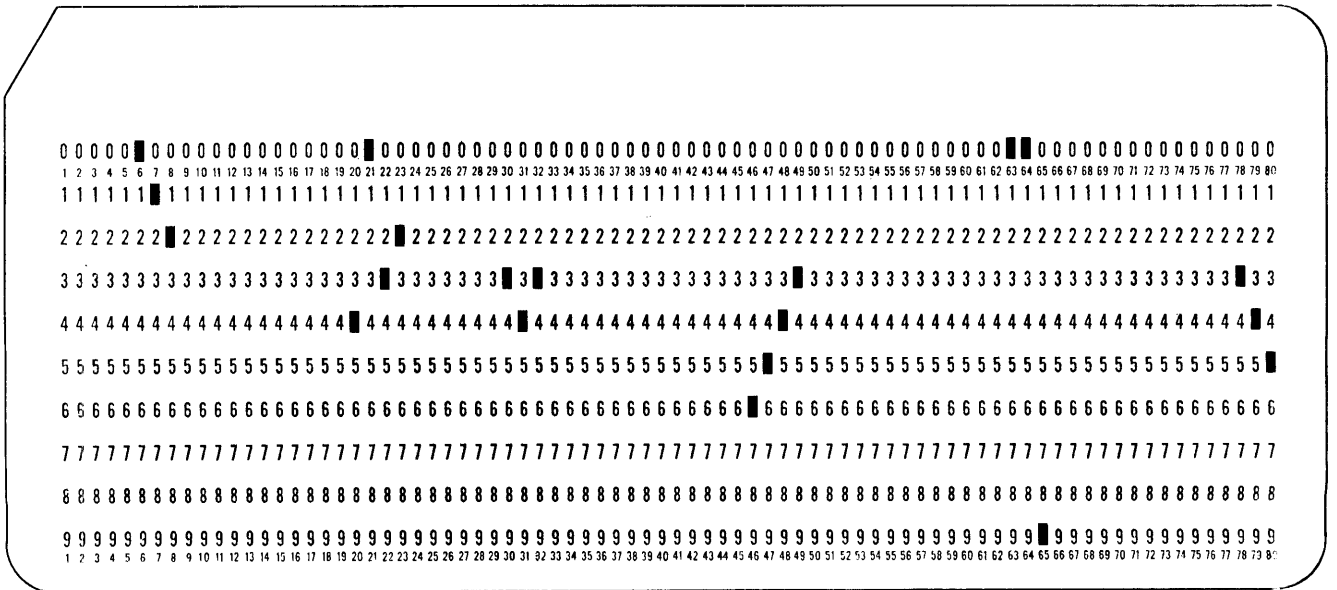
PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 3. THE PUNCHED CARD MEDIUM

- The locations of card columns are shown by two rows of printed numbers. For example, in the following figure, the A locates the printed numbers for column 15. What column is located by the B?



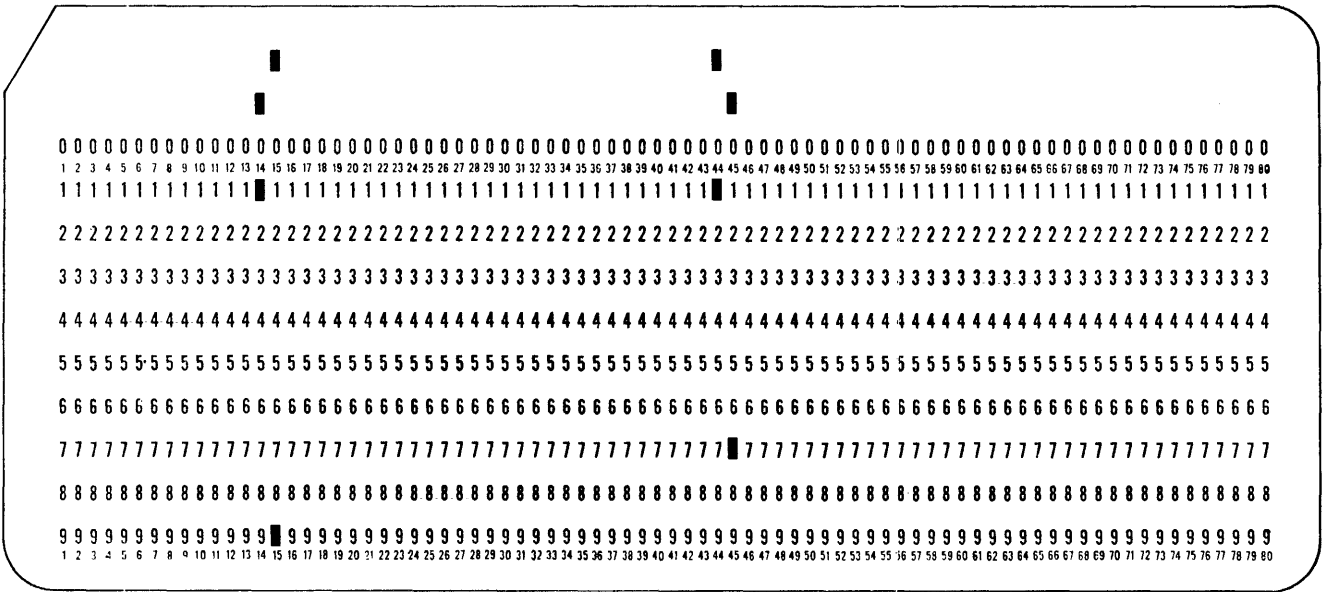
•••
79

- Rectangular holes, punched in columns, represent data. Printed digits (0 through 9) in every column specify the digit punching positions within each column. In the following card, the numeric data 012 is represented by the punched holes in columns 6, 7, and 8. What data is represented in columns 78, 79, and 80?



•••
345

3. The 11-zone punching position is right over the 0 digit punching position in every column.
 The 12-zone punching position is right over the 11-zone punching position in every column.
 The punching positions shown punched in column 14 are 11 and 1 (11-1).
 The punching positions punched in column 15 are 12 and 9 (12-9).
- What punching positions are shown punched in column 44?
 - What positions in column 45?



•••

- 12 and 1 (or 12-1)
- 11 and 7 (or 11-7)

4. The 0-digit position is sometimes called a *zone* position. The 12, 11, and 0 zones are used in combination with the numeric positions (1 through 9) to represent alphabetic data. For example, the data in column 1 (in the following card) is punched with numeric data because *only* a numeric position is punched. The data in column 10, however, is alphabetic because *both* a zone and a numeric position are punched.

The data represented in column 18 of the following card is:

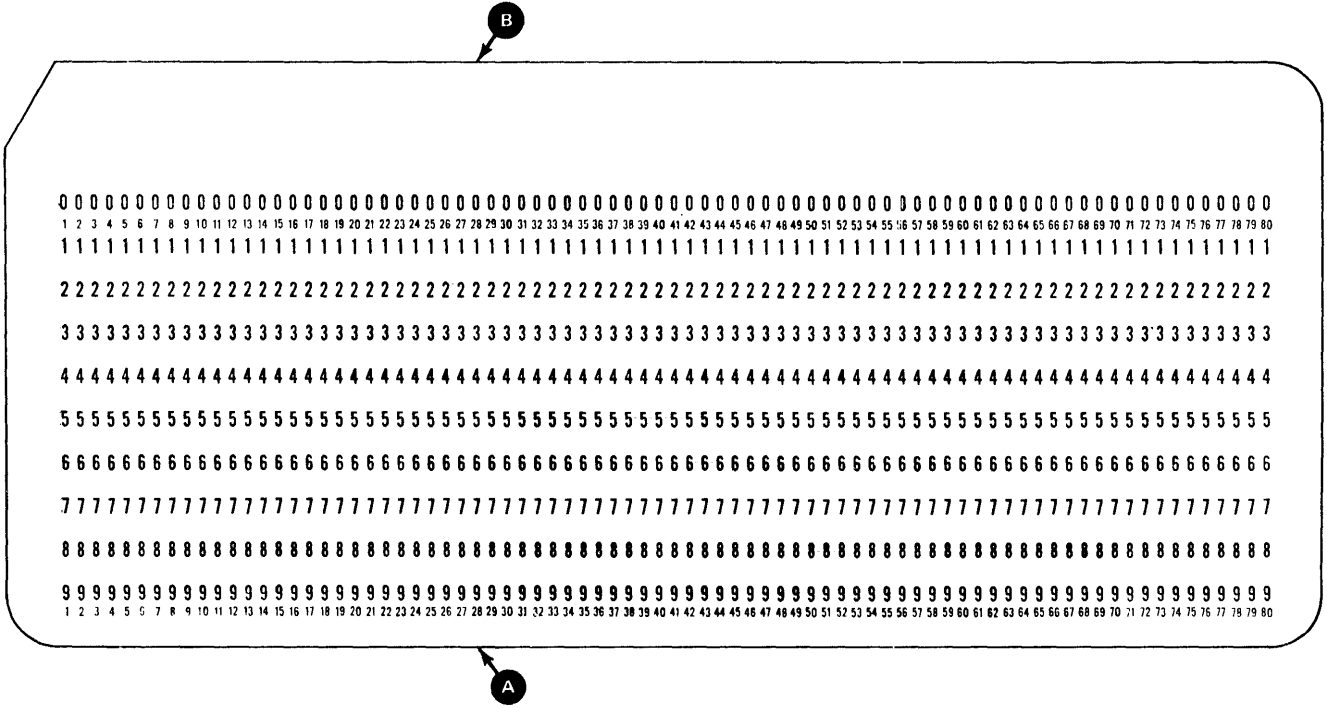
- a. Alphabetic
- b. Numeric

The diagram shows a punched card with 80 columns and 12 rows. The columns are numbered 1 through 80 at the bottom. The card is mostly filled with zeros. Column 18 is highlighted with a vertical bar. Below the card, there are three dots and the text 'a. Alphabetic'.

-
- a. Alphabetic

5. The 12 row is across the top of the face of a card just as the 9 row is across the bottom. Match the two lists (refer to the following card):

- 1. 12 edge A
- 2. 9 edge B



•••

- 1. B
- 2. A

6. Usually printing is only on the face of a card. Sometimes other information is printed on the back of a card. You would expect card column numbers, when they are printed, to be:

- a. On the face of a card
- b. On the back of a card

•••

- a. On the face of a card

9. By looking at the following card, you can see that:
- a. A field may require only one column.
 - b. The largest possible field in a card could be up to 80 columns.

EMPLOYEE NAME	SOCIAL SECURITY NUMBER	EMPLOYEE NO.	RATE	DATE		HOURS			EARNINGS			DEDUCTIONS		NET EARNINGS	STATUS	
				MO.	DAY	REG.	OVER-TIME	TOTAL	REGULAR	OVERTIME	GROSS	OASI	WITH. TAX			
																DEPT.
00000000000000000000	00000000000000000000	00000000000000000000	00000000000000000000	00	00	00	00	00	00	00	00	00	00	00	00	00
11111111111111111111	11111111111111111111	11111111111111111111	11111111111111111111	11	11	11	11	11	11	11	11	11	11	11	11	11
22222222222222222222	22222222222222222222	22222222222222222222	22222222222222222222	22	22	22	22	22	22	22	22	22	22	22	22	22
33333333333333333333	33333333333333333333	33333333333333333333	33333333333333333333	33	33	33	33	33	33	33	33	33	33	33	33	33
44444444444444444444	44444444444444444444	44444444444444444444	44444444444444444444	44	44	44	44	44	44	44	44	44	44	44	44	44
55555555555555555555	55555555555555555555	55555555555555555555	55555555555555555555	55	55	55	55	55	55	55	55	55	55	55	55	55
66666666666666666666	66666666666666666666	66666666666666666666	66666666666666666666	66	66	66	66	66	66	66	66	66	66	66	66	66
77777777777777777777	77777777777777777777	77777777777777777777	77777777777777777777	77	77	77	77	77	77	77	77	77	77	77	77	77
88888888888888888888	88888888888888888888	88888888888888888888	88888888888888888888	88	88	88	88	88	88	88	88	88	88	88	88	88
99999999999999999999	99999999999999999999	99999999999999999999	99999999999999999999	99	99	99	99	99	99	99	99	99	99	99	99	99



Both

10. According to the layout on the following card:
- a. Column 43 is a field.
 - b. Columns 63 through 65 are a field.

EMPLOYEE NAME	SOCIAL SECURITY NUMBER	EMPLOYEE NO.	RATE	DATE		HOURS			EARNINGS			DEDUCTIONS		NET EARNINGS	STATUS	
				MO.	DAY	REG.	OVER-TIME	TOTAL	REGULAR	OVERTIME	GROSS	OASI	WITH. TAX			
																DEPT.
00000000000000000000	00000000000000000000	00000000000000000000	00000000000000000000	00	00	00	00	00	00	00	00	00	00	00	00	00
11111111111111111111	11111111111111111111	11111111111111111111	11111111111111111111	11	11	11	11	11	11	11	11	11	11	11	11	11
22222222222222222222	22222222222222222222	22222222222222222222	22222222222222222222	22	22	22	22	22	22	22	22	22	22	22	22	22
33333333333333333333	33333333333333333333	33333333333333333333	33333333333333333333	33	33	33	33	33	33	33	33	33	33	33	33	33
44444444444444444444	44444444444444444444	44444444444444444444	44444444444444444444	44	44	44	44	44	44	44	44	44	44	44	44	44
55555555555555555555	55555555555555555555	55555555555555555555	55555555555555555555	55	55	55	55	55	55	55	55	55	55	55	55	55
66666666666666666666	66666666666666666666	66666666666666666666	66666666666666666666	66	66	66	66	66	66	66	66	66	66	66	66	66
77777777777777777777	77777777777777777777	77777777777777777777	77777777777777777777	77	77	77	77	77	77	77	77	77	77	77	77	77
88888888888888888888	88888888888888888888	88888888888888888888	88888888888888888888	88	88	88	88	88	88	88	88	88	88	88	88	88
99999999999999999999	99999999999999999999	99999999999999999999	99999999999999999999	99	99	99	99	99	99	99	99	99	99	99	99	99



Neither

11. For the card shown below:

- Is column 41 a field?
- Is column 76 a field?
- What is the number shown punched in the gross field?
- What punching positions are shown punched in column 17?
- What punching positions are shown punched in column 16?

EMPLOYEE NAME	SOCIAL SECURITY NUMBER	EMPLOYEE NO.	RATE	DATE	HOURS			EARNINGS			DEDUCTIONS		NET EARNINGS	STATUS	
					REGULAR	OVERTIME	TOTAL	REGULAR	OVERTIME	GROSS	OASI	WITH. TAX			
															MO.
INTERNATIONAL BUSINESS MACHINES CORPORATION	0000000000000000	0000000000000000	0000000000000000	0000000000000000	00	00	00	00	00	00	00	00	00	00	00
111111111111111111	1111111111111111	1111111111111111	1111111111111111	1111111111111111	1	1	1	1	1	1	1	1	1	1	1
2222222222222222	2222222222222222	2222222222222222	2222222222222222	2222222222222222	2	2	2	2	2	2	2	2	2	2	2
3333333333333333	3333333333333333	3333333333333333	3333333333333333	3333333333333333	3	3	3	3	3	3	3	3	3	3	3
4444444444444444	4444444444444444	4444444444444444	4444444444444444	4444444444444444	4	4	4	4	4	4	4	4	4	4	4
5555555555555555	5555555555555555	5555555555555555	5555555555555555	5555555555555555	5	5	5	5	5	5	5	5	5	5	5
6666666666666666	6666666666666666	6666666666666666	6666666666666666	6666666666666666	6	6	6	6	6	6	6	6	6	6	6
7777777777777777	7777777777777777	7777777777777777	7777777777777777	7777777777777777	7	7	7	7	7	7	7	7	7	7	7
8888888888888888	8888888888888888	8888888888888888	8888888888888888	8888888888888888	8	8	8	8	8	8	8	8	8	8	8
9999999999999999	9999999999999999	9999999999999999	9999999999999999	9999999999999999	9	9	9	9	9	9	9	9	9	9	9

- • •
- Yes
 - No
 - 08428
 - 12-1
 - 11-5

12. The *right-most* position of a field is called the *low-order* column. The *left-most* position of a field is called the *high-order* column. In the following card:
 - a. Column 51 is called the _____ column of the total field.
 - b. Column 48 is the _____ column of the total field.

EMPLOYEE NAME	SEX	SOCIAL SECURITY NUMBER	EMPLOYEE NO.	RATE	DATE	HOURS			EARNINGS			DEDUCTIONS			NET EARNINGS	STATUS
						REGULAR	OVER-TIME	TOTAL	REGULAR	OVER-TIME	GROSS	OASI	WITH. TAX			
00000000000000000000		000000000000000000	000000000000000000	000000000000000000	000000000000000000	000000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
11111111111111111111	M	111111111111111111	111111111111111111	111111111111111111	111111	1111	1111	1111	1111	1111	1111	1111	1111	1111	1111	1111
22222222222222222222	F	222222222222222222	222222222222222222	222222222222222222	222222	2222	2222	2222	2222	2222	2222	2222	2222	2222	2222	2222
33333333333333333333		333333333333333333	333333333333333333	333333333333333333	333333	3333	3333	3333	3333	3333	3333	3333	3333	3333	3333	3333
44444444444444444444		444444444444444444	444444444444444444	444444444444444444	444444	4444	4444	4444	4444	4444	4444	4444	4444	4444	4444	4444
55555555555555555555		555555555555555555	555555555555555555	555555555555555555	555555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
66666666666666666666		666666666666666666	666666666666666666	666666666666666666	666666	6666	6666	6666	6666	6666	6666	6666	6666	6666	6666	6666
77777777777777777777		777777777777777777	777777777777777777	777777777777777777	777777	7777	7777	7777	7777	7777	7777	7777	7777	7777	7777	7777
88888888888888888888		888888888888888888	888888888888888888	888888888888888888	888888	8888	8888	8888	8888	8888	8888	8888	8888	8888	8888	8888
99999999999999999999		999999999999999999	999999999999999999	999999999999999999	999999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999

-
- a. low-order
- b. high-order


13. A data item is the data in a field. Data items contain, just as do the fields from which they are read, high-order and low-order positions. In the following card:
 - a. The high-order digit of the data item punched in the hours regular field is _____.
 - b. The low-order digit of the same data item is _____.

EMPLOYEE NAME	SEX	SOCIAL SECURITY NUMBER	EMPLOYEE NO.	RATE	DATE	HOURS			EARNINGS			DEDUCTIONS			NET EARNINGS	STATUS
						REGULAR	OVER-TIME	TOTAL	REGULAR	OVER-TIME	GROSS	OASI	WITH. TAX			
00000000000000000000		000000000000000000	000000000000000000	000000000000000000	000000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
11111111111111111111	M	111111111111111111	111111111111111111	111111111111111111	111111	1111	1111	1111	1111	1111	1111	1111	1111	1111	1111	1111
22222222222222222222	F	222222222222222222	222222222222222222	222222222222222222	222222	2222	2222	2222	2222	2222	2222	2222	2222	2222	2222	2222
33333333333333333333		333333333333333333	333333333333333333	333333333333333333	333333	3333	3333	3333	3333	3333	3333	3333	3333	3333	3333	3333
44444444444444444444		444444444444444444	444444444444444444	444444444444444444	444444	4444	4444	4444	4444	4444	4444	4444	4444	4444	4444	4444
55555555555555555555		555555555555555555	555555555555555555	555555555555555555	555555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555	5555
66666666666666666666		666666666666666666	666666666666666666	666666666666666666	666666	6666	6666	6666	6666	6666	6666	6666	6666	6666	6666	6666
77777777777777777777		777777777777777777	777777777777777777	777777777777777777	777777	7777	7777	7777	7777	7777	7777	7777	7777	7777	7777	7777
88888888888888888888		888888888888888888	888888888888888888	888888888888888888	888888	8888	8888	8888	8888	8888	8888	8888	8888	8888	8888	8888
99999999999999999999		999999999999999999	999999999999999999	999999999999999999	999999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999

• • •

- a. 6 (column 42)
- b. 0 (column 44)

14. The data items in punched cards can be read by a card reader and stored into main storage. Recall that such an input operation:

- a. Is started by the CPU
- b. Is represented by the flowchart symbol 

• • •

Both

15. A machine called a keypunch can be used to punch data items into cards. The keypunch is usually not connected to a data processing system and, therefore, is not an input device. It is operated in a manner similar to a typewriter.

After the cards are keypunched, they can be put into a card reader. The data items are read from the cards by the card reader and stored into main storage. The punched cards read at the card reader:

- a. Are an input medium
- b. Can contain data that can be entered into the system by means of an input device

• • •

Both

16. Another way to punch data into cards is by means of a card punch. A card punch is an output device, which is controlled by the CPU as the CPU is directed by the program. Cards punched by a card punch can later be read by a card reader. Therefore, such cards can provide input for another data processing job.

A medium that is used for either input or output is called an input/output medium.

The punched card is:

- a. An input medium only
- b. An input/output medium

• • •

- b. An input/output medium

Chapter 4. Data Organization; Terminal and Decision Flowchart Symbols

For directions, refer to the summary on the inside of the front cover.

PRETEST 4

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

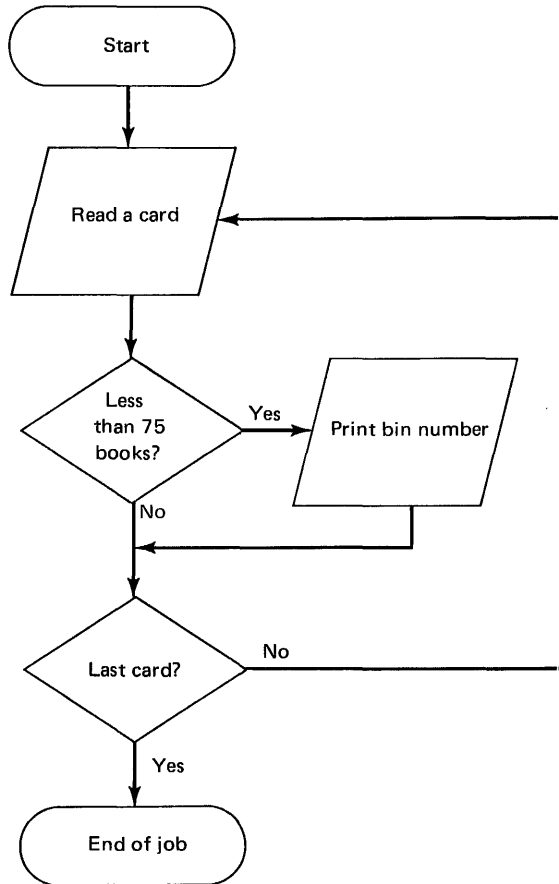
1. Match the two lists:
 - a. Data file
 - b. Record
 - c. Field
 - d. Data item
 - e. I don't know how to match the two lists.
 1. 314728
 2. Collection of related records
 3. Collection of related data items
 4. Location within a record
2. The processing sequence, in most data processing jobs, is:
 - a. Read input field, process, write output field
 - b. Read input record, process, write output record
 - c. Read input data item, process, write output data item
 - d. Read input column, process, write output column
 - e. I don't know
3. Flowchart the following problem statement (use the flowchart symbols printed on the inside back cover as a reference):

Processing of the stock supply job produces a printed list of bin numbers. Each bin number printed on the list specifies a bin that contains less than 75 books. The bin numbers for bins that contain 75 or more books are not printed. Each input card contains a bin number and the number of books in that bin. The job is stopped after the last card is read.
4. A programmer normally writes program instructions:
 - a. In the same order in which the processing operations are to be performed
 - b. In any order
 - c. In non-sequential order
 - d. In non-consecutive order
 - e. I don't know
5. The order of instruction execution can be changed by a program instruction called:
 - a. A judgment instruction
 - b. A jump instruction
 - c. A conditional branch instruction
 - d. A compare instruction
 - e. I don't know
6. Input punched cards that contain related data but do not have the same field arrangements as each other:
 - a. Can never be used for input
 - b. Must be repunched
 - c. Must each be capable of being identified by the program
 - d. All of the above
 - e. I don't know

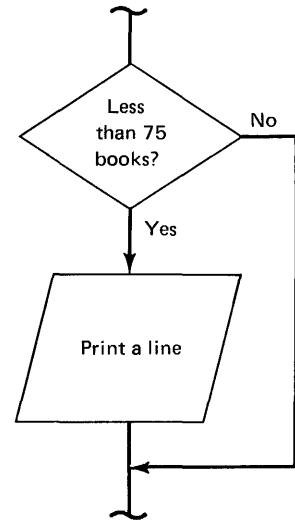
Answers are on the next page.

Answers to Pretest 4

1. a. 2
b. 3
c. 4
d. 1
2. b
- 3.



Note: Your answer should show the same logical operations, even though your wording or arrangement of flowchart symbols is somewhat different than that shown here. For example, in your flowchart you might have shown the sequence



which is just as correct as the answer given here.

4. a
5. c
6. c

Some data files, however, contain more than one record type. A punched card input data file for a billing job, for example, might contain the following record types, each with its own special field arrangement (record layout):

- Sold-to record
- Shipped-to record
- Routing record
- Order record
- Discount record

The billing-job program would handle all of these record types. The same program would probably not be written to handle records of statistical data related to bird migration patterns. Such records would not be related to the billing job.

So that the program can properly process each record in the billing data file, each distinct record type must be identified. A simple way of identifying the different records is to assign a code field. The code field is in the same column(s) in each record, but each record is assigned a different code value.

The preceding records for the billing job could all have a single column, say column 80, as a record-identifier field. The codes might then be:

<i>Record-Identifier Code (Column 80 Punch Position)</i>	<i>Identifies</i>
12	Sold-to record
11	Shipped-to record
0	Routing record
1	Order record
2	Discount record

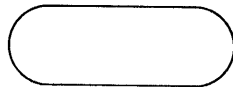
As each record is read at the card reader, the program can determine (through the CPU) the record type by examining the code (in main storage) that is read from column 80.

You are not expected to learn what record types make up a data file for a billing job. You should notice, however, that:

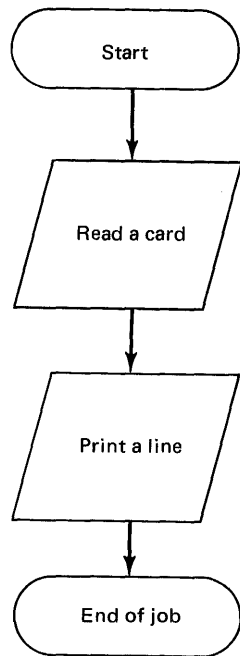
1. All records in an input data file are related. They may not be identical in layout, but they are related to each other.
2. Records that are related, but not identical, must be identified so that they can be distinguished from each other by the program.

Terminal and Decision Flowchart Symbols

A flowchart symbol that is used to indicate the beginning or ending of a sequence of operations is the terminal symbol:

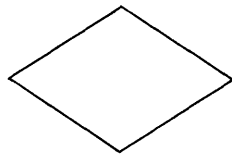


The labels (in this book) that are written in this symbol are start or end of job. Other labels could be used, but start and end of job indicate the normal use of the symbol. Consider, for example, flowcharting of the following problem statement:
Print the data that is read from a card on a single line.



The print-a-line operation would be performed at an output device called a printer. The output medium in the printer is paper. The printer would print for this problem a line of printed-character data on one sheet of paper. The characters printed would be those that correspond to the punched-hole data that is read from the single card. The terminal symbols clearly show the start and end of the job.

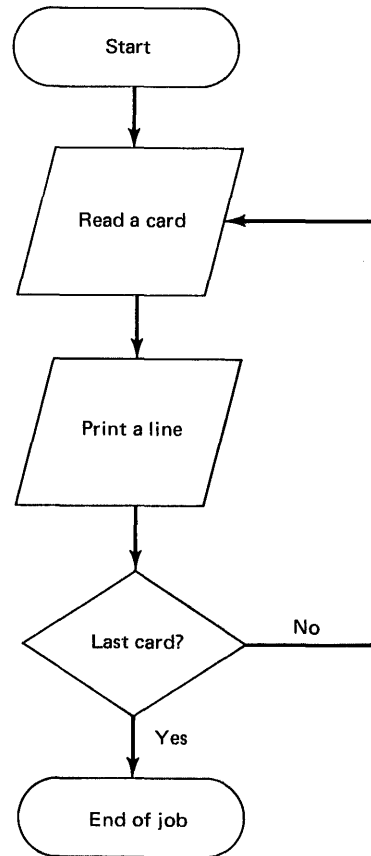
Reading a single card and printing only a single line is not a very realistic problem for a data processing system. In order to flowchart a more realistic problem statement, we need another flowchart symbol, the decision symbol:



We will use the decision symbol to flowchart the following problem statement:

Read each punched-card record of an input file. Print a single line at the printer for each card read; the line of print is a printed-character representation of the data in the card. Stop the job after the last card has been read and the last line has been printed.

Now notice how the decision symbol is used in the flowchart of this problem statement:

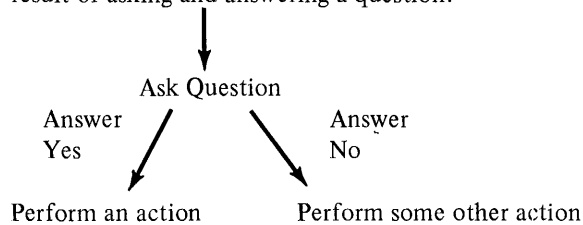


The decision symbol contains the question, "Last card?" which is a shortened form of, "Has the last input card been read at the card reader?"

If the answer to the question is yes (the last card has been read), the job is stopped.

If the answer is no (the last card has not been read), the next card is read.

The decision symbol, then, shows paths to different actions that can be taken as a result of asking and answering a question:



A program instruction that corresponds to such a flowchart symbol is called the branch-on-condition (or conditional-branch) instruction. The branch-on-condition instruction provides for changing the sequence of instruction execution in the CPU. The reasons for having such an instruction are:

1. To save the programmer time when he writes programs
2. More importantly, to use fewer main storage locations in which to store the program

Later we investigate these reasons in more detail. First, recall that before program instructions can be executed by the CPU, they must be in main storage. Now suppose that the program instructions written for the preceding problem statement (the one about printing a line for each card read) are:

<i>Instruction Number</i>	<i>Instruction</i>	
1	Read a card	(These “instructions”, while similar to real instructions in some respects, are <i>not</i> real program instructions.)
2	Print a line	
3	Read a card	
4	Print a line	
5	Read a card	
6	Print a line	
	and so on	
—	Stop	

So far, we have at least seven separate instructions in the program. But instructions numbered 1, 3, and 5 all specify the same operation (read a card), and instructions 2, 4, and 6 all specify the same operation (print a line).

Now suppose that there are 1000 cards in the input file. The program, if it were written in the preceding manner, would require two thousand and one instructions (the one thousand read-a-card and the one thousand print-a-line instructions, plus the one to stop the job). All of these instructions must be loaded into main storage before the program is executed by the CPU. But, by using a branch-on-condition type of instruction, the program can be written as follows:

<i>Instruction Number</i>	<i>Instruction</i>
1	Read a card
2	Print a line
3	Go to instruction 1 if not the last card
4	Stop

Instruction 3 represents a branch-on-condition type of instruction. Written this way, the program can handle any number of input records. Let’s see why.

Normally, most systems store most instructions in main storage in the same sequence as the one in which the instructions are written and executed. Suppose, for example, that the preceding instructions are in the following main-storage locations:

<i>Main Storage Locations Used</i>	<i>For Instruction Number</i>	<i>Instruction Name</i>
1000-1003	1	Read a card
1004-1007	2	Print a line
1008-1011	3	Go to instruction 1 if not the last card
1012-1015	4	Stop

Instruction 1 is read out of main storage and executed by the CPU, then instruction 2, then 3. These instructions are fetched from sequential locations, one after another, and executed in sequence. The operation specified by instruction 3 however, is: “Go to instruction 1 if not the last card.” If the last card has been read, the CPU executes instruction 4 (Stop), after instruction 3.

In other words, instruction 4 is executed in sequence in the same way as instructions 1, 2, and 3, but only after the last card has been read. Otherwise, instruction 3 breaks the “normal” sequence of instruction execution by causing a branch to instruction 1.

Clearly, (when compared to a program without the branch-on-condition) the branch-on-condition instruction used in this program has made it possible:

1. For the programmer to write only five instructions which can be used by the CPU to process the input file regardless of the number of input records
2. To save thousands of locations of main storage

Item 2 may not seem significant at this point. Most application programs that are now in use, however, would not fit into the available main storage in most systems if branching could not be performed.

Notice, from the problems that we have flowcharted so far, that:

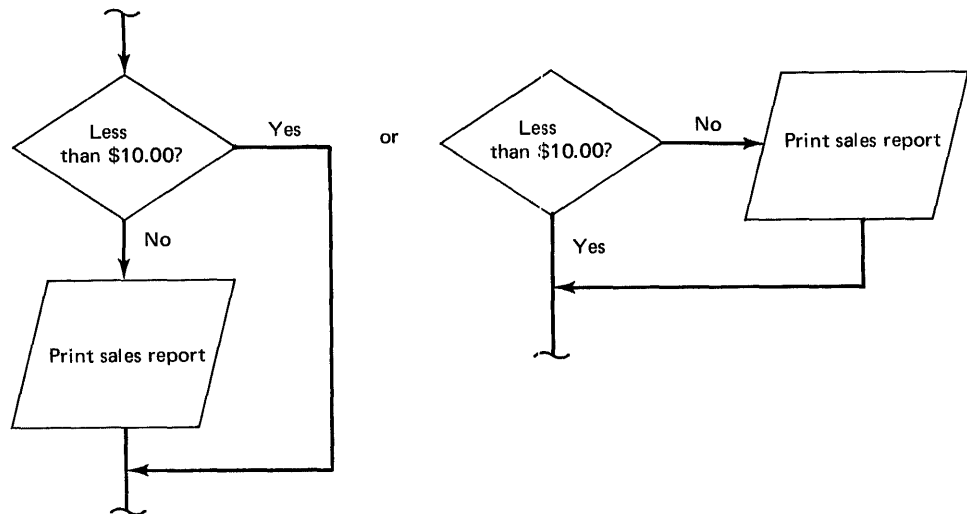
1. Usually one record at a time, such as a single card, is read from an input device.
2. That record is processed.
3. An output record is written to an output device.

In some jobs more than one record may be read before a single record is processed. The preceding, however, is the usual procedure.

Greater-than, Equal-to, and Less-than

Another use of the decision symbol is to represent greater-than, equal-to, and less-than tests, which are characterized in problem statements by sentences such as:

“Any sales report with *less than* \$10.00 in the item-amount field is not printed; otherwise the sales report is printed.”



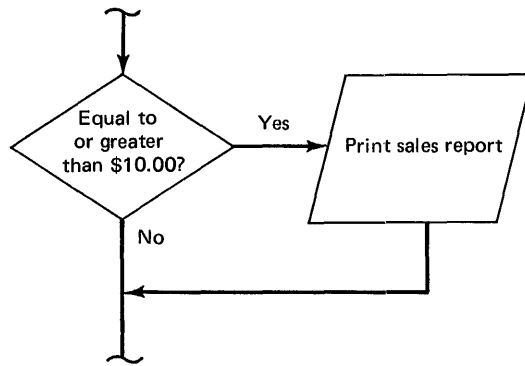
Notice that the two preceding flowcharts are essentially equivalent.

Another way of stating the preceding problem is:

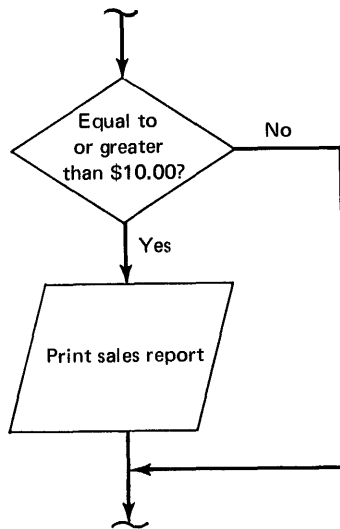
“Any sales report with \$10.00 or more than \$10.00 in the item-amount field is printed; otherwise the sales report is not printed.”

(Notice that the phrase “. . . *with* \$10.00 . . .” is the same as saying “. . . *equal to* \$10.00”.)

In this case, the flowchart, which is equivalent to the preceding flowchart, is:



Another way to draw this same flowchart is:



Compare the two preceding flowcharts. Notice that the actions taken are the same even though the flowcharts are somewhat different. In other words, there is usually more than one way to flowchart a solution to a problem statement. Therefore, do not be concerned if flowcharts you draw for this course are not identical to the “book” answers.

Nevertheless, the most expected operations should normally be drawn in a straight line, from the top to the bottom of a flowchart. Operations that are not frequently expected can be shown to the side of the main data flow. For example, if, in the immediately preceding flowcharts, printing of the sales report is expected for most records, then the print-sales-report operation should be directly under the decision symbol. If, on the other hand, most sales amounts are less than \$10.00, and printing is therefore not performed for most records, then the print-sales-report operation should be shown to the right of the flowchart. The concept stressed here is frequently encountered in both flowcharts and programs. You are advised not to read further until you feel confident that you understand what was just covered.

Showing the most expected operations in this way (top-to-bottom) and less expected operations to the side makes the flowchart easier to read. Changes to the flowchart can therefore be more easily made, because the main data flow is clearly distinguishable from the less expected operations.

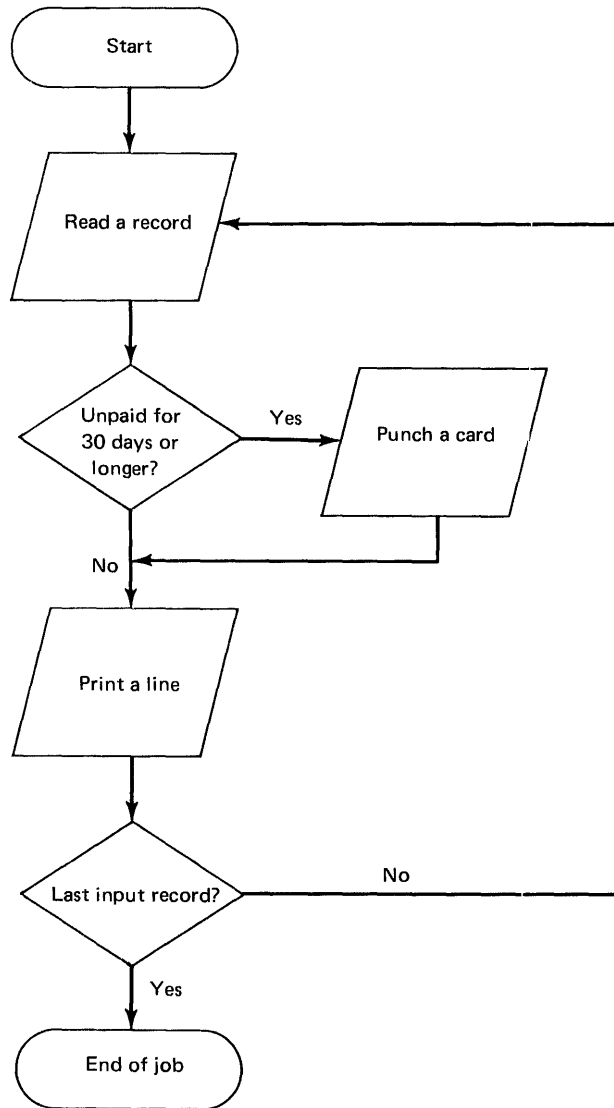
Consider, as a last item in this section, the flowcharting of another problem statement:

An input data file contains only unpaid credit account records. Account records due for 30 days or more are to be punched in cards. All account data is to be printed (one line per input record). The job is to be stopped after the last input record has been read and the last output operation is completed.

This problem statement can be reorganized and rewritten as follows:

- Input* Read a record. (Or, read unpaid credit account record.)
- Process* Unpaid for 30 days or longer?
 Stop the job after the last card is read and the last output operation has been performed.
- Output* Print a line for all records.
 Punch a card for accounts due for 30 days or longer.

The flowchart for the preceding problem statement is:



POST-TEST 4

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. Match the two lists:
 - a. Field
 - b. Record
 - c. Data item
 - d. Data file
 - e. I don't know how to match the two lists.
 1. AJ555671
 2. Location within a record
 3. Collection of related records
 4. Collection of related data items
2. For most data processing jobs, the processing sequence is:
 - a. Read input data item, process, write output data item
 - b. Read input column, process, write output column
 - c. Read input field, process, write output field
 - d. Read input record, process, write output record
 - e. I don't know
3. Flowchart the following problem statement (use the flowchart symbols printed on the inside back cover as a reference):

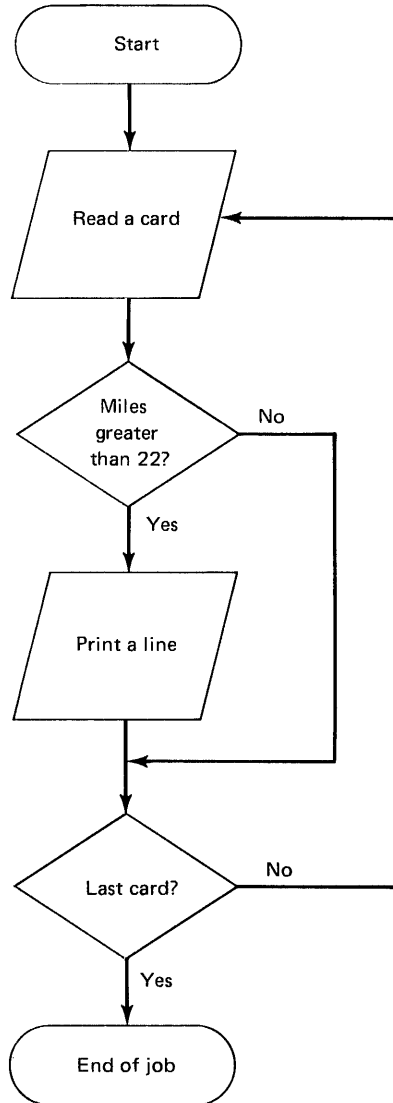
The state road authority requires a printed report of turnpike distances traveled by class-2 vehicles. Input punched card records are *only* for class-2 vehicles. Each input record contains turnpike-distance-traveled data for each toll-paying class-2 vehicle. The job is to be ended after the last card is read.

Only distances greater than 22 miles are to be listed on the printed report. Entering-station and exiting-station information, for each class-2 vehicle, is to be printed next to the distance for that vehicle.
4. Program instructions are normally written by the programmer:
 - a. In non-consecutive order
 - b. In the same order in which the processing operations are to be performed
 - c. In any order
 - d. In non-sequential order
 - e. I don't know
5. A program instruction that is used to change the order of instruction execution is called:
 - a. A conditional-branch instruction
 - b. A jump instruction
 - c. A compare instruction
 - d. A judgment instruction
 - e. I don't know
6. A group of punched cards that have different field arrangements, but contain related input data items:
 - a. Must each be identifiable by the program
 - b. Should be repunched
 - c. Cannot be used for input under any condition
 - d. I don't know

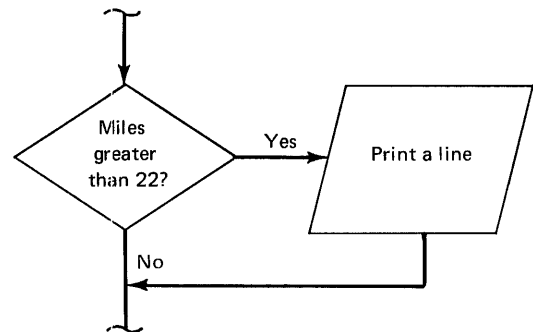
Answers are on the next page.

Answers to Post-Test 4

1. a. 2
b. 4
c. 1
d. 3
2. d
- 3.



Note: Your answer should show the same logical operations, even though your wording or arrangement of flowchart symbols is somewhat different than that shown here. For example, in your flowchart, you might have shown the sequence



which is just as correct as the answer given here.

4. b
5. a
6. a

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 4. DATA ORGANIZATION; TERMINAL AND DECISION FLOWCHART SYMBOLS

1. Each input punched card usually makes up an entire record. The specific items punched in fields of input cards are called data items. Locations in records:
 - a. Are called fields
 - b. Contain data items

• • •

Both
2. Data items in records are related to each other. The names of two related data items in a payroll record are "rate of pay," and "hours worked." Another data item in a payroll record might be named:
 - a. The price of real estate in Minnesota
 - b. Overtime hours worked

• • •

b. Overtime hours worked
3. A collection of related records is called a data file. A list of the names of several data items from records in a payroll data file might be:
 - a.
 - Employee names
 - Deduction amount
 - Rate of pay
 - Overtime hours
 - b.
 - Regular pay amount
 - Hours worked (regular)
 - Selected air pressures of tires
 - Employee number

• • •

 - a.
 - Employee names
 - Deduction amount
 - Rate of pay
 - Overtime hours

4. Two types of punched card records used in a payroll application might have different field arrangements (record layouts) as follows:

Master Record

<i>Field Name</i>	<i>Card Columns</i>
Employee serial number	1-20
Employee name	21-36
Social security number	37-45
Department number	46-49
Rate of pay	50-53
Record type	54 (a master record is specified by a 12 punch in column 54.)

and so on

Weekly Time Records

<i>Field Name</i>	<i>Card Columns</i>
Employee name	1-16
Social security number	17-25
Employee serial number	26-45
Hours worked	46-49
Department number	50-53
Record type	54 (a weekly time record is specified when column 54 contains no punches.)

and so on

Notice that in both types of records (master and weekly time) column 54 is used to identify the record. The two different types of records must be identified so that the program can handle the data items correctly.

Refer to the field names listed above in order to answer the following:

- a. Card columns 50-53 contain rate of pay when column 54 contains_____.
- b. The record is a master record when column 54 contains_____.
- c. When column 54 contains no punches, columns 50-53 make up the_____.

• • •

- a. A 12 punch
- b. A 12 punch
- c. Department number field

5. All records in a data file may or may not have the same field arrangement (record layout). In order for a program to handle several different record layouts:

- a. Each type of record must be identifiable by the program.
- b. An unidentified record must be in punched card form.

• • •

- a. Each type of record must be identifiable by the program.

6. An input data item is processed according to:

- a. The field from which it is read by the program
- b. The particular location of the data item in the record read

• • •

Both

7. The basic operations of reading input data, processing the data, and writing output data are normally handled by reading one input record (or several input records) and processing the record(s). Which one of the following two descriptions is more accurate?
- a. 1. Read the entire data file from the input medium.
2. Process the file.
3. Write the output data file to an output medium.
 - b. 1. Read one (or several) input record(s) from the input medium.
2. Process the record(s).
3. Write output record(s) to the output medium.
- • •
- b. 1. Read one (or several) input record(s) from the input medium.
2. Process the record(s).
3. Write output record(s) to the output medium.
8. Assume that the input data is in a deck of punched cards. The normal method of handling such data is related to the type of job being performed and the requirements of that job. Normally, however, the processing sequence is (choose one):
- a. Read input record, process, write output record.
 - b. Read input data item, process, write output data item.
 - c. Read input field, process, write output field.
 - d. Read input column, process, write output column.
- • •
- a. Read input record, process, write output record. (Data is normally processed a record at a time.)
9. Input data files are made up of many records and are usually:
- a. Processed one record at a time
 - b. Processed all at once
- • •
- a. Processed one record at a time
10. A collection of records makes up a data file. Each record is a collection of data items. Each data item in a specific record can be identified by its location (in the record) which is called a:
- a. Field
 - b. Data file
- • •
- a. Field
11. Match the two lists:
- | | |
|--------------|-------------------------------------|
| a. Data file | 1. Collection of records |
| b. Record | 2. Location within a record |
| c. Field | 3. Collection of related data items |
- • •
- | | |
|--------------|-------------------------------------|
| a. Data file | 1. Collection of records |
| b. Record | 3. Collection of related data items |
| c. Field | 2. Location within a record |

12. A payroll record contains fields, each of which contains a data item. Match the two lists:

- | | |
|--------------|-------------------|
| a. Field | 1. Overtime hours |
| b. Data item | 2. 8.4 |

• • •

- | | |
|--------------|-------------------|
| a. Field | 1. Overtime hours |
| b. Data item | 2. 8.4 |

13. Match the two lists:

- | | |
|--------------|-------------------------------------|
| a. Field | 1. Location within a record |
| b. Record | 2. Collection of related records |
| c. Data item | 3. Collection of related data items |
| d. Data file | 4. 7681246 |

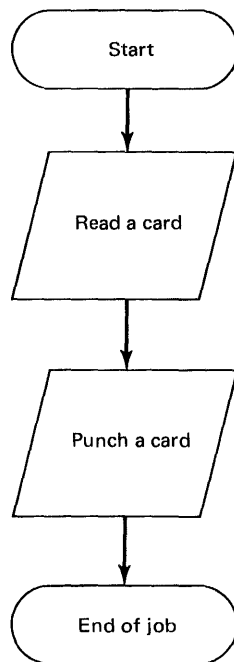
• • •

- | | |
|--------------|-------------------------------------|
| a. Field | 1. Location within a record |
| b. Record | 3. Collection of related data items |
| c. Data item | 4. 7681246 |
| d. Data file | 2. Collection of related records |

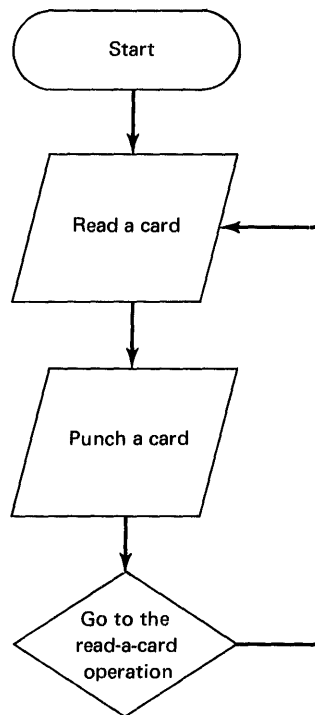
14. The start or end of a flowchart is indicated by the *terminal* symbol. Use the labels start and end of job in the terminal symbols that you draw to flowchart the following problem (refer to the inside of the back cover of this book for a picture of the terminal symbol):

“Read one punched card record, punch the data from that card into another card and then stop the job.”

• • •



15. The flowchart answer to the preceding frame shows the plan for the handling of a:
- a. Record
 - b. Data file
- • •
- a. Record
16. Input records can only be handled one at a time. Therefore, to handle many input records, we add (to the preceding flowchart) an operation that specifies a return to the first operation:




This flowchart:

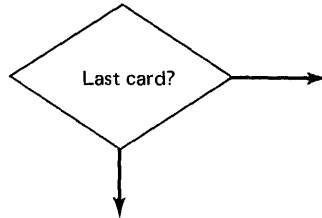
- a. Provides a planned method to end the job
- b. Shows an operation that specifies a return to the read-a-card operation after the punch-a-card operation has been performed

• • •

- b. Shows an operation that specifies a return to the read-a-card operation after the punch-a-card operation has been performed

17. The question of when the job is to be ended can be asked in a *decision* symbol. The decision symbol  represents an operation that asks a question. The paths out of the decision symbol specify what operation(s) should be done as a result of answering the question.

For example, the question “Has the last card been read?” can be represented in the decision symbol as follows:



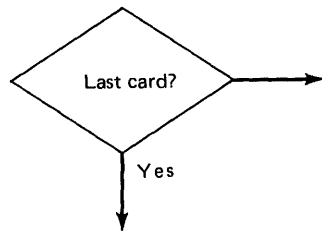
An appropriate answer to “Has the last card been read?” is:

- a. Yes
- b. 15

• • •

- a. Yes

18. Two *paths* lead out of the decision symbol that asks the question “Has the last card been read?” One of these paths can be labelled yes, which indicates that the last card has been read.



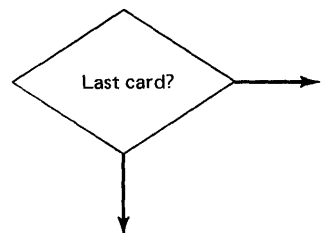
If the last card has not been read, the answer to the “Last card?” question is:

- a. No
- b. Yes

• • •

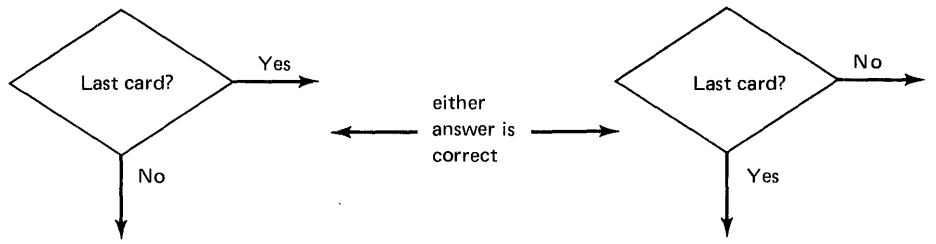
- a. No

19. Label the two flow paths that leave the following symbol:

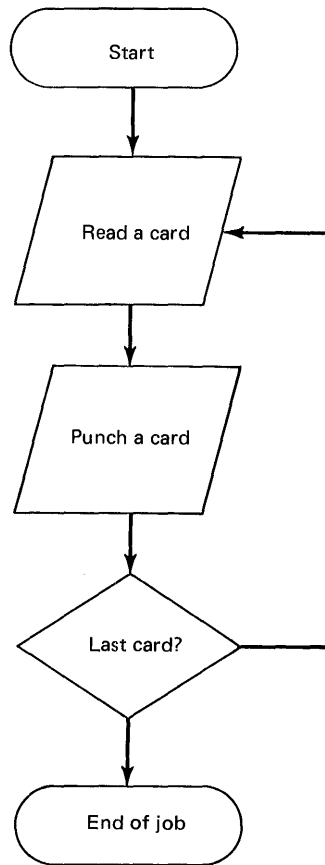


• • •

• • •



20. If the last card has been read, it is desirable not to try to read another card. (There are no more cards to read!) In the following flowchart, two paths leave the decision symbol:



Answer the following with respect to the above flowchart:

- The path going from the decision symbol to the read-a-card input operation should be labelled (Yes/No)_____.
- The path going from the decision symbol to the end-of-job symbol should be labelled (Yes/No)_____.

• • •

- No (The last card has *not* been read.)
- Yes (The end of job occurs after the last card has been read.)

21. It is important to notice that, in the flowcharts we have considered so far, only one card is read at a time. This is the normal processing procedure, to process one record at a time.

A single punched card:

- a. Is usually a single record
- b. Usually contains more than one record

• • •

- a. Is usually a single record

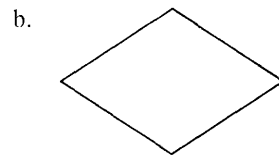
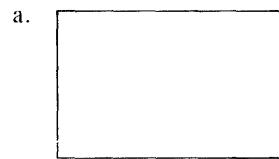
22. The normal sequence (choose one) is:

- a. Read input column, process, write output column.
- b. Read input field, process, write output field.
- c. Read input record, process, write output record.

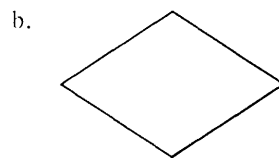
• • •

- c. Read input record, process, write output record.

23. Even though it is usual to “read input record, process, write output record” many times an output record is not written. The decision, as to whether or not to write an output record, can be represented in a flowchart by the symbol:



• • •




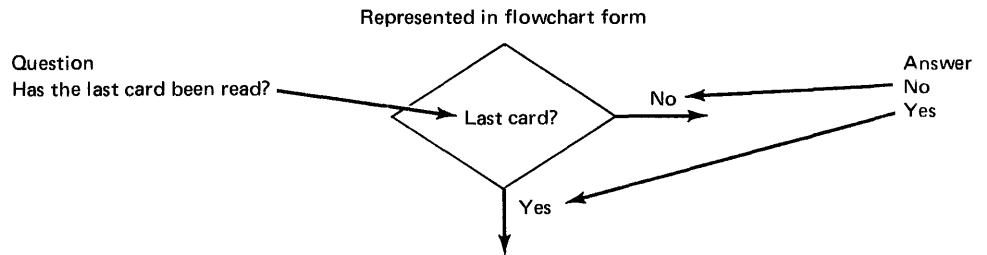
24. The decision symbol represents a type of program instruction called the branch-on-condition instruction (or conditional-branch instruction). This instruction performs an operation similar to that represented by the decision symbol by:

- a. Changing the sequence of program-instruction operations
- b. Testing a condition and, as a result of the test, determining which instruction is to be executed

• • •

- Both

25. The symbol  represents:
- A conditional branch operation
 - An I/O operation
-
- A conditional branch operation
26. The programmer:
- Writes instructions in the same logical order in which processing operations are generally to be carried out
 - Writes a branch-on-condition instruction at a point where a condition must be tested to determine which instruction is to be executed next
-
- Both
27. The programmer writes program instructions in the same order as the one in which the specific processing operations are to be performed. He uses a branch-on-condition type of instruction:
- So that he does not have to rewrite a series of instructions every time the operations performed by that series are required
 - So that the CPU can process a particular sequence of instructions according to the result of a test of a condition
-
- Both
28. A question and its answers are shown in flowchart form as follows:



Notice that the answers are written right next to the data flow lines.
 Show the following question and its answers in flowchart form:

Question

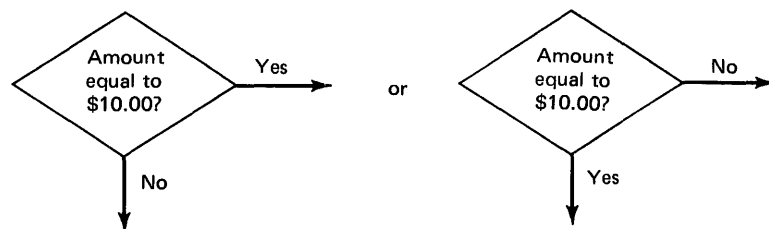
Is the amount equal to \$10.00?

Answers

No

Yes

•••



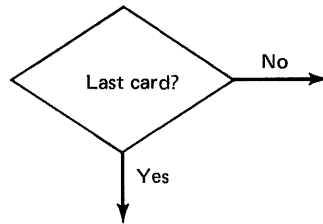
Your wording in the decision symbol may differ from that shown.

29. Flowchart the following question and its possible answers by using a decision symbol. Label the data flow lines (that leave the decision symbol) with the answers to the question. You must provide the answers.

Question

Has the last card been read?

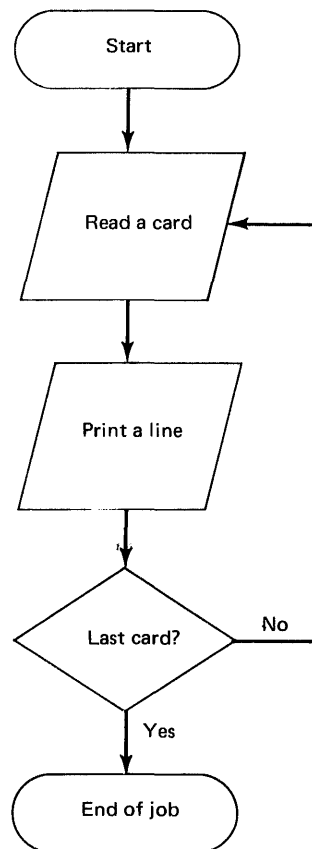
• • •



30. Flowchart the following problem statement:

The data in a deck of cards is to be printed, a single line for each card. A single card is read at a time, the data from that card is printed, and then the next card is read. After the data from the last card has been printed by the printer, the job is completed.

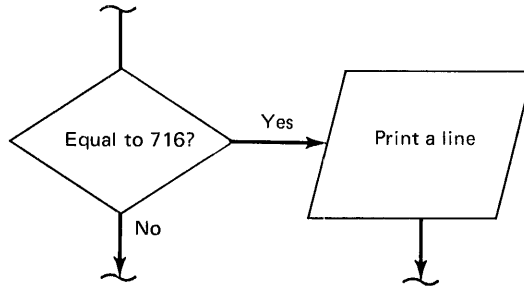
• • •



31. Another variety of condition that is frequently tested is represented by the questions:

- Less than?
- Equal to?
- Greater than?

For example, the problem “all data items equal to 716 are to be printed” can be shown in the flowchart segment:



The no path in the preceding flowchart is taken for all data items that are (your own words)_____

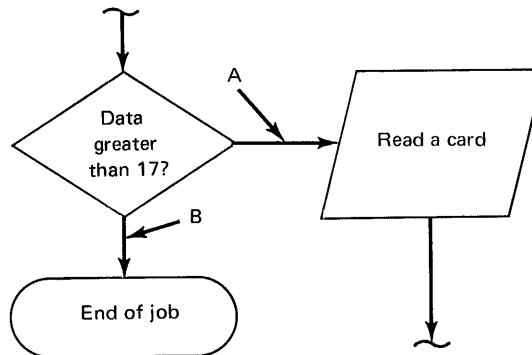
• • •

greater than or less than 716

32. A problem statement segment is:

For data items greater than 17, read a card; otherwise, stop the job.

The corresponding flowchart segment is:



By reading the problem statement segment, you can determine that:

- a. Path A should be labeled (Yes/No)_____.
- b. Path B should be labeled (Yes/No)_____.
- c. The job is ended for data items that are (in your own words)_____.

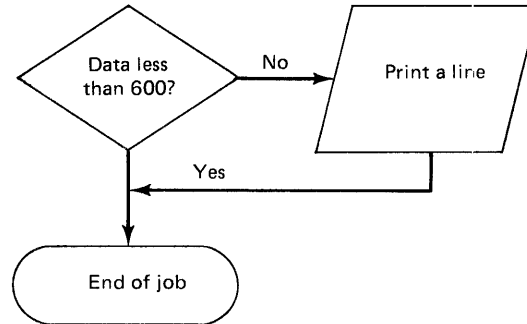
• • •

- a. Yes
- b. No
- c. Less than or equal to 17

33. Flowchart the following problem statement segment (start your flowchart with a decision symbol):

For data items less than 600, stop the job; for data items equal to or greater than 600, print a line and then stop the job.

• • •



34. Reorganize the following problem statement into input, process, and output steps. (Write your reorganized statement on a separate piece of paper. Do not flowchart this problem. Reorganize the statement only.)

XYZ Company uses a printed report that contains name, address, and amount owed for each person who owes more than \$10.00. Nothing is printed for persons who owe \$10.00 or less. The printed report is produced from punched card input records. Each card contains name, address, and amount owed for only one person. After the last card is read and the last line printed, the job is stopped.

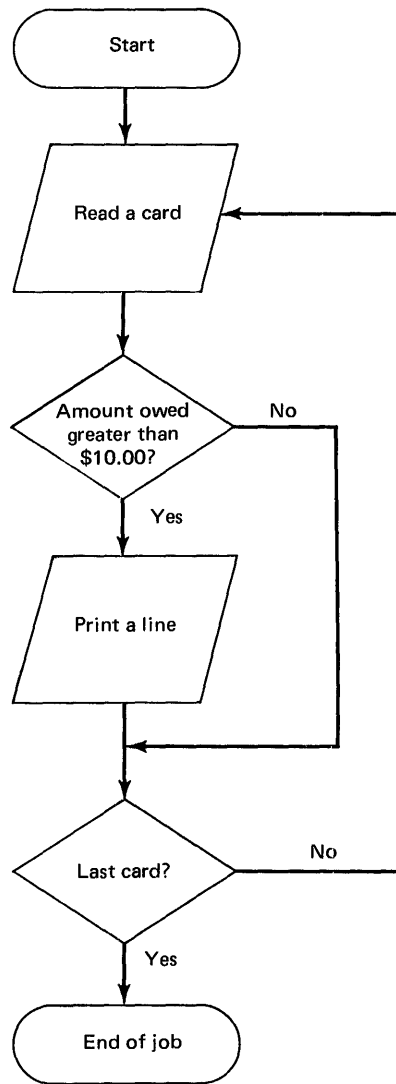
• • •

<i>Input</i>	Read a card
<i>Process</i>	Amount owed greater than \$10.00? After the last card is read, stop the job.
<i>Output</i>	Print a line for each amount owed that is greater than \$10.00.

Note: The following frames to the end of this section require more time and thought than those that you have read so far in this course. The reason is that these frames involve problem-solving concepts that require you to relate several ideas simultaneously. Therefore:

1. Read slowly and carefully.
2. Draw flowcharts or flowchart steps on a separate piece of paper.

35. A correct flowchart of the problem statement of the preceding frame is:



Answer the following questions by referring to this flowchart:

- Is the question "amount owed greater than \$10.00?" asked after each card is read?
- The object of the job is to print a line for each amount owed that is greater than \$10.00. Should a line be printed for amounts that are equal to or less than \$10.00?
- If a line is not printed for a card that was read, is the "Last card?" decision step performed anyway?

•••

- Yes
- No
- Yes

36. Please have on a separate piece of paper your reorganized statements for the problem statement that was just flowcharted. You need them for reference in the next several frames.

We will examine, step-by-step, how the flowchart for the preceding problem statement was developed. (The flowchart is shown in its entirety in the preceding frame.) You should attempt to answer the following frames, however, *without* looking at the flowchart in the preceding frame. Use only the reorganized statements from the original problem statement.

No response required. Go on to the next frame.

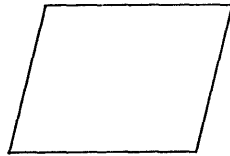
• • •

37. Recall that the *first* symbol in a flowchart is:

- a. A terminal symbol



- b. An input symbol



• • •

- a. A terminal symbol



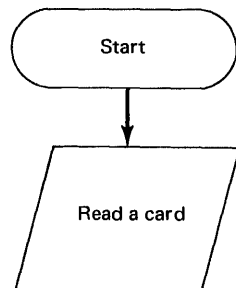
38. In the reorganized problem statement the first operation is:

Input Read a card.

Draw the beginning of the flowchart to show the read-a-card operation.

Note: Have plenty of room to complete the flowchart.

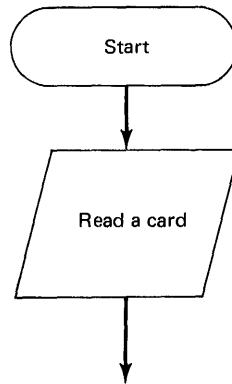
• • •



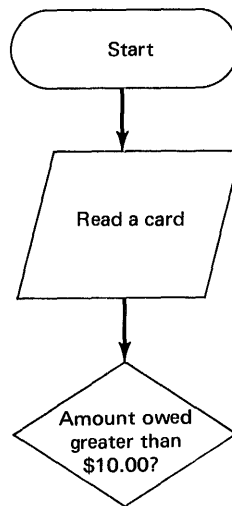
39. The next operation in the reorganized problem statement is:

Process Amount owed greater than \$10.00?

Add the flowchart step that specifies this operation (not the resulting actions) to the flowchart we have developed so far:



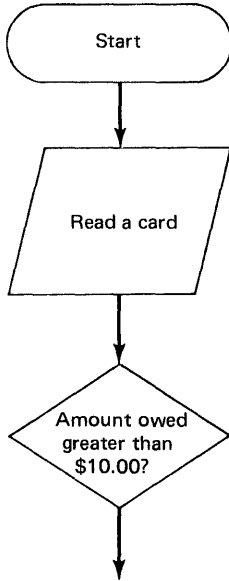
...



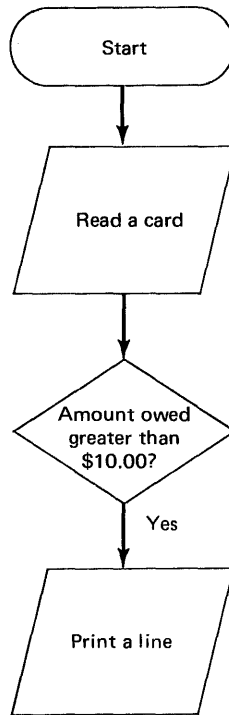
40. The reorganized problem statement indicates:

Output Print a line for each amount owed that is greater than \$10.00.

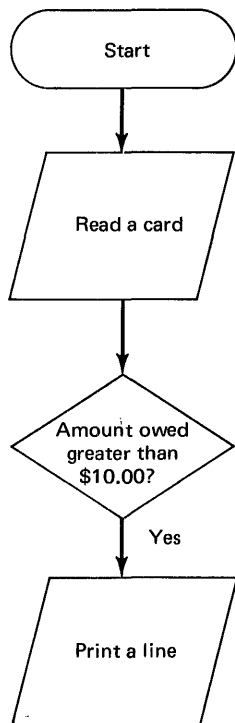
Draw a labeled flowchart symbol that indicates such an action. Label the data flow line that leaves the decision symbol with a yes; make this line go to the symbol you just added to show the output operation.



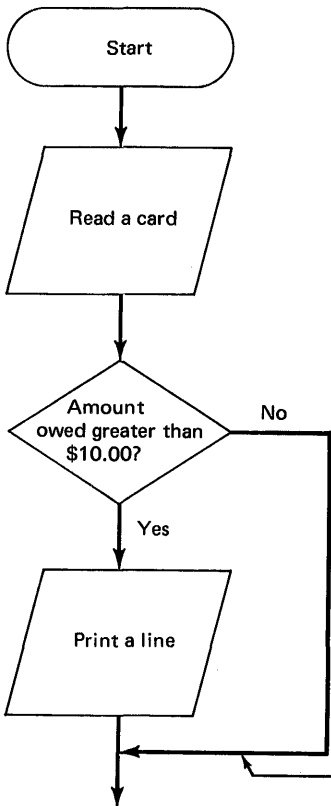
...



41. The problem statement indicates that nothing should be printed if the amount owed is *equal to* or *less than* \$10.00. Draw and label the data flow line that leaves the decision symbol to show this path:



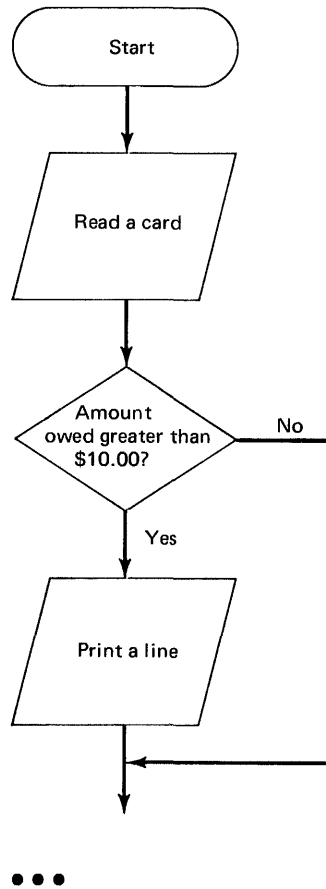
• • •



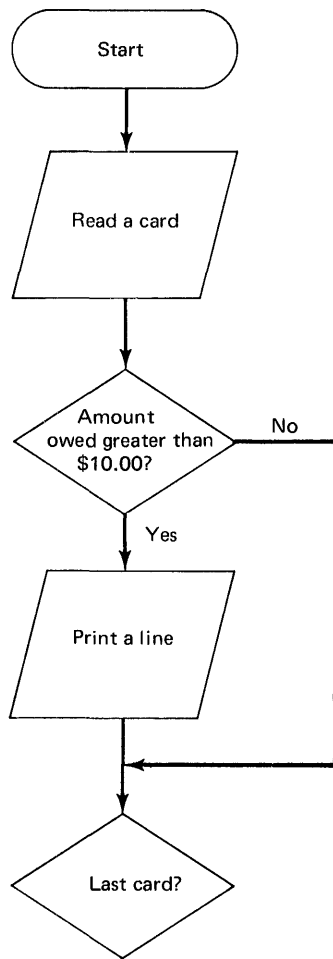
The no path merely skips around the print-a-line operation for amounts less than or equal to \$10.00. Notice that it is not correct for the no path to go directly to the read-a-card operation. Do not forget that a last-card test must be made first. We examine this in the next couple of frames.

Note: It is important for your answer to show where the no data flow line from the decision symbol goes.

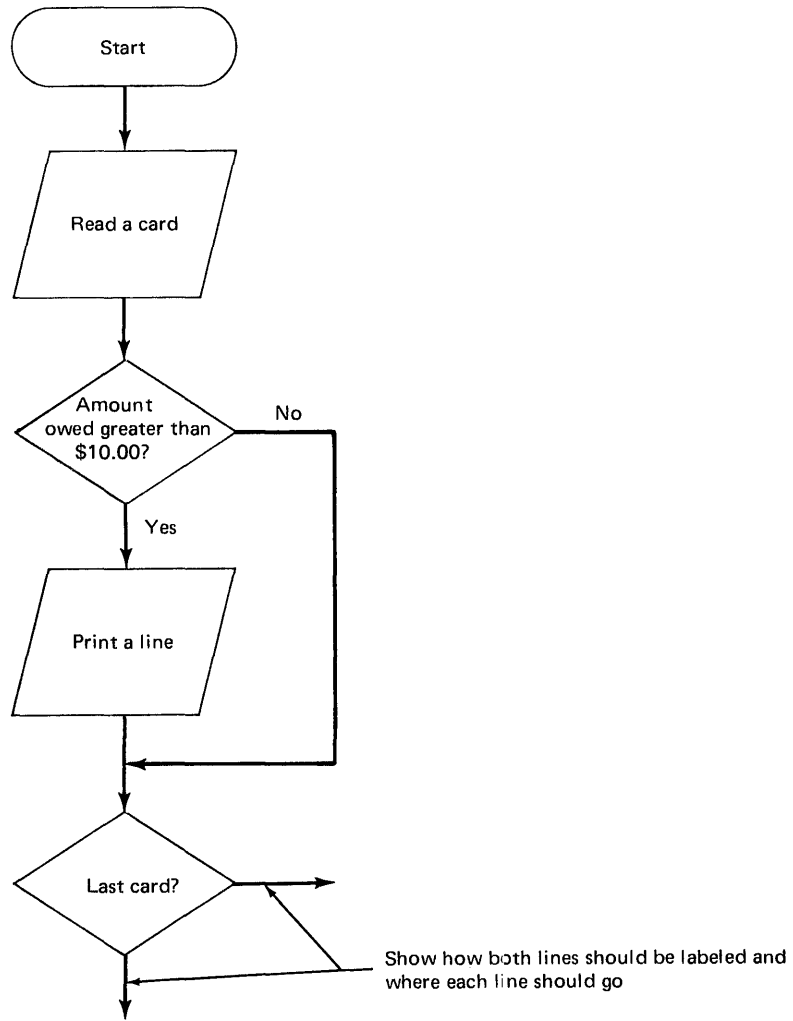
42. The problem statement indicates that the job is to be stopped after the last card is read, and the last line is printed. Note, however, that the test for a last card should be made whether or not a line is printed.
Show, by use of a properly labeled decision symbol, the test for the last card.
Do *not* yet show where the data flow lines from this block should go.



• • •

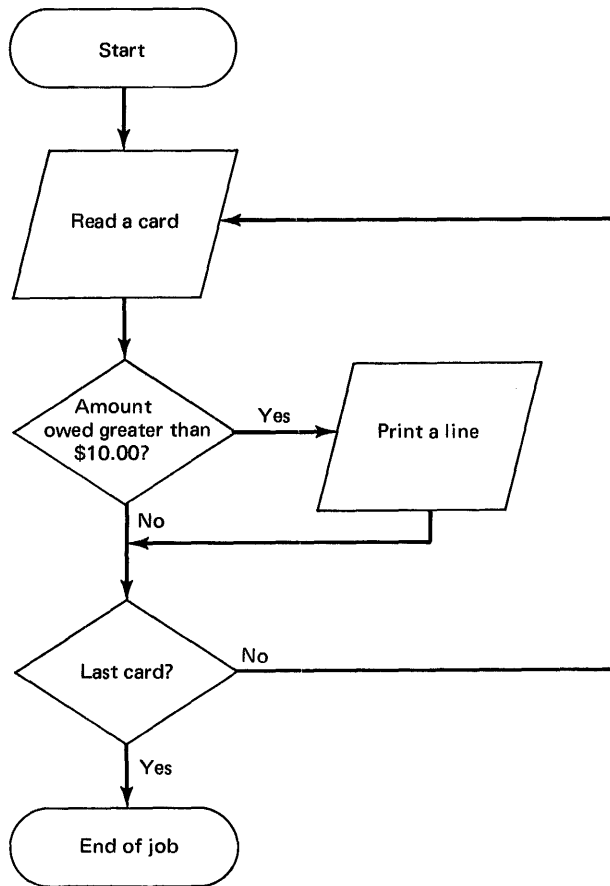


43. If the last card has been read, the job should be stopped (end of job). If the last card has not been read, the next card should be read. Label the data flow lines coming out of the “Last card?” symbol and show where they should go. Indicate, by a properly labeled flowchart symbol, the end-of-job step.



•••

•••



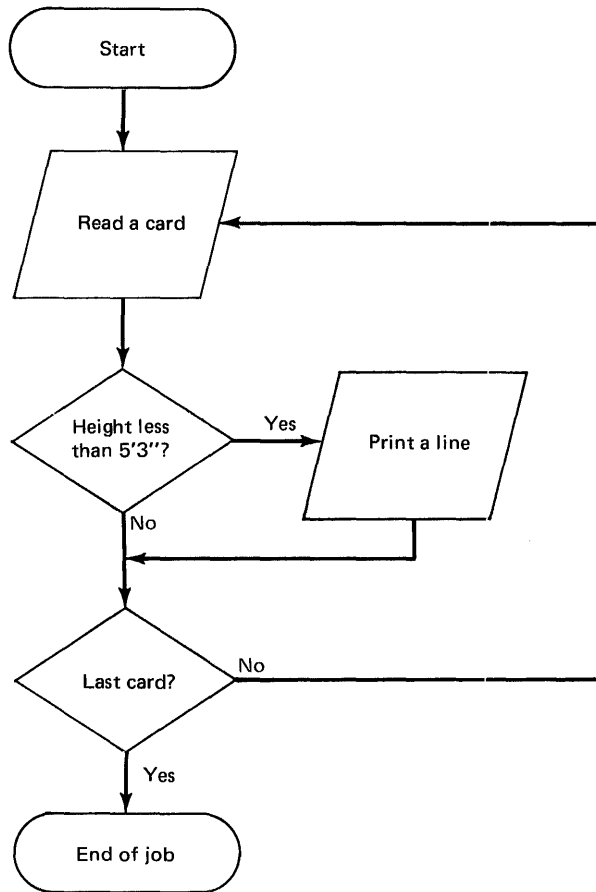
44. Flowchart the following problem statement:

A printed report of all ABC Company employees shorter than 5 feet 3 inches is to be prepared. Physical characteristics, names, and addresses of all employees are on input card records (one card per employee). The name, address, and height of an employee is printed on one line, but only when the employee is shorter than 5 feet 3 inches. Nothing is printed on the report for employees who are 5 feet 3 inches or taller. Stop the job after the last card has been read and printing operations have been completed.

Hint: First reorganize the problem statement into input, process, and output statements, then flowchart the reorganized statements.

•••

...



Chapter 5. Job Overview: From Planning to Output Data; Programming Languages

For directions, refer to the summary on the inside of the front cover.

PRETEST 5

Please write your answers on a separate piece of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. Compilation is:
 - a. The process of translating a source program to an object program
 - b. The instruction that is used to complete a process-to-output sequence
 - c. The branching path that leads to end of job
 - d. Used only in commercial data processing jobs
 - e. I don't know
2. A PL/1 compiler can be used to produce an object program from a source program written in:
 - a. FORTRAN
 - b. RPG
 - c. COBOL
 - d. None of the above
 - e. I don't know
3. The programmer writes programs by using symbols which:
 - a. Are more meaningful to the programmer than are machine-language numeric symbols
 - b. Can be processed, after being put in machine-readable form, by the compiler for the programming language being used
 - c. Allow the programmer to concentrate on the solution to the problem rather than on such tasks as assigning or relocating instruction sequences to specific main storage locations
 - d. All of the above
 - e. I don't know
4. Match the two lists:

a. PL/1	1. Preparation of reports
b. Assembler	2. Scientific
c. COBOL	3. Scientific and business
d. FORTRAN	4. Business
e. RPG	5. One-for-one
f. I don't know how to match the two lists	
5. Relist the letters to show what must be done first, second, third, and so forth:
 - a. Compiler is loaded into main storage and source program is read at card reader.
 - b. Object program is loaded into main storage, job input data is read and then processed, and job output data is produced.
 - c. Flowchart is drawn.
 - d. Object program is result of compilation.
 - e. Source program is written and then keypunched.

6. Testing, by use of test data, should be performed on:
- a. The object program for the solution of a problem
 - b. The flowchart for the solution of a problem
 - c. Both of the above
 - d. Neither of the above
 - e. I don't know

Answers are on the next page.

Answers to Pretest 5

1. a
2. d (PL/1)
3. d
4. a. 3
b. 5
c. 4
d. 2
e. 1
5. c
e
a
d
b
6. c

DESCRIPTIVE MATERIAL FOR CHAPTER 5. JOB OVERVIEW: FROM PLANNING TO OUTPUT DATA; PROGRAMMING LANGUAGES

From Flowcharting to Programming

After the flowchart for a problem statement has been produced, that flowchart is tested for errors. To do this, the programmer:

1. Collects representative input-data examples (test data)
2. Processes the test data “by hand” according to the logical steps in the flowchart

If all test data cannot be processed properly, according to the logical steps in the flowchart, then the flowchart must be revised. Testing of a flowchart is an important step in planning the solution of a problem. After the flowchart is successfully tested, the process of producing a program can be started.

Programmers write programs by using defined sets of symbols that are similar in many respects to those the programmer would use to solve the problem if he did not have a data processing system. Such a set of defined symbols and the rules for using them is called a *programming language*.

But a CPU executes only *machine-language* instructions, which are *not* the same as the programming-language instructions written by the programmer. Machine-language instructions are usually represented as series of numbers. For example, the following numbers represent *four* instructions in a machine-language program:

```
01000001001100001100000011001000
01011000111100001100000000011100
0000010111101111
111100001100100001110000010110001101000011110000
```

It is not necessary for you to know what specific instructions (in what particular machine) are represented by these numbers. You should merely notice that such long lists of numbers are not very meaningful to people. Fortunately, a method exists for producing machine-language instructions from the programming languages without the programmer writing them. They must be produced because the CPU can execute only machine-language instructions.

Note: The machine-language instructions are in the form of electrical signals when they are being executed by the CPU. Of course, people do not write instructions by using electrical signals. Such signals can be “translated” from numbers.

The electrical signals used by machines are translated from numbering systems that are different than the one people usually use. People normally use what is called the decimal numbering system. This system has ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Consequently the decimal system is called a numbering system to the base ten. The base of any numbering system is the same as the number of symbols in that system. Our normal numbering system (base ten) was, perhaps, developed as a result of our having ten fingers. If we had only eight fingers, we might use the octal system which has eight symbols.

Most data processing systems use the binary numbering system which has two symbols (0 and 1) and, therefore, a base of two. As already pointed out, the machines operate on electrical signals. Such signals are easily “translated” from the binary numbering system because of the function of most electrical switches. Such switches, of which there are many varieties in data processing machines, can be either on or off. Hence, we need only two symbols:

<i>Switch</i>	<i>Symbol</i>
on	1
off	0

Other numbering systems can be used to “translate” the electrical signals of machines to a more easily understood numeric form. However, people do not write instructions for machines by using long strings of numbers.

There are jobs in the data processing industry for which it is sometimes convenient to know one or more numbering systems other than the decimal system. We suggest, however, that you wait until you require such information before studying numbering systems. If you feel that you require such information, you may wish to order the book *Number Systems* (Order No. GC20-1618), which is not a requisite for completing this course.

Machine-language instructions can be produced as output data by a data processing system. A certain type of program translates instructions written by a programmer into machine-language instructions. The program that does the translating is generally called a *compiler*. The process of translating is called compilation. (Other terms, such as *assembling*, *generating*, and *translating* are frequently used instead of the term compilation. Each of these terms has a slightly different meaning. They all, however, convey the same basic idea: production of a machine-language program from a program written, in a programming language, by a programmer. In this book, we use the term compilation to refer to this process.)

Many tedious, time-consuming tasks are avoided when a program is written in a programming language rather than in actual machine language. You are not expected to learn what these tasks are. An example is provided here merely to clarify what is meant by “tedious, time-consuming tasks.”

Suppose that, instead of using a programming language, a programmer writes an actual machine-language program. Four instructions from this fictitious program and the main storage locations at which each instruction is assigned by the programmer are:

*Addresses of
Main Storage*

<i>Locations</i>	<i>Instruction</i>
1000-1003	10000000101111111000000011001101
1004-1005	1011010011000000
1006-1007	0010111110000000
1008-1011	10000000101111111000100011001101

Recall that most instructions must be in sequential locations, one right after the other, in order for the program to be executed properly by the CPU.

Further, suppose that:

1. The entire program is *four-thousand* instructions long.
2. Some of the instructions take up two locations, some four locations, some six locations, and some eight locations of main storage.
3. Each instruction must be assigned to its main-storage location by the programmer.

Now, imagine what the programmer must do if, after the program is written, he discovers that in order to make the program work properly one *extra* instruction must be put between the instructions at locations 1004-1005 and 1006-1007 (see the previous location chart). But there is no room for the new instruction! Therefore, the programmer must put the new instruction into storage starting at location 1006. Then he must reassign, by hand, all of the following instructions (thousands of them!) to new locations. This is a tedious, time-consuming task, a task that is performed by a compiler. To summarize:

1. Programming languages allow the programmer to write programs in symbols that are meaningful to him. He can, thereby, concentrate on solving problems rather than on performing tedious, time-consuming tasks, such as relocating program instructions by hand.
2. A program called a compiler is used to produce an actual machine-language program from the program written (in a programming language) by the programmer.

Note: You will not be asked to write a program in this course. This course helps *prepare* you to take a course in a programming language.

Programming Languages

Historically, programming languages have been developed to meet a need required by a particular class of applications. FORTRAN, for example, is a programming language that was developed for the class of applications related to scientific problems. (The term FORTRAN comes from FORMula TRANslation.) FORTRAN symbols and their rules of usage are very similar to the symbols used in mathematical formulas (the language of scientific problems). A segment of a FORTRAN program is shown in Figure 2.

A programming language developed primarily for use in business applications is COBOL. (The term COBOL comes from COMmon Business Oriented Language.) An example of part of a program written in COBOL is shown in Figure 3.

In recent years, PL/1 has been developed. This versatile programming language is applicable to both business and scientific problems. A series of statements, which make up part of a PL/1 program, is shown in Figure 4.

Another programming language, originally developed primarily for the generation of reports, is RPG (Report Program Generator). Modifications and additions to RPG have been made so that now it is a general purpose language. Figure 5 shows a typical RPG program segment as written, by a programmer, on an RPG specifications sheet.

Each programming language has its own compiler. A PL/1 compiler can be used to compile instructions written in PL/1 by the programmer. The same compiler could not be used to compile instructions written in FORTRAN. A FORTRAN compiler must be used.

PL/1, RPG, FORTRAN, and COBOL are high-level languages. Each instruction or statement written by the programmer in such a language frequently produces, as a result of compilation, more than one machine-language instruction.

A programming language in which each instruction (written by the programmer) has one machine-language instruction counterpart is called a one-for-one programming language. An example of such a language is Assembler. A sample of instructions written in the Assembler language is shown in Figure 6. (Sometimes, even in Assembler, more than one machine-language instruction is produced for certain kinds of programmer-written statements. The one-for-one description is sufficient for our purposes here.)

Other programming languages have been developed. For now, it is sufficient for you to know that:

1. PL/1 is applicable to both scientific and business problems.
2. RPG is applicable to the preparation of reports.
3. FORTRAN is used primarily for scientific problems.
4. COBOL is used primarily for business problems.
5. Assembler is an example of a one-for-one programming language; generally one machine-language instruction is produced for each program statement written by the programmer.
6. Compilation can be performed only by the compiler for the programming language used. For example, a COBOL compiler is applicable only to a program written in COBOL.

From Source Program to Job Output Data

A program written in a programming language is called a *source* program. Before a source program can be compiled, it must be put into machine-readable form. Key punching, for example, can be used. A source program, handwritten on program coding sheets, is given to a keypunch operator. Key punching then produces punched cards which can be read as input data at a card reader. The source program must then be compiled.

Before it can be executed, the compiler must be loaded into main storage, just as any other machine-language program. After the compiler is in main storage, the source program (in punched cards) can be read as input data. Compilation is then performed to produce an *object* program. The object program is a program in actual machine-language form. It is the result (output data) of compiling the source program.

PROGRAM		PUNCHING INSTRUCTIONS		PAGE	
PROGRAMMER	DATE	GRAPHIC	PUNCH	PAGE	OF

STATEMENT NUMBER	FORTRAN STATEMENT	IDENTIFICATION SEQUENCE
10	IF(N)40,30,10	
	NFACT=1	
	I=N	
20	NFACT=NFACT*I	
	I=I-1	
	IF(I)40,40,20	
30	NFACT=1	
40	(continue on in the program)	

Figure 2. Portion of a Program Written in FORTRAN

SYSTEM		PUNCHING INSTRUCTIONS		PAGE	
PROGRAM	DATE	GRAPHIC	PUNCH	PAGE	OF

SEQUENCE	CONT	A	B	COBOL STATEMENT	IDENTIFICATION
1	3	4	6	7	8
				WORKING-STORAGE SECTION.	
				01 HEADING-CONSTANTS.	
				02 B1. PICTURE X. VALUE SPACES.	
				02 C1. PICTURE X(5). VALUE 'C.NO.'	
				02 B2. PICTURE XX. VALUE SPACES.	
				02 C2. PICTURE X(24). VALUE 'CUSTOMER NAME'	
				02 B3. PICTURE XX. VALUE SPACES.	
				02 C3. PICTURE X(14). VALUE 'INVOICE NUMBER'	
				02 B4. PICTURE XY. VALUE SPACES.	
				02 C4. PICTURE X(11). VALUE 'DOLLARS DUE'	
				02 B5. PICTURE X(3). VALUE SPACES.	
				02 C5. PICTURE X(8). VALUE 'DATE DUE'	
				02 B6. PICTURE X(61). VALUE SPACES.	
				PROCEDURE DIVISION.	
				BEGIN-RUN	
				OPEN INPUT ACCOUNTS-RECEIVABLE. OUTPUT ACCOUNT-LIST: MOVE	
				SPACES TO ACCOUNT. PERFORM OVERFLOW-HEADING-ROUTINE.	
				PROCESS-DATA.	
				READ ACCOUNTS-RECEIVABLE. AT END. CLOSE ACCOUNTS-RECEIVABLE.	
				ACCOUNT-LIST. STOP RUN. MOVE I-CUSTOMER-NUMBER TO	
				Q-CUSTOMER-NUMBER. MOVE I-CUSTOMER-NAME TO Q-CUSTOMER-NAME.	
				MOVE I-INVOICE-NUMBER TO Q-INVOICE-NUMBER. MOVE I-AMOUNT TO	
				Q-AMOUNT. MOVE I-DUE-DATE TO Q-DUE-DATE. IF AT-BOTTOM-OF-FORM	
				PERFORM OVERFLOW-HEADING-ROUTINE. WRITE ACCOUNT AFTER	
				SINGLE-SPACE. GO TO PROCESS-DATA.	
				OVERFLOW-HEADING-ROUTINE.	
				WRITE ACCOUNT AFTER SINGLE-SPACE. WRITE ACCOUNT FROM	
				HEADING-CONSTANTS AFTER SKIP-TO-NEXT-FORM. MOVE SPACES TO	
				ACCOUNT. NOTE THAT WRITE FROM WILL CAUSE A MOVE AND WRITE.	

Figure 3. Portion of a Program Written in COBOL

```
XCHRT:  PROCEDURE OPTIONS (MAIN):
/* FILE DECLARATIONS */
DECLARE PRTFIL FILE OUTPUT RECORD
ENVIRONMENT (F(132) MEDIUM (SYSLST,1403)):
/* I-O AREA DECLARATIONS */
DECLARE 1 OUTPUT,
2 NAME CHARACTER (9),
2 RATE PICTURE '2229V.99',
2 FILL CHARACTER (116);
DECLARE 1 HEAD,
2 NAMET CHARACTER (9),
2 FILA CHARACTER (7),
2 FILL CHARACTER (116);
/* ELEMENT VARIABLES */
DECLARE NAMTAB (20) CHARACTER (9),
RATTAB (20) PICTURE '9999V99',
PROD DECIMAL FLOAT (8);
/* PROCESSING BEGINS */
LOOP:   ICT = 0;
OPEN FILE (PRIFIL);
A:      GET EDIT (NAMTAB (ICT+1), RATTAB (ICT+1))
        (A(9), F(6.2) );
```

Figure 4. Portion of a Program Written in PL/1

IBM

International Business Machines Corporation

Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Page

1	2
---	---

Program Identification

75	76	77	78	79	80
----	----	----	----	----	----

Line	Form Type	Filename	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators			Field Name	Edit Codes Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Edit Codes						Sterling Sign Position				
								Commas	Zero Balances to Print	No Sign					CR	-	X = Remove Plus Sign	Y = Date	Z = Field Edit	Z = Zero Suppress					
																						And	And	Not	Not
01	O	P R I N T			2		1																		
02	O		O																						
03	O																								
04	O												3												
05	O												3												
06	O												6												
07	O		O		2																				
08	O																								
09	O																								
10	O																								
11	O																								
12	O		T		3																				
13	O																								
14	O																								
15	O																								
	O																								

Figure 5. Portion of a Program Written in RPG

Usually, the compiler is also capable of producing a listing that is a printed copy of:

1. The original source statements written by the programmer
2. The object program produced during compilation
3. Error messages produced as a result of the compiler detecting errors in the source program

The object program is available in machine-readable form (such as punched cards) as it is produced as output data by the compiler.

The object of the entire process, so far, is to produce an object program. The steps are:

1. Problem statement is given to programmer.
2. Flowchart is drawn, tested with test data, and revised if necessary.
3. Source program is written.
4. Source program is put into some machine-readable form, such as punched holes in cards.
5. Compiler is loaded into system.
6. Source program is read at an input device (such as a card reader).
7. Object program (and related listing) is produced as output data, such as in a deck of punched cards, by compiler.

Any errors detected by the compiler must be corrected by the programmer. The listing is read by the programmer to determine such errors. After correction of the source program, compilation is performed again to produce a corrected object program.

The object program is then loaded into main storage so that it can process input test data. The results of processing the test data can be examined by inspecting the output data. Any errors or unusual results may call for program revision. Few object programs operate correctly on the first test run.

After a fully corrected object program is produced, the job can finally be run.

Again, the object program is loaded into main storage. Job input data is then read from an input device, that data is processed, and job output data is produced.

IBM Printed in U.S.A.

IBM System/360 Assembler Coding Form

PROGRAM MORTGAGE PAYMENT TRANSACTION REPORT	PUNCHING INSTRUCTIONS	GRAPHIC	PAGE	OF	CARD ELECTRO NUMBER
PROGRAMMER (STANDARD INSTRUCTION SET)	DATE	PUNCH			*

STATEMENT										Identification- Sequence								
1	Name	8	10	14	16	20	25	30	35	40	45	50	55	60	65	71	73	80
READ	GET	CARDIN								READ	SUBROUTINE							
	BR	14																
* WRITE	PUT	ALINE								WRITE	SUBROUTINE							
	BR	14																
* ASSEMBLE AND PRINT THE HEADER LINES																		
* START	MVC	OUTPUT,OUTPUT-1								CLEAR	OUTPUT AREA							
	MVC	HEADER,HDR1								MOVE	FIRST HEADER TO OUTPUT AREA							
	BAL	14,WRITE								PRINT	FIRST HEADER LINE							
	MVC	HEADER,HDR2								MOVE	SECOND HEADER TO OUTPUT AREA							
	BAL	14,WRITE								PRINT	SECOND HEADER LINE							
	MVC	OUTPUT,OUTPUT-1								CLEAR	OUTPUT AREA							
* READ THE TRANSACTION CARDS																		
* NEXT	BAL	14,READ								READ	A CARD							
	PACK	PPRIN,PRIN								REFORMAT								
	PACK	PRATE,RATE																
	PACK	PPAY,PAY								INPUT								
	CVB	3,PPRIN																
	CVB	7,PRATE								DATA								

Figure 6. Portion of a Program Written in Assembler

POST-TEST 5

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. List the letters to show the order in which the following must be performed.
 - a. Source program is written and then keypunched.
 - b. Compiler is loaded into main storage and source program is read at card reader.
 - c. Object program is loaded into main storage, job input data is read and then processed, and job output data is produced.
 - d. Flowchart is drawn.
 - e. Object program is result of compilation.
2. The process of compilation is:
 - a. An end of job branching sequence.
 - b. The translating of a source program to produce an object program.
 - c. A commercial data processing option only.
 - d. A process used only in scientific applications.
 - e. I don't know.
3. Match the two lists:

a. Business	1. FORTRAN
b. One-for-one	2. RPG
c. Scientific and business	3. Assembler
d. Preparation of reports	4. PL/1
e. Scientific	5. COBOL
f. I don't know how to match the two lists	
4. The symbols that are used in writing programs:
 - a. Allow the programmer to center his attention on the solution to the problem, rather than requiring him to concentrate on such tasks as relocating instruction sequences in main storage
 - b. Can be processed, after being put in machine-readable form, by the compiler for the programming language being used
 - c. Are more meaningful to the programmer than are machine-language numeric symbols
 - d. All of the above
 - e. I don't know
5. A COBOL compiler can be used to produce an object program from a source program written in:
 - a. PL/1
 - b. FORTRAN
 - c. RPG
 - d. None of the above
 - e. I don't know
6. Test data should be used to test:
 - a. The flowchart for the solution to a problem
 - b. The object program for the solution to a problem
 - c. Both of the above
 - d. Neither of the above
 - e. I don't know

Answers are on the next page.

Answers to Post-Test 5

1. d
a
b
e
c
2. b
3. a. 5
b. 3
c. 4
d. 2
e. 1
4. d
5. d
6. c

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 5. JOB OVERVIEW: FROM PLANNING TO OUTPUT DATA; PROGRAMMING LANGUAGES

1. A documented plan (or guide) that the programmer produces before he writes a program is called a:
 - a. Flowchart
 - b. Medium

• • •

 - a. Flowchart
2. Before the program is written, the flowchart is tested by use of representative test data. The test data is processed (through the logical steps of the flowchart) by the:
 - a. CPU
 - b. Programmer

• • •

 - b. Programmer
3. After the flowchart has been tested and corrected, the program can be written by use of a programming language. The symbols of a particular programming language are similar to those normally used by people who solve problems in the same problem area. On the other hand, machine-language instructions are generally represented by series of numbers, which are not very meaningful to people.

The program that is written by the programmer is called a *source* program. A source program is written by the programmer in:

 - a. A programming language
 - b. A machine-language

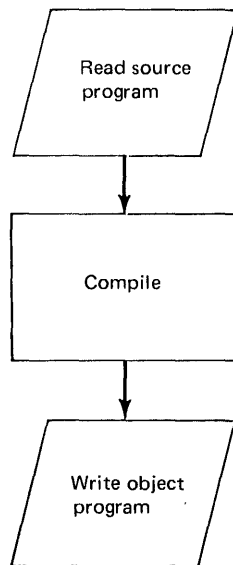
• • •

 - a. A programming language
4. The CPU can execute only instructions that are in machine-language form. The instructions in a source program:
 - a. Can be executed by the CPU
 - b. Cannot be executed by the CPU

• • •

 - b. Cannot be executed by the CPU
5. The job of converting the source program to a machine-language program is a tedious task when the converting is done by hand by the programmer. A series of numbers must be produced for each specific machine-language instruction required by a statement in the source program. Also, each machine-language instruction must be assigned to a main-storage location. Such time-consuming tasks can be performed very quickly by a data processing system.

The program that is executed by the CPU to produce a machine-language program from a source program is called a *compiler*. Output data produced as a result of compilation is called an *object* program. The process can be shown in flowchart form as follows:



Input data to be processed by a compiler is called:

- a. The machine-language program
 - b. The source program
 - • •
 - b. The source program
6. The source program is written by use of a programming language because:
- a. The symbols of a programming language are more meaningful to a programmer than are lists of numbers that represent machine-language instructions.
 - b. The job of converting a source program to a machine-language program can be performed very quickly by a compiler.
 - • •
 - Both
7. The symbols of a programming language mean more to a programmer than do the series of numbers that represent:
- a. Machine-language instructions
 - b. The source program
 - • •
 - a. Machine-language instructions
8. After the programmer writes a source program, that source program must be put into some machine-readable form (such as in punched cards) before:
- a. The source program can be compiled to produce an object program.
 - b. The flowchart can be drawn.
 - • •
 - a. The source program can be compiled to produce an object program.

9. The programmer writes source programs by use of a programming language because after the source program is put into a machine-readable form, such as punched cards, and loaded into the system as input data:
 - a. The source program can process input job data.
 - b. The source program can be translated into a machine-language object program by the compiler.

• • •

 - b. The source program can be translated into a machine-language object program by the compiler.

10. The job of translating a source program to an object program is performed by a compiler. Also, the compiler performs the tasks of assigning or relocating instruction sequences to specific main storage locations. Because all of these tasks are performed by the compiler, the programmer need not worry about them. Instead, he uses programming-language symbols that:
 - a. Allow the programmer to concentrate on the solution to the problem
 - b. Are long series of numbers

• • •

 - a. Allow the programmer to concentrate on the solution to the problem

11. The programmer writes programs by use of the symbols of a programming language because:
 - a. Such symbols are more meaningful than the long series of numbers that represent machine-language instructions.
 - b. Such symbols can be translated into machine-language instructions by a compiler.

• • •

Both

12. Symbols of a programming language:
 - a. Allow the programmer to concentrate on the solution to the problem rather than on such tasks as assigning or relocating instruction sequences to specific main storage locations
 - b. Are long series of numbers

• • •

 - a. Allow the programmer to concentrate on the solution to the problem rather than on such tasks as assigning or relocating instruction sequences to specific main storage locations

13. The output data produced by a compiler is called:
 - a. The source program
 - b. The object program

• • •

 - b. The object program

14. Match the two lists:
- | | |
|-------------------|--|
| a. Source program | 1. Input data for compilation process |
| b. Object program | 2. Output data produced by compilation |
| ● ● ● | |
| a. Source program | 1. Input data for compilation process |
| b. Object program | 2. Output data produced by compilation |
15. Before compilation can be performed:
- The compiler program must be loaded into main storage.
 - The source program must be in some machine-readable form, such as punched cards.
- ● ●
- Both
16. A program is usually not loaded into main storage until that program is to be executed. A program cannot be executed until input data, required by that program, is available at an input device. Key punching a source program takes a relatively long time. A compiler should be loaded into main storage:
- Before the source program is keypunched
 - When the source program, in punched card form, is ready to be loaded into a card reader
- ● ●
- When the source program, in punched card form, is ready to be loaded into a card reader
17. When the compiler is in main storage and the source program is in an input medium in an input device:
- Compilation can be performed.
 - An object program can be produced.
- ● ●
- Both
18. List the letters to show the sequence in which the following steps must be performed:
- Object program is system output.
 - Source program is keypunched.
 - Flowchart is drawn.
 - Source program is written by programmer.
 - Compiler is loaded into main storage.
 - Source program is read at card reader.
- ● ●
- Flowchart is drawn.
 - Source program is written by programmer.
 - Source program is keypunched.
 - Compiler is loaded into main storage.
 - Source program is read at card reader.
 - Object program is system output.

19. Specific programming languages have been developed for particular classes of applications. Two such languages are FORTRAN and COBOL. The word FORTRAN is derived from FORMula TRANslation; the word COBOL is derived from COMmon Business Oriented Language. COBOL is designed for use in business applications, FORTRAN for scientific. You would expect that a source program written for a problem related to predicting orbits of comets would be written in:

- a. COBOL
- b. FORTRAN

• • •

- b. FORTRAN

20. A rule is that a source program can be compiled only by a compiler for the programming language used to write the source program. PL/1 is a general purpose programming language designed to be used with problems of either a scientific or a business nature.

Therefore:

- a. A FORTRAN compiler can be used to compile PL/1 source programs.
- b. A PL/1 compiler can be used to compile a FORTRAN source program.

• • •

Neither

21. Match the two lists:

- | | |
|---------------------------|------------|
| a. Business or scientific | 1. PL/1 |
| b. Business | 2. FORTRAN |
| c. Scientific | 3. COBOL |

• • •

- | | |
|---------------------------|------------|
| a. Business or scientific | 1. PL/1 |
| b. Business | 3. COBOL |
| c. Scientific | 2. FORTRAN |

22. RPG (Report Program Generator) is a programming language that is designed primarily for the preparation of reports. Match the two lists:

- | | |
|--|------------|
| a. Statistical analysis of earthquake shock patterns | 1. RPG |
| b. Billing reports | 2. FORTRAN |

• • •

- | | |
|--|------------|
| a. Statistical analysis of earthquake shock patterns | 2. FORTRAN |
| b. Billing reports | 1. RPG |

23. Frequently, each instruction written in a programming language results in the production, during compilation, of more than one machine-language instruction. The Assembler language is an exception to this rule. The Assembler is called a one-for-one programming language because one machine-language instruction is produced from each source program instruction. (There are exceptions to this in the Assembler language; this relationship, however, indicates what the term "one-for-one" means in relation to programming languages.)

No response required. Go to the next frame.

• • •

24. Match the two lists:

- | | |
|--------------|---|
| a. PL/1 | 1. A one-for-one programming language |
| b. RPG | 2. Scientific problems |
| c. FORTRAN | 3. Scientific and business applications |
| d. Assembler | 4. Business problems |
| e. COBOL | 5. Preparation of reports |

• • •

- | | |
|--------------|---|
| a. PL/1 | 3. Scientific and business applications |
| b. RPG | 5. Preparation of reports |
| c. FORTRAN | 2. Scientific problems |
| d. Assembler | 1. A one-for-one programming language |
| e. COBOL | 4. Business problems |

25. Besides an object program, a compiler usually produces a listing (printed report). The listing contains, among other things, messages about any errors that the compiler detected in the source program.

If the compiler has detected any errors in the source program:

- The source program should be corrected.
- The compiler produces a correct object program because, if it can detect the errors in the source program, it can correct the errors.

• • •

- The source program should be corrected.

26. After a corrected source program has been compiled, the resulting object program is tested with input test data. Before the object program can process input test data:

- The object program must be loaded into main storage.
- Input test data must be available at an input device.

• • •

Both

27. After analyzing the results (output data) of a test run of an object program, the programmer revises the source program. (Almost all object programs do not run correctly the first time. Usually, several test runs are required.)

Next, compilation is performed by using:

- The revised source program as input to the compiler
- The object program as input to the compiler

• • •

- The revised source program as input to the compiler

28. Choose, from the following list, two items that are normally tested with test data:

- The problem statement
- The flowchart for the solution to the problem
- The compiler
- The object program for the solution to the problem

• • •

- The flowchart for the solution to the problem
- The object program for the solution to the problem

29. After an object program that passes all tests has been produced, the job, for which the object program was produced, can be run by:
- a. Loading the object program into main storage
 - b. Making input job data available at an input device
- • •
- Both
30. The goal aimed at during the entire process of planning for the solution of a problem can now be achieved. That goal is:
- a. Job output data
 - b. A properly drawn flowchart
- • •
- a. Job output data
31. List the letters to show the sequence in which the following steps should be performed:
- a. Object program is loaded into main storage, job input data is read and then processed, and job output data is produced.
 - b. Flowchart is drawn.
 - c. Object program is result of compilation.
 - d. Source program is written and then keypunched.
 - e. Compiler is loaded into main storage and source program is read at card reader.
- • •
- b. Flowchart is drawn.
 - d. Source program is written and then keypunched.
 - e. Compiler is loaded into main storage and source program is read at card reader.
 - c. Object program is result of compilation.
 - a. Object program is loaded into main storage, job input data is read and then processed, and job output data is produced.

For directions, refer to the summary on the inside of the front cover.

PRETEST 6

Please write your answers on a separate sheet of paper. Please do not guess. Specify the “I don’t know” answer when appropriate.

Questions

1. Source program instructions can be written according to the plan specified in a document called:
 - a. An object program
 - b. A decision table
 - c. An input source data medium
 - d. An output source data medium
 - e. I don’t know
2. Maintaining program-logic documents (such as a flowchart) after an object program has been produced makes it easier:
 - a. To produce new but similar programs
 - b. To change or modify the program
 - c. For new programmers, to be able to understand the logic of the program
 - d. All of the above
 - e. None of the above
 - f. I don’t know
3. List the letters to show which of the following documents are normally provided to the operator of a data processing system:
 - a. Program flowcharts of problem statements
 - b. Program set up procedures
 - c. Decision tables of problem statements
 - d. Problem statements
 - e. Program-error messages and manual actions required for each
 - f. Media handling procedures
 - g. Machine operation procedures
 - h. I don’t know
4. List the letters to show the most probable order in which the following items are documented:
 - a. Operator instructions for a specific program
 - b. Flowchart
 - c. Input and output record layouts
 - d. Problem statement
 - e. Program listing
 - f. I don’t know

Answers are on the next page.

Answers to Pretest 6

1. b
2. d
3. b
e
f
g
4. d
c
b
e
a

DESCRIPTIVE MATERIAL FOR CHAPTER 6. DOCUMENTATION

Problem Statement, Input/Output Record Layouts, Flowcharts

Various printed documents must be made available to the people who work at data processing jobs.

Problem statements (also called problem descriptions) must be documented to provide an adequate description of the problem to the programmer. In this course, you have drawn flowcharts by using problem statements as sources of information. Usually, however, the programmer specifies input-record and output-record layouts before he flowcharts the solution to a problem. The positions of the fields in a record are the layout of that record. For example, in a punched card input record the layout might be:

<i>Field</i>	<i>Card Columns</i>
Name	1 through 24
Employee number	25 through 31
Physical characteristics	32 through 80

A different arrangement of fields would be a different record layout.

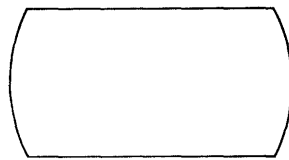
He specifies record layouts first so that he knows the characteristics (types, sizes, and relative positions) of input data before he tries to specify the operations through which that data must be processed. He does the same thing for output data so that he knows precisely the characteristics of data to be produced by processing. Input and output record layouts, then, are roadguides to flowcharting. If you know where you are going (output data) and what you have to start with (input data), you can plan better how to get to where you are going. Three necessary documents, then, and the order in which they are produced, are:

1. The *problem statement* (or problem description), which is the initial documented input to the programmer.
2. The *input and output record layouts*, which are specified by the programmer so that he knows the characteristics of both the data that is to be produced (output) and the data that is available (input) to do the job.
3. The *flowcharts*, which specify the plan for the solution of the problem.

In this course, you have drawn program-flowchart solutions to problem statements. As you know, program flowcharts show in detail the operations that must be performed to solve a particular problem.

Another kind of flowchart, called the *system* flowchart, is used to plan the overall data flow through the machines that are available at the data processing installation. System flowcharts are usually drawn before program flowcharts. Some symbols used in system flowcharts are shown in Figure 7; an example of a system flowchart is shown in Figure 8.

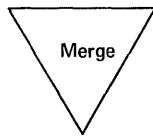
Note: You are not asked to draw system flowcharts for this course. The word flowchart, as used in the remainder of this course, refers to program flowcharts, not to system flowcharts.



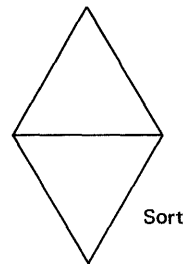
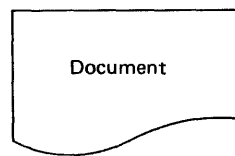
An operation using a key-driven device - -
such as punching, verifying, typing



Input/output function in card medium



Combining two or more sets of items into one set



Arranging a set of items into sequence

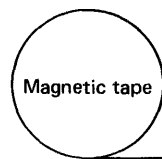
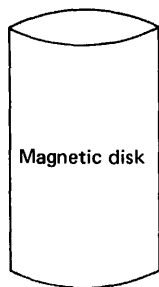
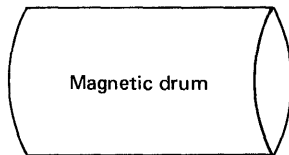


Figure 7. Examples of Some System Flowchart Symbols

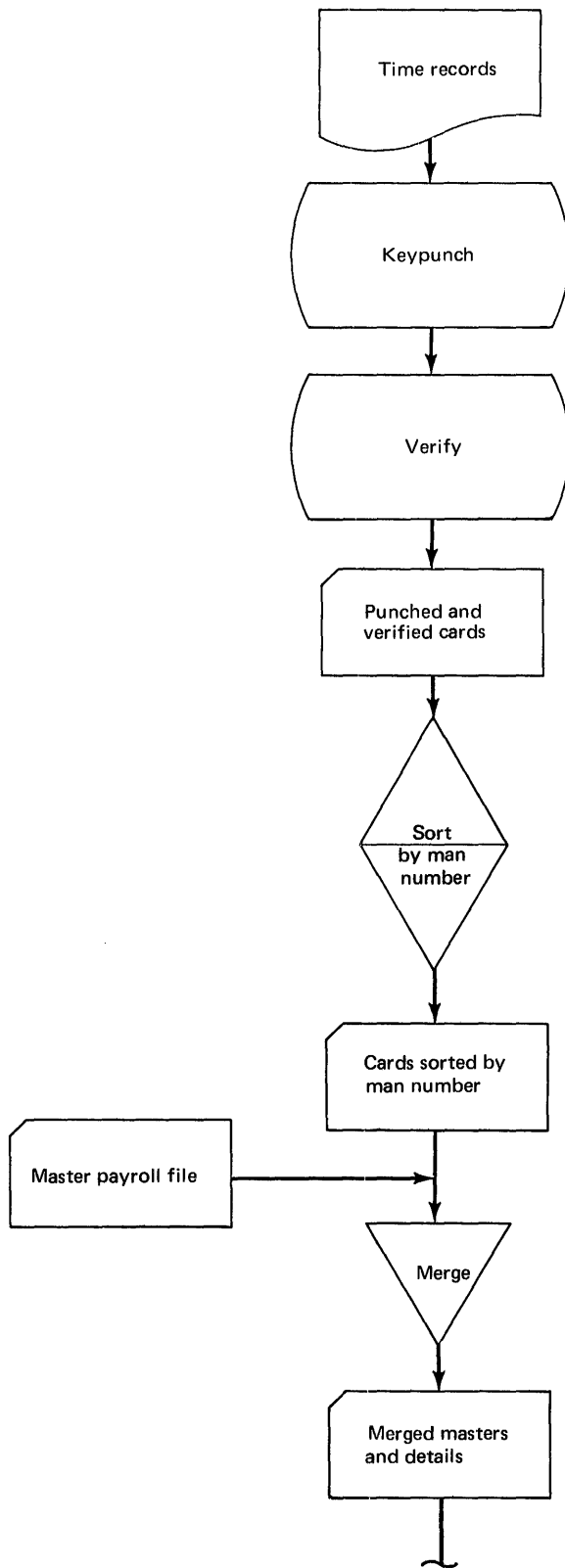


Figure 8. Example of Part of a System Flowchart

Decision Tables

The name of another document that is used in the planning of solutions to problem statements is the *decision table*. When actions to be taken are dependent upon conditions, a decision table may be more applicable (than a flowchart) to document the solution to a problem. An example of a decision table is:

Decision table

Conditions	Part number from plant 2?	Y	N	Y	N
	Last card?	N	N	Y	Y
Actions	Print record	X		X	
	Calculate and punch a card		X		X
	Stop			X	X
	Read another card	X	X		

Notice, in the preceding decision table, that:

- The answers to conditions are specified by Y (yes) and N (no).
- The action to be taken is indicated by X.

This decision table indicates that:

- When a part number is from plant two and it is not the last card (conditions), print a record and read another card (actions)
- When a part number is *not* from plant two and it is not the last card (conditions), calculate and punch a card and read another card (actions)

Note that in either case, if the last card has been read, the appropriate output operation is performed and the job is stopped.

Decision tables may be very useful when a large number of actions are dependent upon a large number of conditions.

You are not asked to draw decision tables for this course. You should know, however, that:

1. Decision tables provide a way of documenting the plan for the solution to a problem.
2. Source-program instructions can be written according to the plan specified in decision tables.

Two tools, then, that the programmer can use to document his solution to a data processing problem are:

1. Flowcharts
2. Decision tables

Maintaining Program-Logic Documentation

Either flowcharts or decision tables or both may be used to document the plan for the solution of a problem. Such permanent records are very useful when a change or modification must be made to an existing program. By the time such a change is to be made, the programmer very likely will have forgotten the exact logic paths he designed into the original program. By referring to the flowchart (or decision table) he can quickly recall the logic that exists and change it. The change may be because of such things as new job requirements or error conditions not previously thought of.

For one reason or another, the programmer who originally produced a program may not be the person who has to modify that program. Here, the need for the flowchart (or decision table) becomes critical. Another programmer must make the change. By using the program-logic documentation he can relatively quickly find the logical paths that must be modified. Without such documentation, this new programmer would waste considerable time finding the areas of the program to be changed. Even then, he would not be certain that he had found all related logic paths.

Another reason for maintaining flowcharts and decision tables is that new problem statements may be similar to problems that have already been programmed. Entire sections of old programs may be directly applicable to the solving of new problems. Program-logic documents (flowcharts and decision tables) can be inspected to verify that such is the case.

Essentially then, maintaining program-logic documents saves man-hours by facilitating the:

1. Changing or modifying of existing programs
2. Understanding, by new programmers, of existing programs
3. Producing of new but similar programs

Flowcharting is facilitated by the use of a program available from IBM. This program provides for the printing of data processing system *generated* flowcharts. This program does not plan a solution to a data processing job. It does, however, provide a way of producing clear, standardized, printed flowcharts.

Program Listings

After the record layouts and flowchart(s) for a problem statement have been produced, the source program can be written. Compilation of the source program then produces:

1. An object program
2. A program listing

The program listing is a printed record of:

1. The source statements written by the programmer
2. The corresponding machine-language instructions produced by compilation
3. Error messages concerning any errors detected in the source program by the compiler

Right after a fully tested object program is available, the procedures used by the person who operates the machines in a data processing system must be documented. The procedures referred to here are those directly applicable to the object program.

Documents for Operator

First, the machine operator must know what to do to operate the machines. Reference manuals are available from the manufacturers of data processing equipment. These manuals specify such things as how to start and stop a machine and the procedures to be followed after each type of machine error condition. Media handling procedures, when required, are also frequently contained in reference manuals. You may have heard, “Do not bend, fold, or mutilate.” That statement pertains to the punched-card medium. Punched cards also have certain relative humidity requirements. As an extreme example, a card soaked in water would have little chance of being read successfully by a card reader. Other media have other, sometimes more stringent handling requirements. Such information must be made available to the operator who handles the media.

The operator must have documents that contain program set-up procedures. These procedures would state such items as which input and output devices must be made ready (turned on, loaded with cards, and so on) before the job is run.

Frequently, while a job is being run, the system detects conditions that require operator action. Many programs provide messages to the operator for such situations. For example, a printer may run out of paper forms. The program may then (by means of an output device called a console printer—similar to a typewriter) print a message to the operator such as:

PRINTER FORMS

The operator would have to know that the message **PRINTER FORMS** means that the manual operation of putting paper in the printer should be performed. More complex operator procedures are frequently encountered.

In summary, then, the operator must have documents for:

1. Machine operation procedures
2. Media handling procedures
3. Program set-up procedures (such as which I/O devices should be made ready for each program run)
4. Program messages to the operator and descriptions of actions required for each message

POST-TEST 6

Please write your answers on a separate sheet of paper. Please do not guess. Specify the “I don’t know” answer when appropriate.

Questions

1. A documented plan that can be used as a guide to write source program instructions is called:
 - a. A decision table
 - b. An input source data medium
 - c. An object program
 - d. An output source data medium
 - e. I don’t know
2. After an object program has been produced, flowcharts and decision tables are maintained in order to make it easier:
 - a. To change or modify the program
 - b. For the new programmer, to be able to understand the logic of the program
 - c. To produce new but similar programs
 - d. All of the above
 - e. I don’t know
3. Indicate which of the following documents are normally provided to the operator of a data processing system:
 - a. Program set-up procedures
 - b. Decision tables of problem statements
 - c. Problem statements
 - d. Program error messages and manual actions required for each
 - e. Machine operation procedures
 - f. Program flowcharts of problem statements
 - g. Media handling procedures
 - h. I don’t know
4. Indicate the most probable order in which the following are documented:
 - a. Program listing
 - b. Flowchart
 - c. Operator instructions for a specific program
 - d. Input and output record layouts
 - e. Problem statement
 - f. I don’t know

Answers are on the next page.

Answers to Post-Test 6

1. a
2. d
3. a
d
e
g
4. e
d
b
a
c

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 6. DOCUMENTATION

1. After a fully tested object program is available, all program-to-operator messages must be documented. Descriptions of the actions to be taken for each such message must also be made available to the operator.

The machine operator can take the appropriate action required for a program-generated message because:

- a. He has a document that describes the actions to be taken for each program-to-operator message.
- b. He is required to memorize all program-generated messages and related manual procedures.

• • •

- a. He has a document that describes the actions to be taken for each program-to-operator message.

2. Certain machine operation procedures require several actions that must be performed in specified ways. Some of these procedures are not frequently performed and, therefore, they are easily forgotten. Manufacturers provide reference manuals that describe such operator procedures.

In order for an operator to make ready the proper input and output devices for a specific object program and to perform related machine-operation procedures correctly, he should have:

- a. Program set-up procedures that specify which input and output devices are required for the particular job
- b. Reference manuals that describe how to operate the input and output devices

• • •

Both

3. The operator's task of setting up the proper input and output devices is described in a document called (choose one or more):

- a. Decision tables
- b. Program set-up procedures
- c. Program listings

• • •

- b. Program set-up procedures

4. Manufacturers also provide reference manuals that describe media-handling procedures. Such a procedure, for example, would describe how a particular medium should be loaded into an input (or output) device.

So that he can properly set up input data files at input devices and make output devices ready to receive output data files, the machine operator should have:

- a. Reference manuals that explain how input and output media are to be handled
- b. Reference manuals that describe flowchart-symbol usage

• • •

- a. Reference manuals that explain how input and output media are to be handled

5. The machine operator, in general, does not need documents that are used by the programmer in the development of a program. Usually, for example, the machine operator has no need of the original problem statement for which the programmer developed a plan of solution.

The machine operator has need, primarily, of documents that help him to operate machines correctly for specific programs.

A flowchart, which is the documented plan for the solution of a particular problem, is:

- a. Usually not needed by the machine operator
- b. Usually needed by the machine operator

• • •

- a. Usually not needed by the machine operator

6. Because the operator must sometimes take action as a result of unusual conditions detected by the program during a job run, he should have:

- a. A flowchart of the program being run
- b. A document that contains messages to the operator and actions required for each message

• • •

- b. A document that contains messages to the operator and actions required for each message

7. Two documents that the machine operator normally does *not* need are:

- a. Flowcharts and problem statements
- b. Reference manuals that contain media-handling procedures and program set-up procedures

• • •

- a. Flowcharts and problem statements

8. Indicate which of the following should be made available to the machine operator:

- a. Flowcharts
- b. Reference manuals that contain media-handling procedures
- c. Program set-up procedures
- d. Problem statements
- e. Reference manuals that contain machine-operation procedures
- f. Program messages to the operator and actions required for each message

• • •

- b. Reference manuals that contain media-handling procedures
- c. Program set-up procedures
- e. Reference manuals that contain machine-operation procedures
- f. Program messages to the operator and actions required for each message

9. Frequently, program-generated messages (and related operator actions) for a fully tested object program are different than those for other object programs. Also, before a program has been fully tested, not all program-to-operator messages have been determined.

Program messages to the operator (for a specified job) and directions for the action that should be taken for each message can be determined and documented:

- a. Only after a fully tested object program (for the job) is available
- b. When the flowchart for the job has been produced

• • •

- a. Only after a fully tested object program (for the job) is available

10. An object program and a program listing are the output data produced by compiling a source program. The program listing is a printed report of (refer to items **1** , **2** , and **3** , in Figure 9):

- a. _____
 b. _____
 c. _____

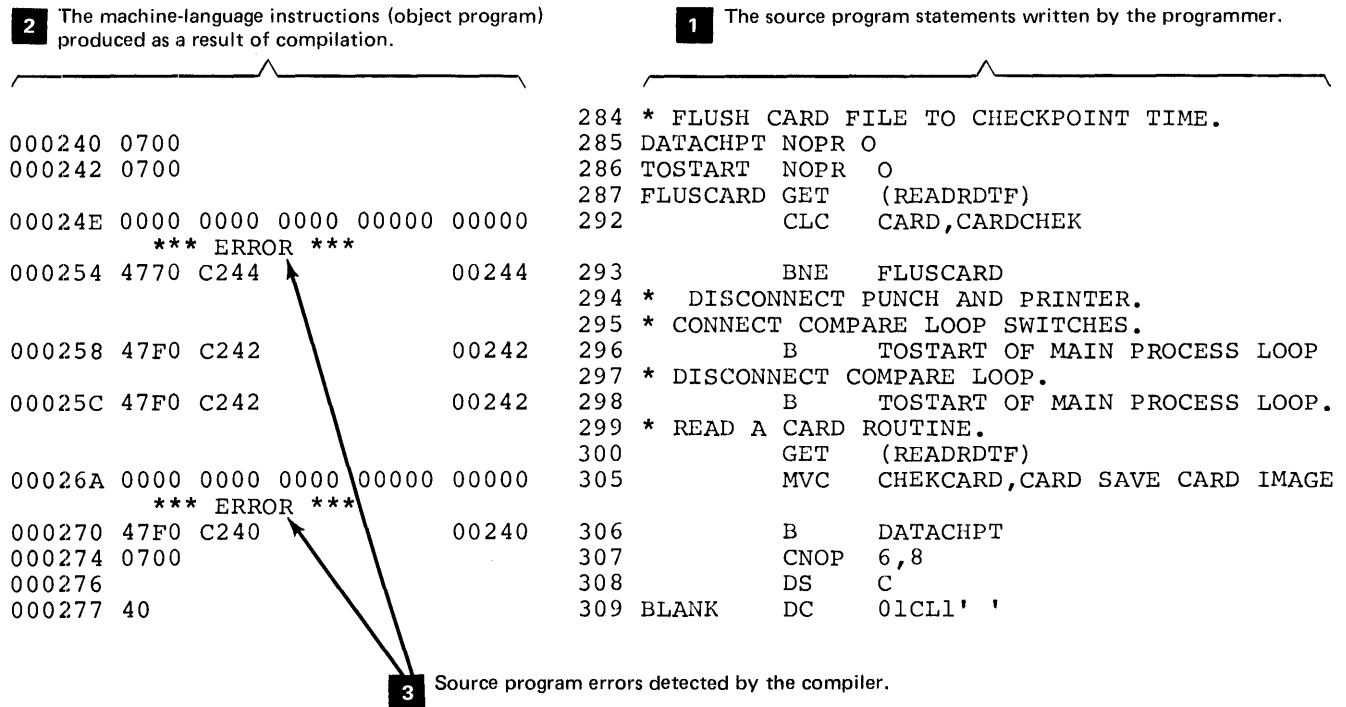


Figure 9. Program Listing from an Assembler Language Source Program

• • •

In any order:

- a. The source program statements written by the programmer
 b. The machine-language instructions (object program) produced as a result of compilation
 c. Source program errors detected by the compiler

11. The final program listing is a result of compiling an error-free, as far as the compiler is concerned, source program. The final program listing contains:

- a. Source program errors detected by the compiler
 b. A printed listing of the object program in machine-language form

• • •

- b. A printed listing of the object program in machine-language form

12. The original source program for a job is written by the programmer according to a plan specified in:

- a. A program listing
 b. A flowchart

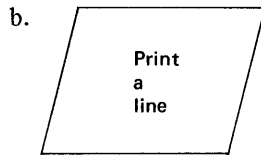
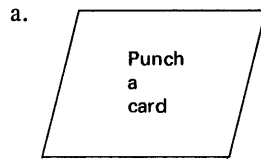
• • •

- b. A flowchart

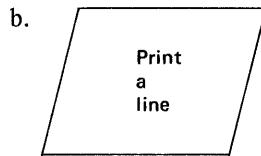
13. *Decision table* is the name of a document that can contain all or part of the plan for the solution of a problem statement. The decision table is useful when the states of a number of conditions determine which one, or more, of several actions is to be taken. The programmer can document his plan for the solution of a problem:
- In a flowchart
 - In a decision table
- • •
- Both
14. The programmer can document his plan of solution to a problem (that is described in a problem statement) in a flowchart. Another similar type of document that the programmer can produce to show his plan for the solution of a problem is called a:
- Decision table
 - Problem statement
- • •
- a. Decision table
15. The programmer, before he begins writing the source program, can show his plan for the solution to a problem in one or both of two types of documents that are called:
- _____
 - _____
- • •
- Flowcharts
 - Decision tables
- (In either order)
16. Most programs contain a large number of logical steps. Programmers change programs they have written by:
- Referring to the program listing for the program that is to be changed
 - Referring to the flowchart or decision table that was produced for the job program that is to be changed
- • •
- Both
17. On the other hand, operators should have available for their information needs which of the following documents?
- Reference manuals that contain media-handling procedures
 - Problem statements
 - Program messages to the operator and actions required for each message
 - Flowcharts
 - Reference manuals that contain machine-operation procedures
 - Program set-up procedures
 - Decision tables
- • •
- Reference manuals that contain media-handling procedures
 - Program messages to the operator and actions required for each message
 - Reference manuals that contain machine-operation procedures
 - Program set-up procedures

18. Changes or modifications to an existing program can also be made by new programmers who have never worked on the program before. New programmers can more easily make such changes by:
- a. Referring to program-logic documents, such as flowcharts or decision tables, that are related to the job program to be modified
 - b. Referring to program listings for the program that is to be changed
- • •
- Both
19. New jobs are sometimes similar to old jobs. Parts of old programs:
- a. May be applicable to new programs
 - b. Can be studied by referring to the flowcharts (or decision tables) from which the old program was written
- • •
- Both
20. Maintaining up-to-date program-logic documentation makes easier the task of:
- a. Modifying a program (by either the programmer who wrote the program or by a new programmer)
 - b. Compiling
- • •
- a. Modifying a program (by either the programmer who wrote the program or by a new programmer)
21. Two program-logic documents that the programmer can use to show his plan for the solution to a problem statement are called:
- a. Flowchart
 - b. Compiler
 - c. Decision table
 - d. Object deck
 - e. Media-handling procedures
- • •
- a. Flowchart
 - c. Decision table
22. Before flowcharts are drawn, the fields in input and output records must be organized. The programmer plans the layout of such records so that:
- a. He knows the sizes and kinds of input data that must be processed.
 - b. He knows the characteristics of the output data to be produced.
- • •
- Both
23. Recall that the flowchart is a document that the programmer produces. The flowchart is a document that represents:
- a. The programmer's plan for the solution of the problem
 - b. The characteristics of input data
- • •
- a. The programmer's plan for the solution of the problem

24. Before he can draw a flowchart, the programmer must know what output data is to be produced. If, for example, output data is only to be printed, you would expect to find which of the following in the flowchart?



• • •



25. Frequently, output data must be organized in very specific ways. For example, fields that are to be in an output printed line might be, in left-to-right order:
- | | | | |
|---------------|---------------|--------|------|
| Employee name | Serial number | Amount | Time |
|---------------|---------------|--------|------|
- Before the programmer plans his solution to a problem statement, then, he should know what arrangement of the output data fields is required.

Before the programmer draws a flowchart:

- a. He plans the layouts of output data records.
b. He writes the source program.

• • •

- a. He plans the layouts of output data records.

26. Before the programmer can plan how to process input data, he must know what input data is available, and how it is arranged.

Before the programmer draws a flowchart:

- a. He must have a problem statement.
b. He plans the layouts of input data records.

• • •

Both

27. Before a flowchart is drawn:

- a. Input record layouts are usually planned.
b. Output record layouts are usually planned.

• • •

Both

28. Input and output record layouts are specified after the programmer has a documented:
- a. Problem statement
 - b. Object program
- • •
- a. Problem statement
29. The first document a programmer must have, before he can plan a solution to a problem, is:
- a. A problem statement which is a description of the problem to be solved
 - b. A source program listing
- • •
- a. A problem statement which is a description of the problem to be solved
30. After he has a problem statement, the programmer can begin planning a solution to the problem. Before he produces a flowchart, however, the programmer usually:
- a. Plans the layouts of input records
 - b. Plans the layouts of output records
- • •
- Both
31. Show the correct order in which the following documents are usually produced:
- a. Input and output record layouts
 - b. Problem statement
 - c. Flowchart
- • •
- b. Problem statement
 - a. Input and output record layouts
 - c. Flowchart
32. After the programmer has flowcharted his initial plan for the solution of a problem, he tests the flowchart with test data. After correcting (if necessary) the flowchart according to the test results, the programmer can begin writing the source program. A document that contains a printed record of the source program and any errors detected by the compiler in that source program is called:
- a. The program listing
 - b. The flowchart
- • •
- a. The program listing

33. After a successfully tested object program has been produced, information concerning any manual actions to be taken while the program is being run can be documented.

Rearrange the following according to which document is produced first, which second, and which third:

- a. Operator instructions
- b. Flowchart
- c. Program listing

• • •

- b. Flowchart
- c. Program listing
- a. Operator instructions

34. List the letters to show the order in which the following documents are usually produced:

- a. Flowchart
- b. Operator instructions for a specific program
- c. Program listing
- d. Input and output record layouts
- e. Problem statement

• • •

- e. Problem statement
- d. Input and output record layouts
- a. Flowchart
- c. Program listing
- b. Operator instructions for a specific program

For directions, refer to the summary on the inside of the front cover.

PRETEST 7

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

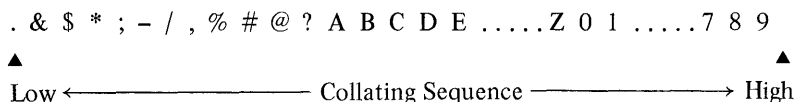
1. Match the two lists:

<ol style="list-style-type: none"> a. p, x, a, q, w, c b. 75, 74, 73, 72, 71, 70, 69, 68 c. a, c, d, f, g, x, z d. 7, 8, 9, 10, 11 e. 9000, 8500, 4500, 1500 f. I don't know how to match the two lists. 	<ol style="list-style-type: none"> 1. Descending sequence, consecutive 2. Ascending sequence, consecutive 3. Descending sequence, nonconsecutive 4. Random 5. Ascending sequence, nonconsecutive
--	---

2. Sort the following records into *ascending* sequence:

	<i>Plant</i>	<i>Amount (Key)</i>	<i>Man Number</i>	<i>Quantity</i>	← Fields
a.	01	0017	1111	59	
b.	20	0045	1114	58	
c.	14	0171	1101	56	
d.	06	0093	1194	28	
e.	02	0147	1061	94	


- f. I don't know how to sort the records.
3. Arrange the following (by use of the following chart) into *descending* collating sequence:
 - a. \$
 - b. B
 - c. D
 - d. #
 - e. 7
 - f. 0 (digit)
 - g. Z
 - h. .
 - i. 1 (digit)
 - j. A
 - k. I don't know how to arrange the items.



Answers are on the next page.


Answers to Pretest 7

1.
 - a. 4
 - b. 1
 - c. 5
 - d. 2
 - e. 3

2. Sorting is performed by key:
 - a. 0017 Ascending
 - b. 0045
 - d. 0093
 - e. 0147
 - c. 0171

3.
 - e. 7
 - i. 1 (the digit one)
 - f. 0 (the digit zero)
 - g. Z
 - c. D
 - b. B
 - j. A
 - d. #
 - a. \$
 - h. .

Descending



DESCRIPTIVE MATERIAL FOR CHAPTER 7. ARRANGING RECORDS

Key

Records must be identifiable by the program. A specific field is examined to identify records. This field is called the *key*. (It is sometimes called the control field.) While the term key indicates which field is to be examined for record identification, the name of that field frequently does not contain the word key. For example, the names of three fields in a payroll record might be:

- Social security number
- Employee number
- Plant number

Any one of these can be called the key; the choice is dependent upon the requirements of the job to be performed.

Ascending, Descending, Consecutive, Nonconsecutive, Random

Many data processing applications require that records be arranged in a particular order. The records are arranged by the data items read from the specified key (or control field). One arrangement is called *ascending sequence*. An example, using numeric data, is:

100, 109, 115, 117, 118, 119, 130
—————→

Notice that “gaps” are in the list. The numbers 101 through 108, as an example, are missing. (Fractional numbers are not used in sequences in this book.) The sequence *ascends* consistently, however, from left-to-right.

When such “gaps” do not exist, the series is said to be *consecutive*. A group of consecutive numbers, arranged in ascending sequence, is:

3, 4, 5, 6, 7, 8, 9, 10, 11
—————→

Alphabetic sequences may also be constructed. The following is a consecutive, ascending alphabetic sequence:

l, m, n, o, p, q
—————→

A nonconsecutive, ascending alphabetic sequence is:

c, f, h, k, l, m, p, t, w
—————→

The opposite arrangement (to an ascending sequence) is a *descending* sequence.

Examples are:

Nonconsecutive, descending numeric sequence:

1000, 985, 432, 211, 43, 10
—————→

Consecutive, descending numeric sequence (fractions excluded):

15, 14, 13, 12, 11, 10, 9, 8
—————→

Nonconsecutive, descending alphabetic sequence:

k, i, f, e, d, c, a
—————→

Consecutive, descending alphabetic sequence:

m, l, k, j, i, h, g, f
—————→

Groups of items that are not arranged are said to be in *random order*. For example, the following two groups are said to be in random order:

- a, z, q, r, t, d, g
- 75, 14, 93, 1041, 82

Sort

Sort means the same thing as arrange. Before sorting a group of records:

1. The sequence (ascending or descending) into which the records are to be sorted must be known.
2. The key (control field) must be known.

For example, assume that the following group of records is to be sorted into ascending sequence:

<i>Man Number</i>	<i>Parts per Hour</i>	<i>Plant (Key)</i>	<i>Total</i>
0014	44	09	1
0119	26	07	4
0011	99	10	9
0187	80	03	3
0103	14	15	2

Sorting the preceding records by *key* results in:

<i>Man Number</i>	<i>Parts per Hour</i>	<i>Plant (Key)</i>	<i>Total</i>
0187	80	03	3
0119	26	07	4
0014	44	09	1
0011	99	10	9
0103	14	15	2

Changing the key from the plant to the total field, in the preceding records, and then sorting into *ascending* sequence produces:

<i>Man Number</i>	<i>Parts per Hour</i>	<i>Plant</i>	<i>Total (Key)</i>
0014	44	09	1
0103	14	15	2
0187	80	03	3
0119	26	07	4
0011	99	10	9

Collate

Putting two similarly arranged groups of items into one group is called *collating*. For example, consider two groups of records, each of which is in ascending sequence:

<i>Group 1</i>	<i>Group 2</i>
<i>Key</i>	<i>Key</i>
1	2
4	3
6	9

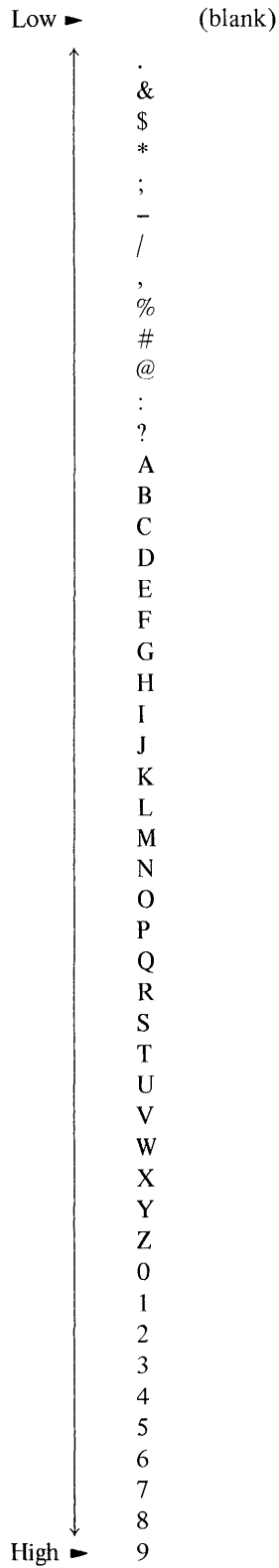
Collating the two groups results in a single group arranged in ascending sequence:

Key
1
2
3
4
6
9

If we had considered both original groups as a single group and sorted that group into ascending sequence, the results would be the same as the results of collating the two groups. The term *collate*, however, means the “putting together” of two already arranged groups of items.

Collating Sequence

The relationships among alphameric characters, with respect to sequencing, is shown by the following collating-sequence chart:



Other collating charts exist which define different relationships between numeric characters and alphabetic characters.

Keys that contain alphabetic, special, and numeric characters, then, can be arranged by use of the preceding chart. As an example, consider the characters:

: ? C \$ * 5 6 0 / F

Arranging these into *ascending*, collating sequence, by use of the preceding chart, results in:

\$ * / : ? C F 0 5 6

Low _____→High

Arranging the same characters into *descending*, collating sequence results in:

6 5 0 F C ? : / * \$

High _____→Low

POST-TEST 7

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. Match the two lists:
- | | |
|---|--|
| a. t, s, r, q, p | 1. Ascending sequence, consecutive |
| b. a, b, x, c, p | 2. Ascending sequence, nonconsecutive |
| c. 14, 15, 16, 17, 18, 19 | 3. Descending sequence, consecutive |
| d. 11, 9, 6, 5, 3 | 4. Descending sequence, nonconsecutive |
| e. 78, 79, 83, 84, 86 | 5. Random |
| f. I don't know how to match the two lists. | |

2. Sort the following records into *descending* sequence:

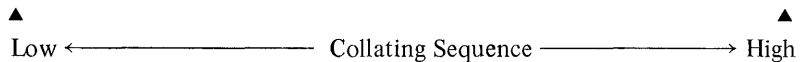
	<i>Country</i>	<i>State</i>	<i>City</i>	<i>County (Key)</i>	← Fields
a.	1	3	91	91	
b.	3	4	18	41	
c.	1	7	75	21	
d.	4	35	16	40	
e.	2	11	20	37	

- f. I don't know how to sort the records.
3. Arrange the following items (by use of the following chart) into *ascending* collating sequence:

- a. -
- b. E
- c. ,
- d. 8
- e. D
- f. /
- g. 9
- h. %
- i. &
- j. A

- k. I don't know how to arrange the items.

. & \$ * ; - / , % # @ ? A B C D E Z 0 1 7 8 9




Answers are on the next page.

Answers to Post-Test 7


1.
 - a. 3
 - b. 5
 - c. 1
 - d. 4
 - e. 2

2. Sorted by key into descending sequence:
 - a. 91
 - b. 41
 - d. 40
 - e. 37
 - c. 21

Descending


- 3.
 - i. &
 - a. -
 - f. /
 - c. ,
 - h. %
 - j. A
 - e. D
 - b. E
 - d. 8
 - g. 9

Ascending



PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 7. ARRANGING RECORDS

1. Arranging the letters *d, f, a* into their normal alphabetic order results in an ascending nonconsecutive alphabetic sequence:

a, d, f
—————→

Inserting the missing letters (b, c, and e) into the preceding sequence results in a consecutive, ascending alphabetic sequence:

a, b, c, d, e, f
—————→

Match the two lists:

- | | |
|---------------|---------------------------------------|
| a. h, j, m | 1. Ascending sequence, consecutive |
| b. h, i, j, k | 2. Ascending sequence, nonconsecutive |

• • •

- | | |
|---------------|---------------------------------------|
| a. h, j, m | 2. Ascending sequence, nonconsecutive |
| b. h, i, j, k | 1. Ascending sequence, consecutive |

2. Arrange the following into an ascending sequence:

x, q, a, d, m, k

• • •

a, d, k, m, q, x
—————→

3. The answer to the preceding frame is a sequence that is:

- a. Consecutive
b. Nonconsecutive

• • •

- b. Nonconsecutive

4. A numeric ascending sequence proceeds from low numbers to high numbers:

7, 11, 15, 19
—————→

Arrange the following into an ascending sequence:

51, 4000, 20, 91, 24

• • •

20, 24, 51, 91, 4000
—————→

5. An ascending numeric sequence is consecutive when no number is missing. (We do not, however, consider fractions.)

A nonconsecutive, ascending sequence:

7, 15, 21, 27
—————→

A consecutive, ascending sequence:

7, 8, 9, 10, 11
—————→

Match the two lists:

- | | |
|--------------------|---------------------------------------|
| a. 7, 8, 9, 10, 11 | 1. Ascending sequence, consecutive |
| b. 81, 90, 105 | 2. Ascending sequence, nonconsecutive |

• • •

• • •

- a. 7, 8, 9, 10, 11 1. Ascending sequence, consecutive
- b. 81, 90, 105 2. Ascending sequence, nonconsecutive

6. Items in descending sequences are in the opposite order as items in ascending sequences (with respect to the starting point of each sequence).

An ascending alphabetic sequence:

Starting point \longrightarrow b, d, g

The same alphabetic characters arranged in a descending sequence:

Starting point \longrightarrow g, d, b

Arrange the following in descending sequence:

k, o, g, l

• • •

o, l, k, g

\longrightarrow

7. A descending numeric sequence:

57, 20, 5, 4, 2

\longrightarrow

Arrange the following in descending sequence:

51, 28, 500, 2, 84

• • •

500, 84, 51, 28, 2

\longrightarrow

8. Descending sequences are consecutive when no element of the sequence is missing. Match the two lists:

- a. a, k, l, m 1. Ascending sequence, consecutive
- b. 900, 899, 898, 897 2. Ascending sequence, nonconsecutive
- c. s, t, u, v, w 3. Descending sequence, consecutive
- d. 58, 21, 13, 8 4. Descending sequence, nonconsecutive

• • •

- a. a, k, l, m 2. Ascending sequence, nonconsecutive
- b. 900, 899, 898, 897 3. Descending sequence, consecutive
- c. s, t, u, v, w 1. Ascending sequence, consecutive
- d. 58, 21, 13, 8 4. Descending sequence, nonconsecutive

9. When a group of items are not in either ascending or descending sequence, that group is said to be in random order.

Match the two lists:

- a. x, k, a, m, b 1. Ascending
- b. 50, 49, 48, 47, 46 2. Random
- c. a, k, m, x, z 3. Descending

• • •

- a. x, k, a, m, b 2. Random
- b. 50, 49, 48, 47, 46 3. Descending
- c. a, k, m, x, z 1. Ascending

10. The process of arranging items in some sequence is called *sorting*. Records are sorted according to a field called the *key* (or control field).

Sort the following records into ascending sequence:

	<i>Town</i>	<i>Hour (Key)</i>	<i>Temperature</i>
a.	aa	2300	10
b.	bx	1400	20
c.	cd	0800	15
d.	ef	1500	21

• • •

Sorted by *key*:

	<i>Town</i>	<i>Hour (Key)</i>	<i>Temperature</i>	
c.	cd	0800	15	Ascending ↓
b.	bx	1400	20	
d.	ef	1500	21	
a.	aa	2300	10	

11. The field that is frequently used by the program to identify records is the same field that is used to sort the records. This field is called the _____.

• • •

Key (or control field)

12. Arranging two already sequenced groups of items into one group is called *collating*. The sequence of the original items (within each group) should be the same as the sequence of the collated group.

Collate the following:

<i>Group 1</i>	<i>Group 2</i>
4	2
6	3
8	5
10	7

• • •

Collated group (in ascending sequence, the same as both original groups):

2, 3, 4, 5, 6, 7, 8, 10

13. Many times, data items are made up of numeric, alphabetic, and special characters. For example, a part number might be #431A21. Frequently, such items must be sorted or collated. A collating chart is used as a reference so that these characters can be sorted or collated:

. & \$ * ; - / , % # @ ? A B C D E Z 0 1 7 8 9

▲
Low ←————— Collating Sequence —————→ High ▲

(Note that the between E and Z and between 1 and 7 simply indicate that the normal alphabetic sequence is followed between E and Z and the normal numeric sequence is followed between 1 and 7.)

An *ascending* collating sequence is from low to high (see preceding chart). Arrange the following into ascending collating sequence:

& Z A ; 8 1

• • •

& ; A Z 1 8

—————→

14. Arranging from HIGH to LOW (see chart in preceding frame) produces a *descending* collating sequence.

Arrange the following into descending collating sequence:

Z D C # \$ 8 0

↑

(The digit zero)

• • •

8 0 Z D C # \$

—————→

Chapter 8. File Organization and the Magnetic Tape I/O Medium

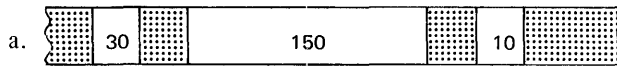
For directions, refer to the summary on the inside of the front cover.

PRETEST 8

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

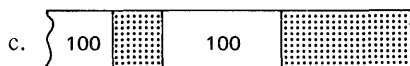
- Records are accessed from magnetic tape in a way that is called:
 - Sequential
 - Index sequential
 - Random
 - Direct
 - I don't know
- The intervals between physical records on magnetic tape are called:
 - Label increments
 - Control fields
 - IRG's
 - Field aligners
 - CFT's
 - I don't know
- A single logical record on magnetic tape:
 - Can be the same size as a physical record
 - Can be part of a physical record
 - Can be smaller than a physical record
 - All of the above
 - I don't know
- Which of the following, on magnetic tape, is usually identified by a key:
 - Physical record
 - Logical record
 - Field
 - Home loop
 - I don't know
- Match the two lists (the numbers in the tape records represent relative amounts of data):



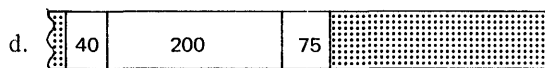
1. Fixed length, blocked



2. Fixed length, unblocked



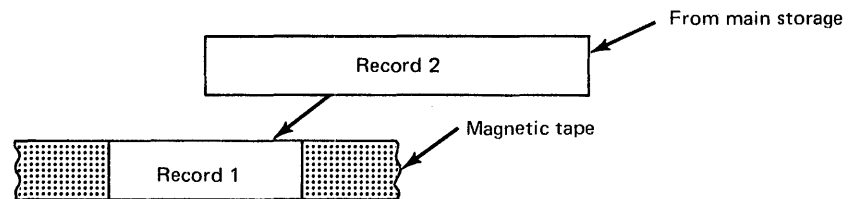
3. Variable length, blocked



4. Variable length, unblocked

- e. I don't know how to match the two lists.

6. The job is a file update on magnetic tape. Match the two lists (more than one item on the right may apply to a single item in the left list):
 - a. Input 1. Transaction
 - b. Output 2. New master
 - c. I don't know how to match the two lists. 3. Old master
7. Batch processing can be described as:
 - a. Processing each single record as it becomes available
 - b. Processing data files by use of a single, master program
 - c. Reading input punched card files and producing output magnetic tape files
 - d. Collecting a group of input records and then processing them
 - e. I don't know
8. Writing the new record 2 (see the following diagram) where record 1 is already located, on magnetic tape, results in (note that record 2 is twice as large as record 1):
 - a. The erasing of both records
 - b. The erasing of record 1 and writing of record 2
 - c. No erasing but the writing of record 2 on top of record 1 so that both record 1 and 2 are on the same location at the end of the operation
 - d. Loss of record 2 with no affect on record 1
 - e. I don't know



9. A file update on magnetic tape (to change, add, and delete records in the existing file) is performed after:
 - a. All records in all input files are arranged in a specified sequence (either all ascending, or all descending).
 - b. All records in all files can be accessed directly.
 - c. All records are in form sequence.
 - d. Records in input files are arranged in any order.
 - e. Records in one input file are arranged in ascending sequence and records in the other input file are arranged in descending sequence.
 - f. I don't know.
10. In order to plan the proper size I/O areas in main storage, the programmer must know:
 - a. The average size of variable-length records
 - b. The maximum size of variable-length records
 - c. The minimum size of variable-length records
 - d. The nominal size of variable-length records
 - e. I don't know

Answers are on the next page.

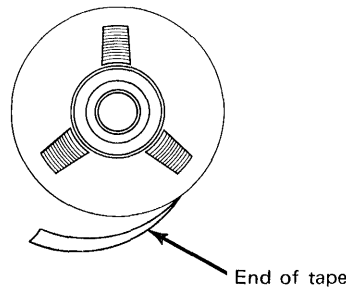
Answers to Pretest 8

1. a
2. c
3. d
4. b
5. a. 4
b. 1
c. 2
d. 3
6. a. 1, 3
b. 2
7. d
8. b
9. a
10. b

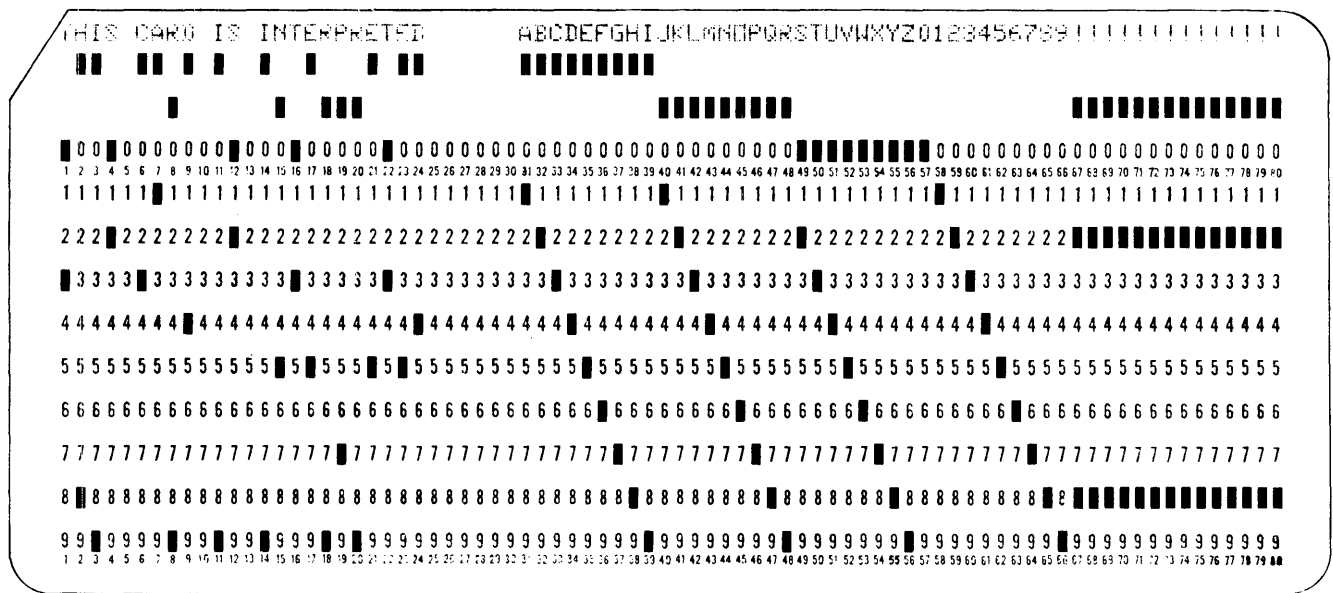
DESCRIPTIVE MATERIAL FOR CHAPTER 8. FILE ORGANIZATION AND THE MAGNETIC TAPE I/O MEDIUM

Magnetic Tape Contrasted with Punched Cards

Data is written on or read from the magnetic tape medium (reel of magnetic tape) by an I/O device called a magnetic tape unit. A reel of magnetic tape looks like this:



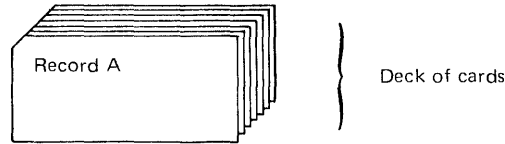
Recall that you can pick up a punched card, look at the punched-hole patterns, and determine the data items that are in the card. Frequently, you will encounter punched cards that have a line of print above the 12 row of the card. This printing is done by a machine called an interpreter and represents the data in the card. The process of producing such printing on the card is called interpreting. An interpreted card looks like this:



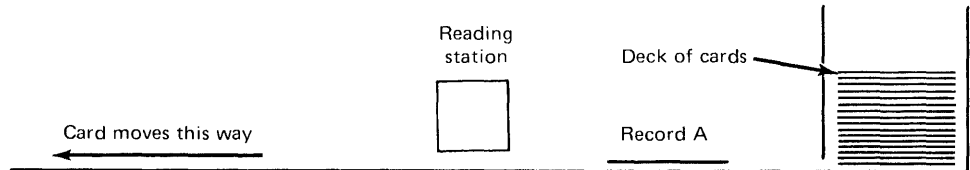
Magnetic tape, on the other hand, contains data in the form of very small spots on the surface of the tape. These spots are magnetic in character and consequently cannot be seen. Also, data is never printed on the edge of tape as it is in cards (see preceding figure). The point here is that you can determine the data in punched cards by looking at them; you cannot know what is recorded on magnetic tape by looking at it.

Data files on reels of magnetic tape, nevertheless, are similar in many respects to data files in punched cards. Therefore, before considering data files on magnetic tape, we will briefly investigate data files on punched cards.

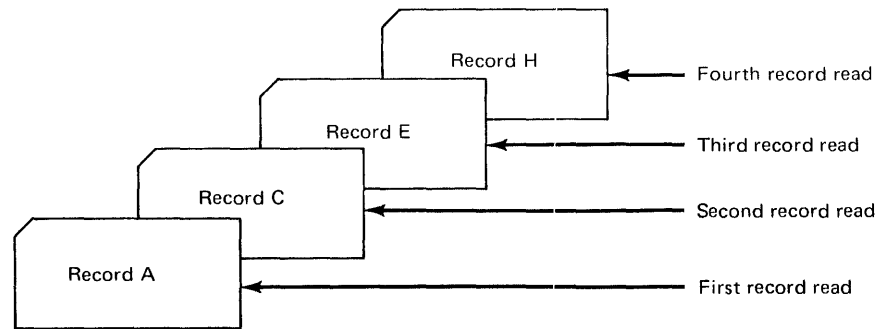
An input data file on punched cards might look like this (a small input data file):



When this deck is put into a card reader, record A is the first record read:



How a card reader works (mechanically or electrically) is unimportant to this discussion. The point is that the card reader reads only one record (card) at a time. Because the card with record A (see preceding figure) is the first one in the card reader, then that is the card that is read first. Now assume that cards are put in the card reader and read as follows:



Before the card reader can read the second card (record C) it must read record A. It cannot read record C before record A. It must read the cards in sequence.

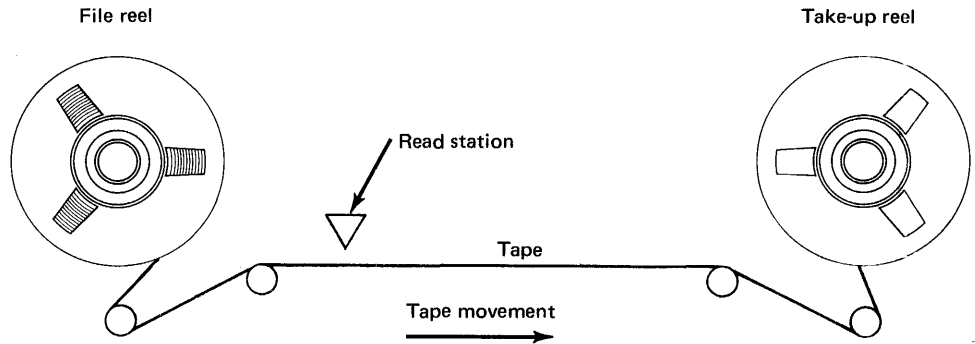
Recall that the CPU can access data directly from main storage. All the CPU needs is a main storage address. Data can be written into or read from the addressed main storage location. No other main storage location need be read first.

Two access methods, then, are:

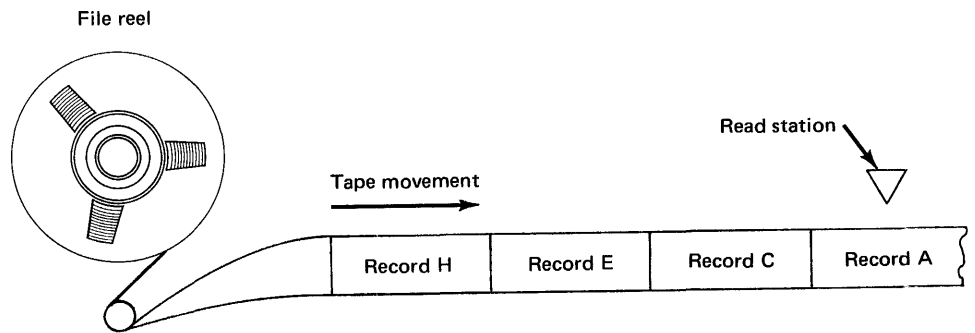
1. *Direct*, exemplified by the way in which data is accessed from main storage by the CPU.
2. *Sequential*, exemplified by the way a card reader accesses records from a deck of cards.

It is important to notice that a card reader cannot use direct access. Notice that the physical characteristics of the card reader impose the restriction that only the sequential access method can be used on input card files.

A file on magnetic tape is read as shown in the following diagram:



Records are read from the input file reel by moving the tape past the read station. (The same station, which is also called the read/write head, is used for writing on the tape during output operations. Some magnetic tape devices are built differently; such mechanical considerations are not, however, important for our purposes.) Now suppose that the records A, C, E, and H are on magnetic tape as follows:



Record A must pass the read station before record C, record C must pass the read station before record E, and finally record E must pass the read station before record H. Records on magnetic tape, then, are accessed sequentially, just as records are accessed from punched cards.

A significant difference between tape and cards, however, is that:

1. A punched card can be moved past the reading station in a card reader only once, under program control. (After it has been read, it is moved into a place, in the card reader, called a stacker.)
2. A record on magnetic tape is wound onto the take-up reel after it has been read. The tape that has been read can be rewound back onto the file reel and then reread, all under program control.

Note, however, that for either cards or tape, the input records may be read again. For the card records, the operator of the card reader must take the cards out of the stacker and manually reload them into the card reader. For the tape records, the tape is first rewound and then reread under program control. (Some magnetic tape devices can read tape backwards as the tape is being wound back on the file reel. The implications of such an operation are beyond the scope of this course.)

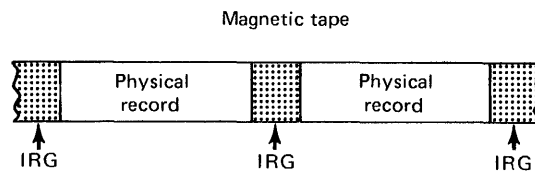
The basic difference between rereading card and tape records is speed. It is much faster to rewind tape under program control than to manually reload cards into a card reader and then restart the system. Also, reading data from magnetic tape is a much faster operation (for the majority of magnetic tape devices) than reading an equivalent amount of data from punched cards. An 8,000-character tape record, for example, can be read much more quickly than 100 cards of 80-columns each.

Physical Records

The amount of data that an I/O device handles (during a data transfer between the device and main storage) is called a *physical record*. Each punched card, for example, is a physical record. Even though not all of the 80 columns of a particular card contain punched-hole patterns, that card is still considered a physical record.

Physical records written on (or read from) magnetic tape are not similarly limited in size. The only physical restriction is determined by the length of the tape, though the program has other restrictions as to the size of the physical records it handles. Usually, only a relatively small portion of the tape on a file reel is needed for a physical record. (For comparison purposes, think of the punched card as holding 80 characters of information and a single reel of tape holding several million characters of information.)

The physical records on magnetic tape are separated by IRG's (Inter-Record Gaps):

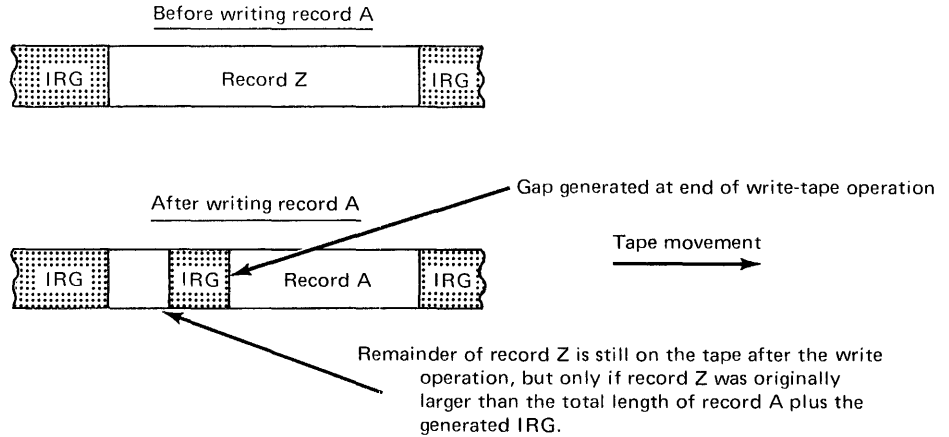


When the CPU requests the magnetic tape device for a read-tape operation, one physical record is sent from the tape device to main storage. Basically, data transfer starts after the first IRG encountered and stops after the end of the physical record.

Writing a physical record on magnetic tape proceeds in the same manner. That is, after a write-tape operation is completed, IRG's are on both sides of the physical record that is recorded on the tape.

Magnetic tape is somewhat similar to main storage in that writing (storing) a physical record in the same location as an existing record (on tape) first causes automatic erasure of the existing record. On tape, however, IRG's are also produced. For example, assume that record A (in main storage) is smaller than record Z, which is on the tape.

Writing record A at the same place on tape as record Z, then, can be shown as follows:



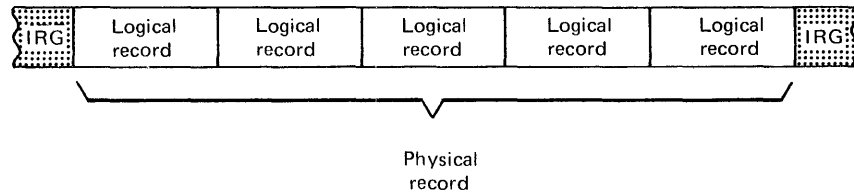
Logical Records

A collection of data items handled as a logical unit by the program is called a *logical record*. In most punched card files, each card is both a physical record and a logical record. This results from the fact that each card is usually handled as a single record (unit record) by most programs.

The difference between logical and physical records can be more clearly described with respect to the magnetic tape medium. Consider a typical card-to-tape operation:

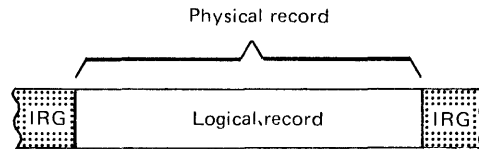
1. Five punched card input records are read at a card reader.
2. The input data from the cards is processed by the CPU.
3. Five logical records are written as one physical output record on magnetic tape. (One write-tape operation is required to write all five logical records.)

The output physical record and five logical records can be represented by:



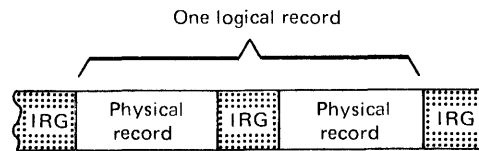
Five physical card records, then, have become one physical (but five logical) tape record. For this example, the five logical records are arranged in main storage by the CPU before the write-tape operation is requested by the CPU. The place that is used for the logical records in main storage is called an output area.

Sometimes a physical record on magnetic tape contains only one logical record:



This arrangement may be used when a logical record is very large. Physical records are frequently several thousand characters long.

Theoretically, a single logical record may require more than one physical record on magnetic tape:



The IRG is not read as part of any record; it simply separates the physical records on tape.

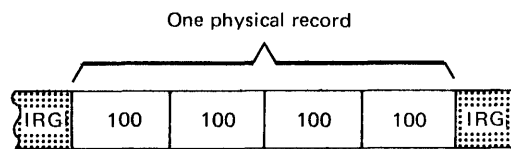
The immediately preceding arrangement, however, is extremely rare. Normally, several logical records make up one physical record as previously described.

Keys

Logical records contain a field called the key (or control field). As discussed in the previous section, the key is used by the program to identify records. The records identified, however, are logical records, not physical records. A key for a particular record is assigned by the programmer. He then writes the program so that records are examined and identified according to the location of the key in the logical records.

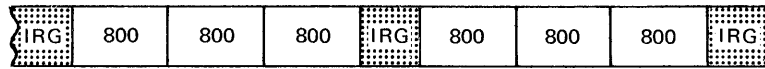
Fixed-Length, Blocked and Unblocked Records

A particular type of record that is always the same size is called a *fixed-length* record. An input punched card record, of 80 columns length, for a payroll application, is a fixed-length record. Logical tape records that are always 100 characters long, as another example, are fixed-length records. Four such fixed-length records can be shown as follows:



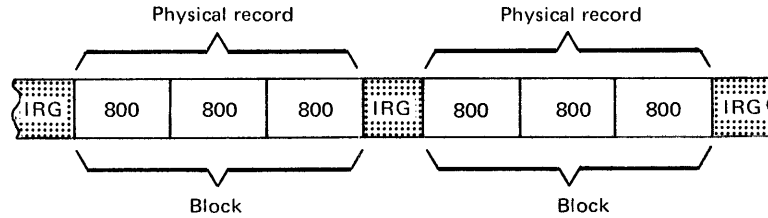
Made up of four logical fixed-length records

Normally, fixed-length logical records are grouped together in a single physical record (as shown in the preceding diagram). Grouping logical records together in this way is called blocking. Another example of fixed-length, blocked records is:

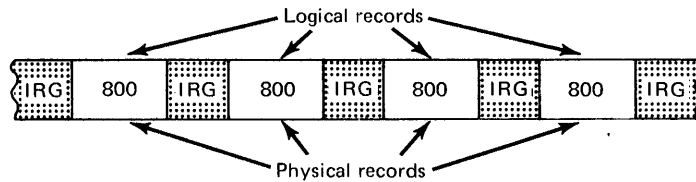


The number of logical records per physical record is sometimes called the blocking factor. In the preceding diagram, the blocking factor is three because three 800-character logical records are in each physical record.

The group of logical records in a single physical record is called a block. Block, then, means the same thing as physical record:



Logical records on magnetic tape are usually blocked. If they are not blocked, they are called unblocked. An example of fixed-length, unblocked records is:

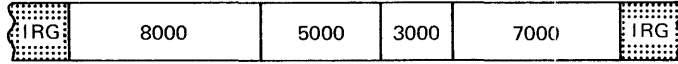


In this book, we use the IRG as a separator between physical records. The term Inter-Block Gap, IBG, is frequently used to specify the same thing.

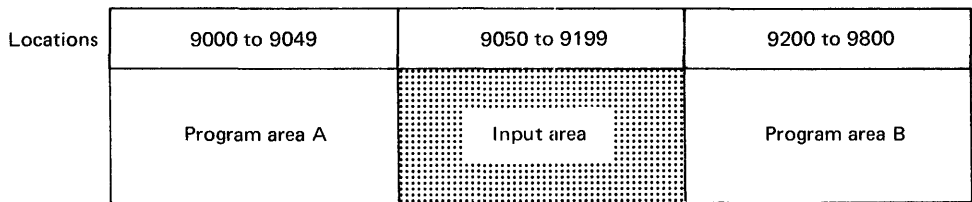
Variable-Length, Blocked and Unblocked Records

A logical record that may contain a different amount of data than other similar records in the same data file is called a *variable-length record*. The maximum size of variable-length records must be known so that the proper input or output area in main storage can be reserved to hold the maximum size record.

Variable-length, blocked records can be represented by:



Areas in main storage must be reserved for input and output records. The reason for reserving areas is related to the fact that storing data in main storage results in erasing data that is already in that location. As an example, assume that an input area of 150 main storage locations is reserved for input records to be read from magnetic tape. Also assume that main-storage locations on both sides of this input area contain the program. This arrangement can be shown pictorially as follows:



As long as input records are stored only into the input area, no problem occurs. But suppose that input data is stored into part of Program Area B. The portion of the program that is in the locations, into which the new data is stored, is erased. Therefore, the maximum size of the variable-length input record must be known. If this is known, storing over other information in storage will not be done because sufficient room has been provided for the largest input record.

A job application that normally requires variable-length records is freight-train accounting. A separate field (in the logical record) is assigned for each freight-train car. Each field contains a description of the contents of a specific car. Clearly, the number of cars in a specific freight train determine the number of fields in this variable-length record. The programmer would have to determine the maximum record size that he could expect. Such information would have to be made available by someone who knows how large the freight trains could be. The maximum record size would have to be used to determine the size of the I/O area for that record in main storage. Most records would be smaller than the maximum, however.

Variable-length, unblocked records on tape can be represented by:

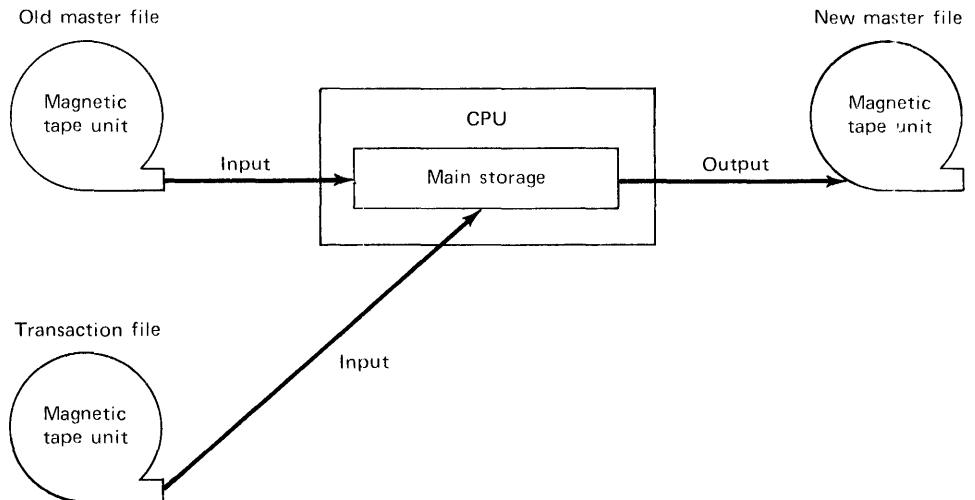


Organization of Records on Magnetic Tape and File Update

The purpose of the following description is to acquaint you with the way in which records on magnetic tape are normally handled.

Many job applications require updating of existing data files, files that are on magnetic tape. An existing data file that contains relatively permanent data is called a *master file*. New input data, which is used to update the data in the master file, comes from a *transaction file*. The file-update job produces, as output data, a new master file. (The old master file is undisturbed by the job and may be retained intact if necessary.)

Assuming that magnetic-tape medium is used for all of the files (old master, transaction, and new master), then the file update can be shown schematically by:



Three basic operations that may be performed during updating of the master file are:

1. *Change* data in an old master record.
2. *Delete* (or remove) an old master record.
3. *Add* a new master record.

An example of file update is the monthly updating of a parts-sold master file. Both the old master and the transaction files are made up of unblocked, fixed-length records. Each old master record contains, among other things:

- Part-number field (key)
- Year-to-date-quantity-sold field

We need concern ourselves here only with the key (part-number field).

The transaction records are of three types:

- *Change*, to update the data in an existing old master record and produce a new master
- *Add* a new master for a new part (one not manufactured before)
- *Delete* an old master for a part that is no longer produced

A master record should be in the old master file for each change or delete transaction record. We do not investigate what would happen if there were not. As a matter of fact, for our purposes the processing steps required to update the master records are not important. We examine only the basic way that magnetic-tape records should normally be arranged on the tape.

It is important, with respect to speed of processing, that records in both the master and transaction files be arranged in the same sequence. For our example, all master and transaction records are on their respective reels of magnetic tape in ascending sequence. They are sequenced by the key, which is a three-position part-number field. The example records are:

<i>Old Master Records</i>	<i>Transaction Records</i>	
<i>Part Number (Key)</i>	<i>Part Number (Key)</i>	<i>Record Type</i>
010	010	Change
	011	Add
013	013	Delete
014	014	Change
	015	Add
016	016	Change
017		
018		
019	019	Change
020	020	Change
021	021	Change

Notice that the records are arranged in ascending sequence in both files.

The following operations occur during processing of the first transaction record:

1. Read master record (key = 010).
2. Read transaction record (key = 010; this is a change record).
3. Because the keys are the same (010 = 010), the master record is updated according to data in the transaction record.
4. Write an updated master on the new master file.

The operation then proceeds as follows:

1. Read next master record (key = 013).
2. Read transaction record (key = 011; this is an add new master record).
3. Because the key of the old master is higher, the new master is written on the new master file.

Next:

1. Another transaction record is read (key = 013; this record specifies deleting of an old master, the old master with a key of 013).
2. Because the key of the last old master matches the key of the delete record, a record is not written on the new master file. The old master, with a key of 013, is thereby deleted from the file.

Up to this point, only the following two records are on the new master reel:

<i>Part Number (Key)</i>	<i>Description</i>
010	Changed old master
011	New Master

The operation continues in a similar manner.

To find out what happens when records are not sequentially organized on tape, consider a second example. Suppose that the records in the old master file are still (as above) in ascending sequence by key but the records in the transaction file are not arranged in any sequence:

<i>Old Master Records</i>	<i>Transaction Records</i>	
<i>Part Number (Key)</i>	<i>Part Number (Key)</i>	<i>Description</i>
010	014	Change
013	020	Change
014	013	Delete
016	021	Change
017	011	Add
018	010	Change
019	019	Change
020	015	Add
021	016	Change

Because the records that are read from magnetic tape must be accessed sequentially, notice what must be done in processing the out-of-sequence transaction records. First:

1. Read master record (key = 010).
2. Read transaction record (key = 014; this is a change record).

But the keys do not match $010 \neq 014$. (The symbol \neq means does not equal.) The old master record has a lower key than the transaction record. Therefore, the first old master (key = 010) is written as output on the new master reel. Then the next old master is read. Its key (013) is still lower than the key of the first transaction record (014). Therefore, this master is also written on the new master reel.

Finally, a master with a key of 014 is read and updated according to the change transaction record. The updated master is written out on the new master reel.

A similar series of operations occurs after the next transaction record (key = 020) is read. Old masters are written on the new master file until the old master with a key of 020 is found. After the old master (key = 020) has been updated and written out on the new master reel, the new master reel contains the following:

<i>Record Key</i>	<i>Description</i>
010	Unchanged old master
013	Unchanged old master
014	Updated master
016	Unchanged old master
017	Unchanged old master
018	Unchanged old master
019	Unchanged old master
020	Updated master

Now notice what happens. The next transaction record is a *delete* record (key = 013). But where is the old master with a key of 013? On the new master reel. Therefore, the new master reel must be rewound and then read so that the old master can be found and deleted.

The important thing to notice is that a lot of time is being wasted:

- Time to read records and examine their keys that must be bypassed because they are the wrong ones
- Time to rewind the tape

These two problems do not occur when both the master and the transaction files are arranged in the same sequence (whether ascending or descending). Notice, then, that efficient use of magnetic tapes requires that records on the tape be in sequence. Also notice that we considered only a few records. File update typically involves updating of thousands of records.

Frequently, the original transaction records are received as punched card records that are not arranged in any sequence. In this case, sorting of the records is performed before the file update is attempted. How the sorting is done depends upon the number of punched cards in the input file and the procedures used in the data processing installation.

One way is to sort the cards by key (in a card sorting machine called a sorter, which is not connected to the data processing system) and then put the cards into an input device (card reader). Processing of the sorted transaction records and the old master records (on tape) then result in production of a new master file.

Another way is to first sort the transaction records by a program called a sort routine. The sorted transaction records are written as output on magnetic tape to produce a sequentially organized transaction file. This file is then processed against the old master records as previously described.

Batch Processing

A file update on magnetic tape is normally performed by first collecting, over some period of time, transaction records. Then, after a group of records has been collected and sequenced, that group is processed. This type of processing is the only efficient way to process records when a medium that can be accessed only in a sequential manner is used (for example, punched cards or magnetic tape).

The procedure of collecting records and then processing them as a group, rather than individually, is known as *batch processing*. To see how inefficient the handling of a sequential file is if batch processing is not used, consider what would happen if a single record is processed as it is received in the data processing department of a company. The job to be done is the same file update (parts-sold masters) described previously. The job proceeds as follows:

1. When a single transaction record is received, it is sent to be keypunched. After it is keypunched it is given to the system operator.
2. The system operator, provided no other job is in progress,
 - a. Loads the parts-sold file update program (object program) into main storage
 - b. Gets the old master file (a reel of magnetic tape) and puts it in a magnetic tape device
 - c. Puts the single transaction record in a card reader. (No sorting is necessary because there is only one transaction record.)
 - d. Gets a new reel of magnetic tape (for new master records) and puts it on another magnetic tape device
 - e. Starts the system
3. The CPU reads the transaction record and starts looking for the corresponding master.
Note: Because a new master tape is being produced, the entire old master file must be written on the new master.
4. Assume the transaction is a change record. After the old master is found, the change is made and then the remainder of the old master file is written on the new master reel.

Now suppose that just after the job is completed (the tape reels have been put away and another program is being executed) a messenger from the keypunch department brings the next transaction record to the system operator. The entire series of operations must be performed again!

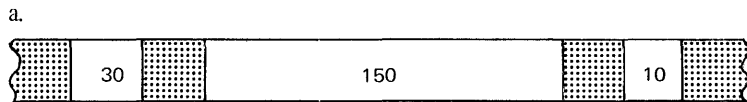
Therefore, for file update jobs on magnetic tape, input records are batched before the job is run. (The number of records collected and the period of time over which they are collected depends upon the requirements of the job. Such considerations are beyond the scope of this book.)

POST-TEST 8

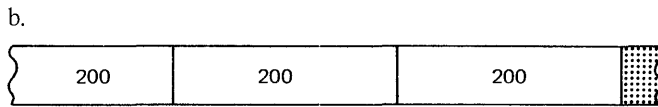
Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

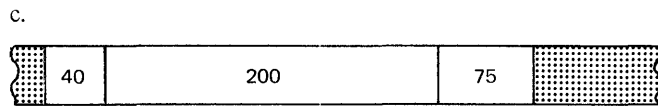
1. The method in which records are accessed from magnetic tape is called:
 - a. Random
 - b. Indexed sequential
 - c. Sequential
 - d. Direct
 - e. I don't know
2. Physical records on magnetic tape are separated by intervals that are called:
 - a. Control fields
 - b. IRG's
 - c. CFT's
 - d. Label increments
 - e. Field aligners
 - f. I don't know
3. On magnetic tape, a single logical record:
 - a. Can be smaller than a physical record
 - b. Can be part of a physical record
 - c. Can be the same size as a physical record
 - d. All of the above
 - e. I don't know
4. Of the following, which is usually identified by a key, on magnetic tape:
 - a. Field
 - b. Logical record
 - c. Physical record
 - d. Home loop
 - e. I don't know
5. Match the two lists (the numbers in the tape records represent relative amounts of data):



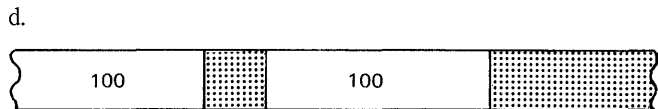
1. Fixed length, blocked



2. Fixed length, unblocked



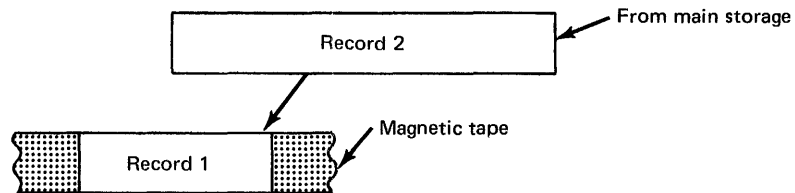
3. Variable length, unblocked



4. Variable length, blocked

e. I don't know how to match the two lists.

6. For a file-update job on magnetic tape, match the two lists (more than one item on the left may apply to a single item on the right):
 - a. Transaction
 - b. Old master
 - c. New master
 - d. I don't know how to match the two lists.
 1. Output
 2. Input
7. A description of batch processing is:
 - a. Reading input punched card files and producing output magnetic tape files
 - b. Collecting a group of input records and then processing them
 - c. Processing data files by use of a single, master program
 - d. Processing each single record as it becomes available
 - e. I don't know
8. Record 2, which is twice the size of record 1, is written on magnetic tape in the same location as record 1 (see the following diagram). As a result:
 - a. Record 2 is lost with no affect on record 1.
 - b. Record 2 is written on top of record 1 so that both records are on the tape in the same location at the end of the operation.
 - c. Record 1 is erased and record 2 is written.
 - d. Both records are erased.
 - e. I don't know.



9. A file update can be performed on magnetic tape to change, add, and delete records in the existing file, but only after:
 - a. All records are in form sequence.
 - b. Records in one input file are arranged in ascending sequence and records in the other input file are arranged in descending sequence.
 - c. All records in all files can be accessed directly.
 - d. Records in input files are arranged in any order.
 - e. All records in all input files are arranged in a specified sequence (either ascending, or descending).
 - f. I don't know.
10. The programmer, in order to plan the proper size I/O areas in main storage, must know:
 - a. The nominal size of variable-length records
 - b. The average size of variable-length records
 - c. The maximum size of variable-length records
 - d. The minimum size of variable-length records
 - e. I don't know

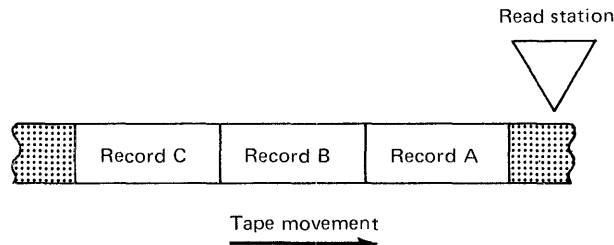
Answers are on the next page.

Answers to Post-Test 8

1. c
2. b
3. d
4. b
5. a. 3
b. 1
c. 4
d. 2
6. a. 2
b. 2
c. 1
7. b
8. c
9. e
10. c

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 8. FILE ORGANIZATION AND THE MAGNETIC TAPE I/O MEDIUM

1. The following diagram shows three records on magnetic tape:



The records can be read only when the magnetic tape moves past the read station.

In the preceding diagram, record A is read before record B. Record C is read

- a. before or
 - b. after
- record B?

• • •

- b. *after* record B.

2. Accessing records one after the other in the order they appear (as on magnetic tape) is called the sequential access method. The CPU, on the other hand, when it addresses any main-storage location, accesses that location directly. It is not necessary for the CPU to access any other main-storage location before the addressed location.

The CPU can, however, read the data from a series of consecutive main-storage locations in the same manner as a magnetic tape device reads records from magnetic tape.

The CPU is capable of accessing data from main storage by use of the:

- a. Sequential access method
- b. Direct access method

• • •

Both

3. Records can be read from magnetic tape by a magnetic-tape device that must access the records by:

- a. The direct access method
- b. The sequential access method

• • •

- b. The sequential access method

4. The locations of records on magnetic tape are not addressable. All main-storage locations:

- a. Are addressable by the CPU
- b. Must be sequentially accessed by the CPU

• • •

- a. Are addressable by the CPU

5. Data on magnetic tape:
 - a. Has assigned addresses
 - b. Must be read sequentially as the tape passes the read station in a magnetic-tape device

• • •

 - b. Must be read sequentially as the tape passes the read station in a magnetic-tape device

6. Data is recorded on magnetic tape in the form of magnetic spots. Therefore, you cannot see data on magnetic tape. The only way the CPU can find a record on magnetic tape is to initiate a read tape operation. The record is then read from magnetic tape by the magnetic-tape unit.

The CPU identifies a record that has been read from magnetic tape by:

 - a. Examining the key in the record, which is in main storage after the tape-read operation.
 - b. First punching the record in an 80-column card and then reading the card at a card reader.

• • •

 - a. Examining the key in the record, which is in main storage after the tape-read operation.

7. A record that is treated as a logical unit by the program is called a *logical record*. Each record of a group of employee-payroll records, as an example, contains the same fields as other records in the group. Each record is for a different employee and is handled as a logical unit by the program. These example employee-payroll records:
 - a. Are logical records
 - b. Contain a key that is used to identify the record

• • •

Both

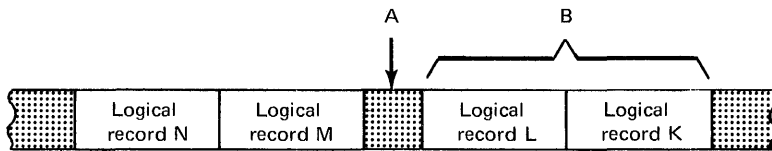
8. A logical record is identified by the program according to:
 - a. Its location on magnetic tape.
 - b. The data item in the key of the record.

• • •

 - b. The data item in the key of the record.

9. A record that is handled as a physical unit by an I/O device is called a *physical record*. Physical records (which can contain logical records) on magnetic tape are separated from each other by IRG's (Inter-Record Gaps). In the following diagram:

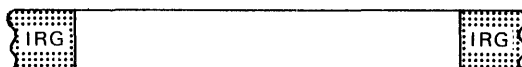
A points to _____.
 B points to _____.



• • •

A points to *an IRG*.
 B points to *a physical record*.

10. During a read tape operation, a magnetic tape device sends the data that is between two IRG's to main storage:



The amount of data sent to main storage during the tape-read operation is called a _____.

• • •

physical record

11. During a write-tape operation, data is sent from main storage to the magnetic-tape device. The magnetic-tape device causes IRG's to be produced on both sides of the data that is written on the tape during the write-tape operation.

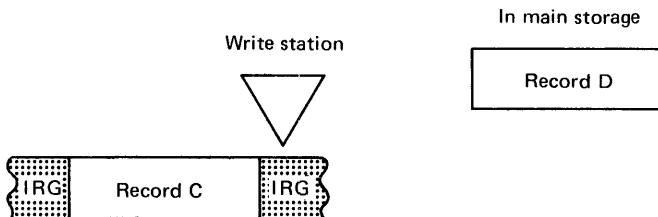
The data written on tape during a single write-tape operation is called _____.

• • •

a physical record.

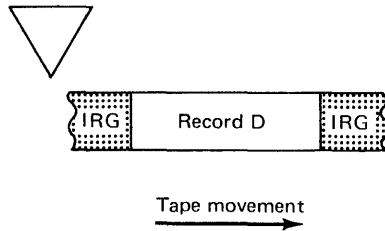
12. Recall that data already in a main-storage location is erased when new data is stored into the same location. The operation of writing on magnetic tape is similar, except that IRG's are produced on both sides of the record.

- Magnetic tape before write operation:



- Magnetic tape after record D is written (assume that record D is larger than record C):

Write station



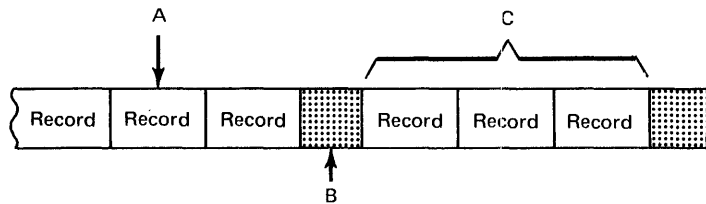
Writing a physical record on magnetic tape:

- Should be done only if there is no data on the tape
 - First causes erasure of data in the area that is being written on
- • •
- First causes erasure of data in the area that is being written on
13. New data can be written on magnetic tape that already contains data:
- Because the data on the tape is erased before the new data is written.
 - Because all programs are written to ignore the data which stays on the tape, after new data is written.
- • •
- Because the data on the tape is erased before the new data is written.
14. Usually, more than one logical record is in a single physical record. Fill in the blanks according to the following diagram:

A points to _____.

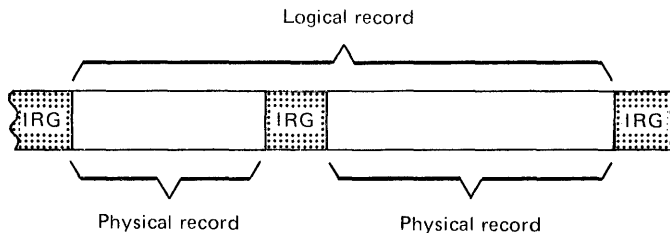
B points to _____.

C points to _____.



- • •
- A points to *a logical record*.
- B points to *an IRG*.
- C points to *a physical record*.

15. Sometimes only a single logical record is contained in a physical record. It is also possible, though very unusual, for a single logical record to be in more than one physical record:



Logical records:

- a. May be the same size as physical records
- b. May be larger than a physical record

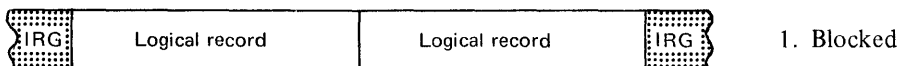
• • •

Both (Not, of course, both at the same time. Both answers, however, are correct. Logical records may be the same size or larger than physical records.)

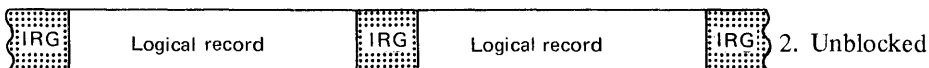
16. The process of grouping two or more logical records into a single physical record is called *blocking*. Such an arrangement of logical records is called blocked records. When logical records on tape are arranged such that one, or less than one, logical record is in each physical record, the logical records are called unblocked records.

Match the two lists:

a.



b.



• • •

- a. 1. Blocked
- b. 2. Unblocked

17. When logical records in a file are always the same length, they are called *fixed-length* records. When they are of varying (or different) lengths, they are called *variable-length* records.

In the following diagrams, the numbers represent amounts of data. Match the two lists:



• • •

- a. 2. Fixed length
b. 1. Variable length

18. The programmer must reserve areas of main storage for input and output records. Certain data files are made up of only fixed-length records. The size of the main-storage area for such records is the same as the size of each physical record.

For example, an input file is made up of 80-column punched cards. Each card contains one 80-column logical record. Assume that each card column corresponds to one main-storage location. The input area in main storage for this card file:

- a. Is made up of 80 main-storage locations
b. Is made up of 60 main-storage locations

• • •

- a. Is made up of 80 main-storage locations

19. An input area in main storage for a file that contains variable-length records is handled somewhat differently than one for fixed-length records. The area reserved is determined by the largest variable-length record expected.

What is the size of the input area required if the following represent the only record lengths expected? (Assume that each main-storage location can hold one character.)

- 500 characters
400 characters
800 characters
300 characters

• • •

- 800 main-storage locations.

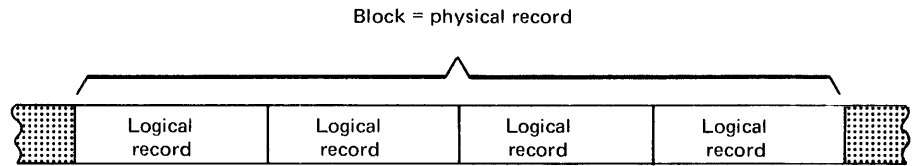
20. In order to reserve an input area in main storage for variable-length input records, the programmer must know:

- a. The size of the smallest output record
b. The average size of the variable-length records in the input file

• • •

- Neither (He must know the largest variable-length input record.)

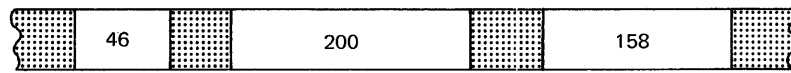
21. An area, in main storage, can be reserved for variable-length output records when (in your own words) _____
- • •
- the size of the largest variable-length output record is known.
22. Two types of logical records, then, are variable-length and fixed-length. The program identifies either type, with respect to the kind of information in that record (such as the data in an employee-payroll record), by:
- The key (or control field) in the logical record
 - The size of the logical record
- • •
- The key (or control field) in the logical record
23. When several logical records are in one physical record, the logical records are called:
- Blocked
 - Unblocked
- • •
- Blocked
24. A block of logical records is not identified by key. Instead, a key is in each logical record in the block. A physical record is the same thing as a block of records:



- A physical record:
- Contains a key that identifies the physical record
 - Is not identified by a key
- • •
- Is not identified by a key (each logical record is identified by a key, not the physical record)
25. A key is used to identify a _____ record.
- • •
- logical

26. Match the two lists:

a.



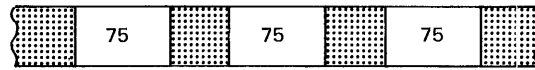
1. Fixed-length, unblocked

b.



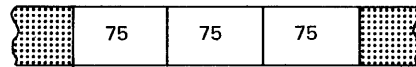
2. Variable length, blocked

c.



3. Fixed length, blocked

d.



4. Variable length, unblocked

• • •

- a. 4. Variable-length, unblocked
- b. 2. Variable-length, blocked
- c. 1. Fixed-length, unblocked
- d. 3. Fixed-length, blocked

File Update

27. Magnetic-tape, file-update jobs typically involve:

- *Changing* data in an old record
- *Deleting* an old record
- *Adding* a new record

The *old master file* is updated through use of information in records that make up the *transaction file*. The result of the file-update is a *new master file*. Match the two lists:

- a. Transaction file 1. Input
- b. Old master file 2. Output
- c. New master file

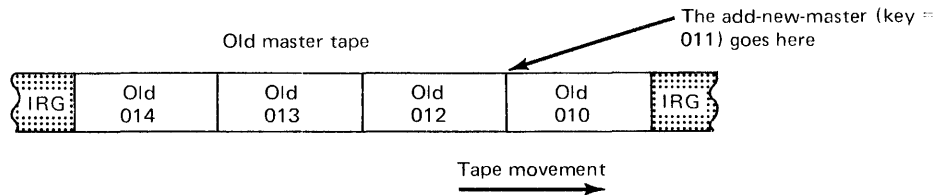
• • •

- a. 1. Input
- b. 1. Input
- c. 2. Output

28. Suppose, instead of producing a new master file, an attempt is made to update by writing on the old master tape. First assume that all logical records are fixed-length and blocked, four to a physical record. Our example input records are identified by key as follows:

<i>Old Master (Key)</i>	<i>Transaction (Key)</i>	<i>Type of Transaction Record</i>
010	010	Change old master
012	011	Add new master
013	012	Change old master
014	013	Change old master

Now notice what happens as a result of processing the first add-new-master transaction record (key = 011):



The preceding diagram shows where to write the new master to keep the records sequentially organized by key. The add-new-master transaction record (key = 011):

- Cannot be written because there is no room on the tape between the old master records with keys of 010 and 012
- Can be written because IRG's are between the logical records, and IRG's are always larger than the records

• • •

- Cannot be written because there is no room on the tape between the old master records with keys of 010 and 012

29. While it is possible to move all the old master records down the tape and then write the new master (key = 011, see preceding frame), a lot of time would be required to do this. To avoid performing such a time-consuming operation:

- Output is written on a new master reel of magnetic tape.
- New master records, resulting from add-new-master transaction records, can be skipped.

• • •

- Output is written on a new master reel of magnetic tape.

30. Because records on magnetic tape are not addressable, they should be arranged in sequence on both the old master and on the transaction files.

Records are written on the new master file (which becomes the old master file for the next update):

- In sequence according to key
- In random order

• • •

- In sequence according to key

31. Records in both the old master and the transaction files should be arranged in the same sequence. Both files should be in ascending sequence or both files should be in descending sequence. Records on the old master and transaction files are in sequence according to:
- a. IRG
 - b. Key
- • •
- b. Key
32. Suppose that records in the old master file are incorrectly set up in random order, and records in the transaction file are in sequence. The first transaction record read is a *change-old-master* and has a key of 1141. The CPU, in order to locate the old master with a key of 1141:
- a. Directs the magnetic tape device to read records from the old master reel of magnetic tape and examines the key of each until the desired record is found
 - b. Directs the magnetic tape device to go directly to address 1141 on the old master reel of magnetic tape
- • •
- a. Directs the magnetic tape device to read records from the old master reel of magnetic tape and examines the key of each until the desired record is found
33. If the records are not arranged in the same sequence on both the old master and transaction reels of tape:
- a. Records written on the new master reel are produced only after a lot of record searching (rereading and rewinding tape) of the old master file.
 - b. The file update takes much longer than if the records are arranged in the same sequence on both files.
- • •
- Both
34. Suppose that only one transaction record (a change record on a punched card) is used to update a single record on the old master reel of tape. The CPU:
- a. Must examine a large number of old master record keys before the desired old master record can be found
 - b. Can direct the magnetic tape device to directly address the desired old master record on the old master reel.
- • •
- a. Must examine a large number of old master record keys before the desired old master record can be found
35. Now suppose that another single change transaction record is to be used for a second file update. The CPU must again examine a large number of old master record keys before the old master (on magnetic tape) can be found. But such record searching was already done on the first file update (preceding frame). During the two file updates, for the two single change records, many old master records were:
- a. Read from tape and their keys examined twice, once for each file update
 - b. Read from tape and their keys examined only once
- • •
- a. Read from tape and their keys examined twice, once for each file update

36. Searching the old master records only once (to update according to data in the two transaction records) can be done by:
- a. Arranging the old master records in random order on the tape
 - b. Waiting until both transaction records are available and then processing them in one file update
- • •
- b. Waiting until both transaction records are available and then processing them in one file update
37. To avoid record searches on magnetic tape, where most records found are not to be updated, many more than two records are collected before the file update is performed. This procedure of collecting records over a period of time and then processing them is called *batch processing*. Batch processing is particularly useful in updates of sequentially organized data files.
- Sequential file organization is used in data files that are in punched-card or magnetic tape. Batch processing is applicable to data files in the:
- a. Magnetic-tape medium
 - b. Punched-card medium
- • •
- Both
38. Collecting records over a period of time and then processing them is called_____.
- _____.
- • •
- batch processing

Chapter 9. File Organization and Direct Access Storage Devices (DASD)

For directions, refer to the summary on the inside of the front cover.

PRETEST 9

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. A cylinder, in a DASD, is composed of:
 - a. The tracks that are available for reading or writing at each position of the access mechanism
 - b. The number of accesses performed during one movement of the access mechanism
 - c. All of the tracks across one disk surface
 - d. All of the tracks of all of the disks
 - e. I don't know
2. The index for an indexed-sequential file organization contains:
 - a. The addresses of groups of records and the key of the last record in each group
 - b. Identifiers and counters of intermediate records
 - c. Page numbers and symbolic entry names
 - d. A random directory of record ID's
 - e. I don't know
3. On DASD, a data file that is organized by the indexed-sequential method contains records that:
 - a. Are arranged in ascending sequence by key
 - b. Need not be arranged in sequence because of the addressing capabilities of the DASD
 - c. Must be arranged in ascending sequence by check character
 - d. Can be located from address information in the index when they are arranged in random order on the DASD
 - e. I don't know
4. A seek operation is performed to:
 - a. Find a record in a data file that is organized on DASD by sequentially reading the keys
 - b. Read a data record from DASD and store that record in main storage
 - c. Position the access mechanism at some specified cylinder or switch to another read/write head in the same cylinder
 - d. Cycle the access mechanism piston chamber to one-half of the cylinder length
 - e. Locate and identify one of the several records on a track
 - f. I don't know
5. The operation of finding a specific record key on a track (in a DASD) that contains several records, when that track is being read, is called a:
 - a. Key-sequence access
 - b. Track flagging operation
 - c. Seek operation
 - d. Search operation
 - e. I don't know

6. A direct file organization on a DASD is one in which the address (or location) of a record is:
 - a. Determined by the amount of data in the record
 - b. Specified in ascending, collating sequence according to the high-order character of the key of that record
 - c. Identical to the main storage address where that record is stored before processing
 - d. Derived from the value of the key of that record according to some specified mathematical rule of procedure
 - e. I don't know
7. The term "inline" means that:
 - a. Input records can be processed as they are received and need not be sorted and/or batched.
 - b. Record keys are aligned in ascending sequence throughout a cylinder.
 - c. The DASD carries out operations specified by either of two CPU's to which it is connected.
 - d. The data on a DASD is exactly duplicated in main storage.
 - e. I don't know
8. Match the two lists (consider that all files are on DASD):
 - a. Reorganize the file when overflow areas are full and new records must be added to the file. 1. Indexed sequential
 - b. Rewrite the file whenever new records are added to the file. 2. Sequential
 - c. I don't know how to match the two lists.

Answers are on the next page.

Answers to Pretest 9

1. a
2. a
3. a
4. c
5. d
6. d
7. a
8. a. 1
b. 2

Direct Access in DASD

Certain file organizations that are used with the input/output direct access storage devices (DASD's) are constructed to take advantage of the direct access capabilities of these devices. We, therefore, consider what is meant by direct access (when applied to DASD) before investigating related file organizations. (DASD is pronounced daz-dee.)

Recall that the CPU can directly access any location in main storage in order to read or store data. All that is required is the address of the desired location. No other main-storage location need be addressed first. The CPU, then, uses the direct access method when it reads data from or writes data into main storage.

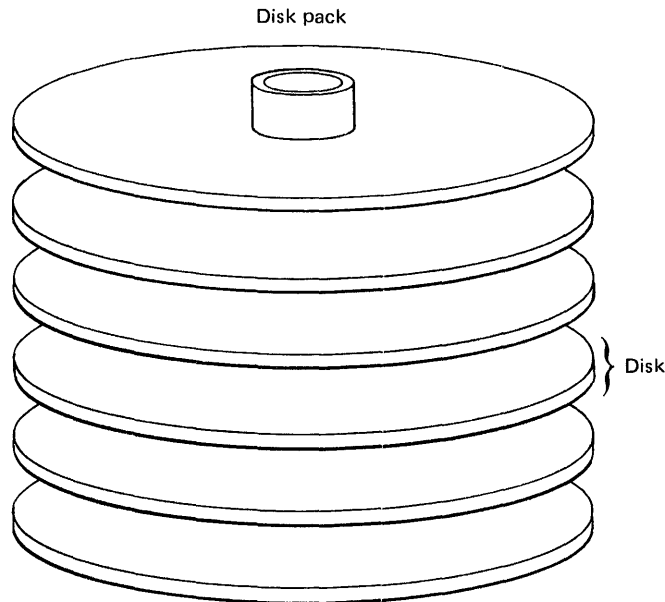
On the other hand, the sequential access method must be used to read data from or write data on magnetic tape. This results from the fact that the only way to read from or write data on tape is to move the medium (magnetic tape) past the read/write station (read/write head). The read/write head in a magnetic tape device does not move.

In DASD's, which are direct access input/output devices, both the medium and the access mechanism are positioned in order to read or write data. (Usually, the medium is in constant motion.)

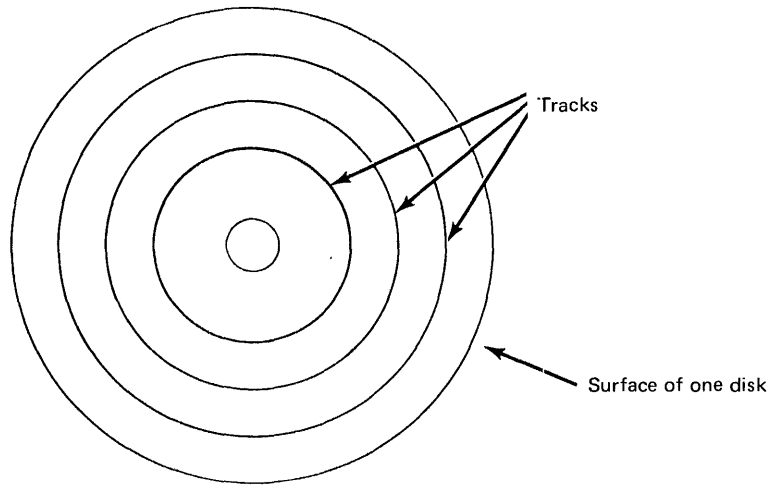
Disk Storage Device

There are a number of such direct access storage devices. We, however, investigate only the DASD called a disk-storage device. (Other DASD's, such as the drum or strip reader devices, operate in a manner similar to the disk-storage device.)

The medium used in the disk-storage device is called a disk pack, which is made up of several disks attached together (sometimes only one or two disks are in the pack):

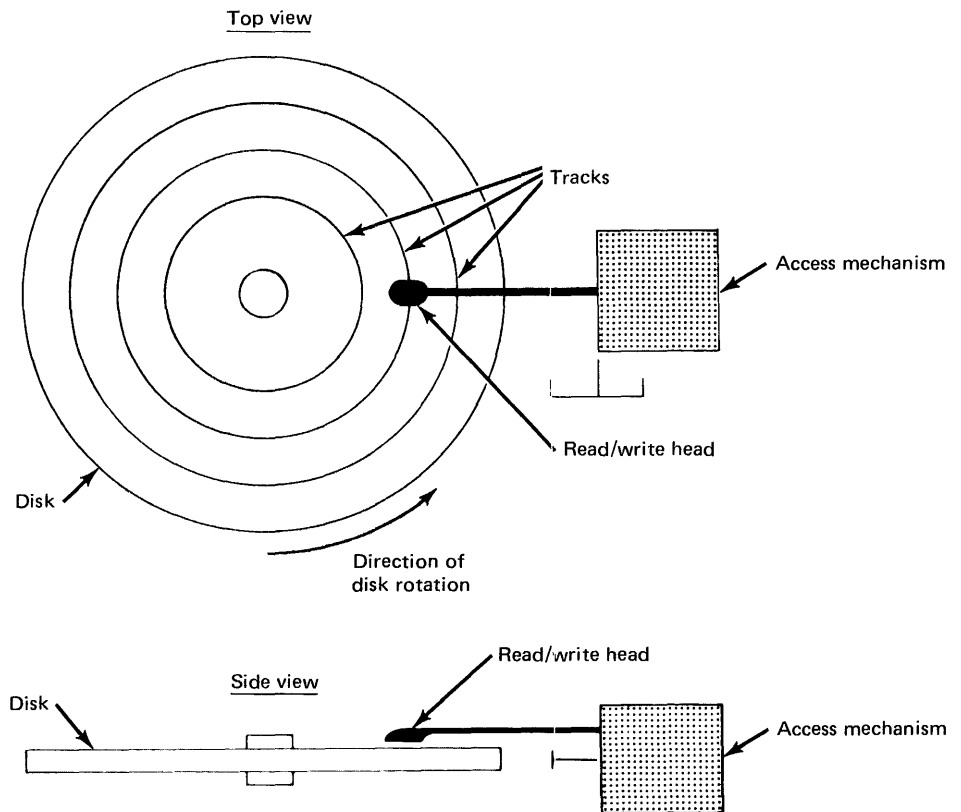


First, consider a single disk. Data is recorded on a disk surface in the form of magnetic "spots." These spots can be recorded only in specific tracks:



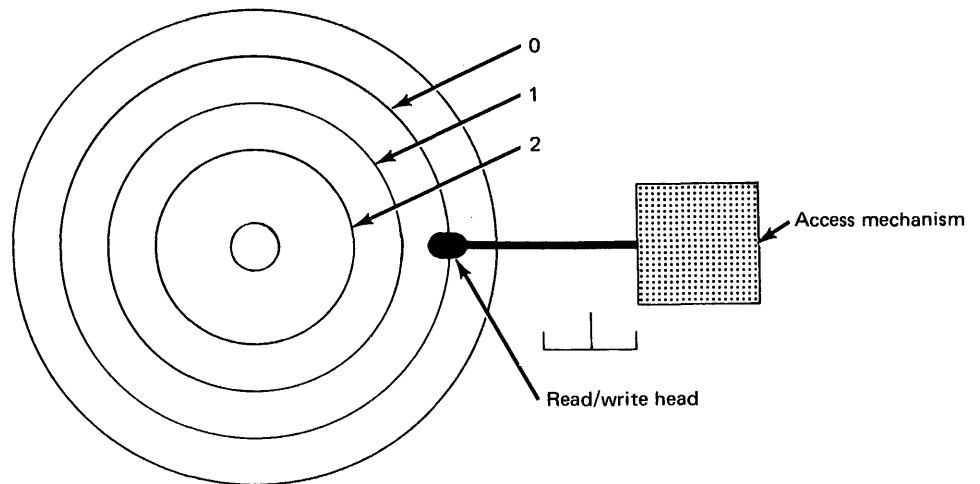
(Each disk has many more than three tracks. For the moment, however, we consider only three.)

Data is written on or read from a disk surface by a read/write head. The read/write head is attached to an access mechanism:



During normal operation, the disk rotates constantly at high speed under the read/write head; the disk does not stop even when the access mechanism is moving the read/write heads to a new track.

Assume that the three tracks have the addresses 0, 1, and 2, as shown in the following diagram:



When the read/write head is in the position shown in the preceding diagram (over the track with address 0), data can be written on or read from track 0 only. The read/write head must be at a track position (address) before data can be read or written.

The access mechanism moves the read/write head from one track to another during a *seek operation*. (A seek operation, performed by the disk-storage device, is a result of the CPU executing an instruction that requests such an operation.) The access mechanism can stop only at certain specific positions. Because the read/write head is attached to the access mechanism, the head stops only where the access mechanism allows it to stop. For our example, the read/write head can stop only at the track-0, track-1, or track-2 position.

After a seek operation is completed and the read/write head has stopped moving, data can be read from or written on the track that is under the read/write head. Data can be written at high speed because the disk that the track is on is rotating at high speed.

So that you have some idea of relative data capacity and speed of reading or writing, note that one particular kind of disk pack in a specific machine that is in use now in the data processing industry can:

1. Rotate at 2,400 revolutions per minute
2. Hold about 3,000 characters of data on one track (compare to an 80-column punched card)

Data from this disk can be read or written at a rate of about 150,000 characters per second. All 3,000 characters on a track, then, can be read and stored in main storage in about two one-hundredths of a second. Looking at it another way, the 3,000 characters on a single track can be read over about 42 times in one second. Reading it once, of course, is the usual procedure.

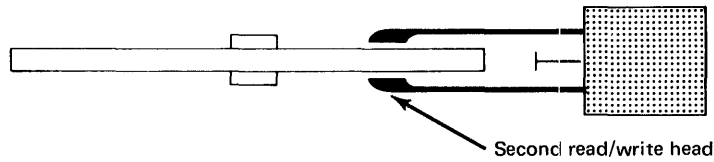
The important facts that we have examined so far are:

1. DASD's are direct access input and output storage devices.
2. Both the medium and the read/write head move.
3. Data is recorded on addressable tracks.
4. A seek operation involves moving the read/write head to an addressed track.

Cylinder

So that the bottom of the single disk can be used for data, another read/write head can be added:

Side view



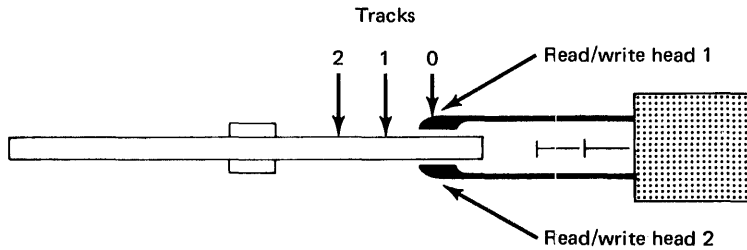
Both surfaces of the disk, for our description, have three tracks. Both sets of tracks have the *same* addresses:

	<i>Inside Track</i>	<i>Middle Track</i>	<i>Outside Track</i>
Top surface of disk	2	1	0
Bottom surface of disk	2	1	0

The locational difference between the two sets of tracks can be specified by assigning an address of 1 to the top read/write head and an address of 2 to the bottom read/write head. A two-digit number can now specify any single track on the example disk:

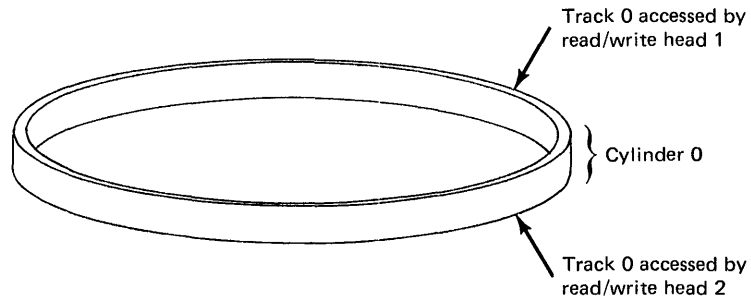
<i>Track</i>	<i>Read/Write Head</i>	<i>Complete Address</i>
0	1	= 01
0	2	= 02
1	1	= 11
1	2	= 12
2	1	= 21
2	2	= 22

Now notice what happens if a seek to either address 01 or 02 is performed. The access mechanism moves *both* read/write heads to their outermost positions:



Two tracks, then, are available for the reading or writing of data at each position of the access mechanism; track 0 read by read/write head 1, and track 0 read by read/write head 2.

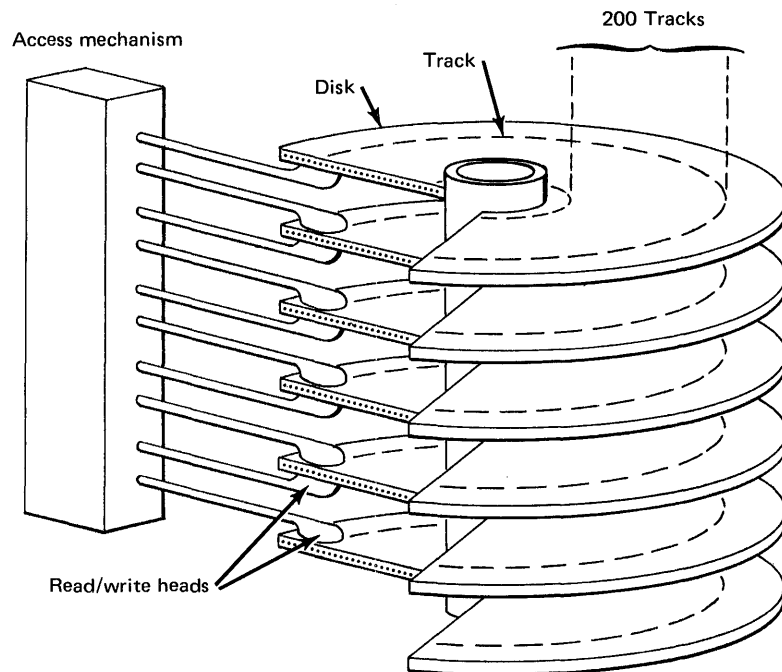
The number of tracks available for the reading or writing of data at each position of the access mechanism is called a *cylinder*. For our example, then, the two tracks available at each position of the access mechanism make up a cylinder. The word “cylinder” is derived from the cylindrical form represented by the tracks. For the previous example, the cylinder is:



For greater disk capacity, expansion can be made by:

1. Attaching several disks together to form a disk pack
2. Providing many tracks, for example 200, on each disk surface
3. Providing a sufficient number of read/write heads to read the disk surfaces

This example arrangement looks like the following:



Because there are ten read/write heads (in the preceding diagram), ten tracks are available for the writing or reading of data at each position of the access mechanism. In this case, then, the cylinder is made up of ten tracks.

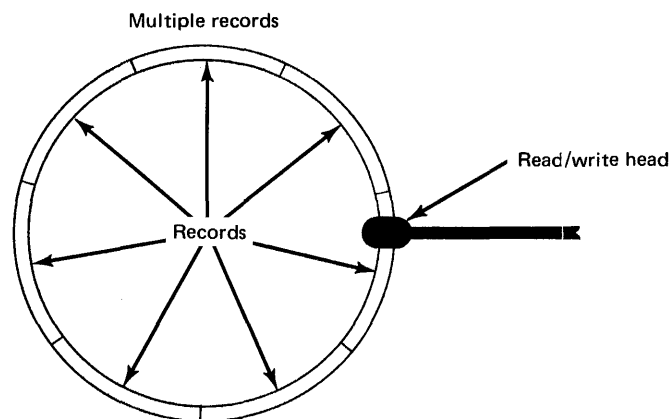
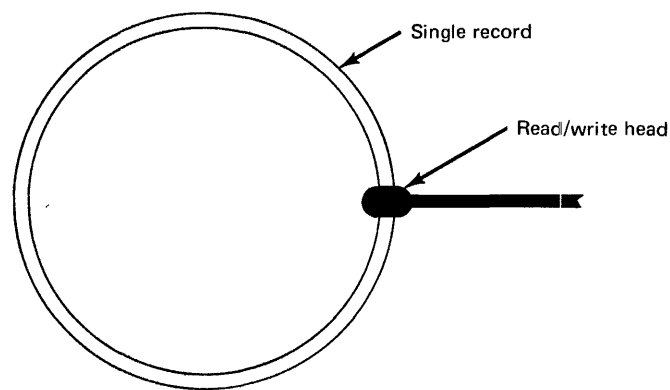
Data files on DASD are usually stored according to cylinder rather than in sequential tracks across a single disk surface. The reason for this is related to speed of operation. It is faster to electronically switch from one read/write head to the next one (in a cylinder) than it is to mechanically move all read/write heads from one cylinder to the next across the disk surface. (All read/write heads move together when a seek to a new cylinder is performed in a disk storage device such as the one described here.)

Two types of seek operations in a disk storage device, then, are:

1. A seek to a new read/write head (new track) in the cylinder at which all read/write heads are currently positioned.
2. A seek to a new cylinder; here, all read/write heads are moved by the access mechanism.

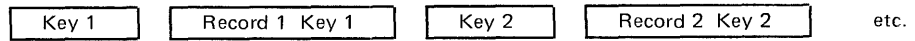
Track Format

One or more records may be on each track:



(Records on a track may be fixed or variable length, blocked or unblocked, depending upon the data file organization scheme used.)

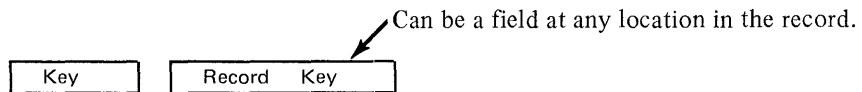
There are many items on a track. The basic functions of DASD can be examined, however, through consideration of only the following:



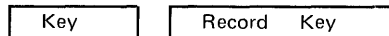
Here, a search of the record keys can be performed to find the records. The search is performed without moving the records to main storage. (Purpose and definition of keys is described on the first page of the descriptive material for Chapter 7.)

Looking for a particular record on a track, when several records are on that track, is called a *search operation*. A search of the keys is called a search-key operation. Before such a search can be performed, the records must be correctly formatted on the track.

During a certain type of write operation, when the records are recorded on the track, the record keys are separated from the records. Before the write operation, the key and record are set up in main storage by the CPU (according to the program instructions that specify such an operation) in the following way:



In other words, the key (which remains in the record in its assigned field) is duplicated just ahead of the record. Then, during the special write operation (write-key-and-data), the record is written on the track as follows:



Because the key is separated from the record, the record can be found later, when needed, by a search operation. The reason for separating the key ahead of the record is related to the fact that the key can be a field at *any* location within the record. The DASD does not automatically recognize keys within the record. It does, however, have the ability to search keys that are formatted ahead of records. Therefore, the key is separated so that this search can be performed by the DASD.

First, the CPU, according to the program being executed, causes the disk to seek to a particular cylinder; a specific track within the cylinder is also specified by the CPU. The search operation is then performed in the disk storage device in the following manner:

1. The CPU requests, according to the program being executed, that the disk storage device perform a search operation.
2. The CPU sends a key (of the record to be found) from main storage to the disk storage device.
3. The disk storage device searches the currently addressed track for the desired key.

If the key is found, the disk storage device signals the CPU. Immediately, because the disk is spinning rapidly and the record will very shortly pass the read/write head, the CPU must request a read-record operation. The record is then sent to main storage and stored into an input area specified by the program.

Various complications may occur, such as when a desired key is not found during the search operation. We do not examine these. The basic point made here is that records on a track can be found *without* reading the entire record, storing it in main storage, and then examining it.

In DASD, therefore, a single record may be located by a combination of *seek* and *search* operations. This is not possible in sequential I/O devices, such as magnetic tape devices.

Note: Even when leading keys are not written on the track ahead of their corresponding records, methods of locating a particular record on a DASD are available. For example, records can be assigned a record-number position on the track. If a specific record is assigned number 3, then it is the third record on the addressed track. We do not, however, examine such record formats in this book.

Sequential File Organization on DASD

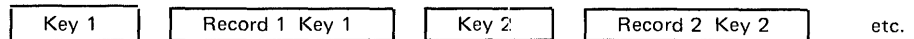
A sequential file organization on DASD is very similar to a sequential file organization on magnetic tape. For example, the first record in a file can be written at the first track in a cylinder. The next record follows this on the same track, and so on until the track is full. Subsequent records are written in sequential tracks down the cylinder. The records are arranged in either ascending or descending sequence according to key. The key may be recorded separately in front of the record on the track. Or, the record may be written without a preceding key. (In either case, the key is also in the record, and the records are arranged in sequence by key.)

Direct access of individual records is not performed if the file is sequentially organized. A particular record can be found by a sequential search operation (if the keys are recorded separately ahead of the records) throughout the entire file.

The file is processed in much the same manner as a file on magnetic tape. A new master file is written as output because (for all add-new-master transaction records) there is no room for the new records between the old records that are already on the tracks.

Indexed Sequential File Organization on DASD

The direct access capabilities of DASD can be used if a data file is organized by the indexed-sequential method. In this type of organization, records are arranged on the medium in ascending sequence by key. The track format is basically:



In addition, an index is constructed. The index contains:

1. The address of each group of records
2. The key of the last record of each group which is the highest key in the group.

The index greatly decreases the number of keys that must be searched to find a particular record. Consider, for example, construction of an index for records with the following disk addresses and record keys:

Address

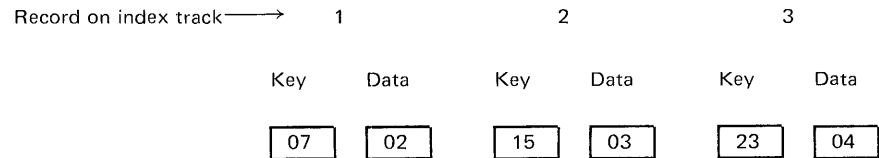
<i>Cylinder</i>	<i>Read/Write Head</i>	<i>Record Key</i>	
1	2	01	
		04	Group of records
		05	
		07	
		07	
1	3	11	
		13	Group of records
		14	
		15	
		15	
1	4	17	
		19	Group of records
		20	
		20	
		23	

All of the preceding records are in cylinder 01, four records are on each track, and the records are arranged in ascending sequence by key. (Only basic index construction is described here. It is not the purpose of this section to present a scheme that handles all conditions that might occur in files using indexed-sequential organization.)

The index for the preceding records is:

<i>Highest Key in Group (by Track)</i>	<i>Beginning of Group Address (Read/Write Head)</i>
07	2
15	3
23	4

This, of course, is a very small index for a very small file. The index is then written on track 1 in cylinder 1 in the following manner:



Now, suppose that a file update is performed and the record to be updated has a key of 13. First, the index track 1 in cylinder 1 is searched. The search is for a key that is equal to or greater than 13. The first key read from the index track is 07, which is less than 13. The search continues. The next key read from the index track is 15, which is greater than 13. A signal is sent from the disk storage device to the CPU indicating this fact, and, consequently, the CPU instructs the disk storage device to read the data, after the key that is greater than 13 which is key 15. The data following this key is then stored into main storage as a result of the read operation at the disk storage device. The data stored into main storage is 03, which is the address of the track on which the record with a key of 13 is stored.

The CPU uses the data 03 and requests the disk storage device to seek to that track. Then another search is performed, again for a key of 13. This time the search is only for an equal condition. When the key of 13 is found, the disk storage device again signals the CPU. The CPU immediately requests a read operation to read the data record (following the key of 13 on the track). As a result, the desired data record is stored in main storage in an input area.

The preceding example contained very few records. But imagine a file of thousands of records. Searching an index and then seeking to the track specified by the index is a much faster operation than a sequential search through the entire file.

Also, several levels of index may be constructed. A data file that uses, say, ten cylinders would have a cylinder index. This is made up of a list of the highest key in each cylinder and the corresponding cylinder address. When a data record is to be found:

1. The cylinder index is searched, and, as a result, a seek to the located cylinder can be performed.
2. Then a track index (in the cylinder) is searched to find (as in the preceding example) the track on which the desired record is located.
3. A seek is then performed to the desired track.
4. Finally, the track is searched until the desired data record key is found. Then the following record, on the track, is read at the disk storage device and stored in main storage.

Overflow Areas

Frequently, new data records must be added to a file. In files organized by the indexed-sequential method, overflow areas are provided for records. Overflow areas are separate tracks, usually in the same cylinder(s) as the main file, set aside for use in holding records when new records are added to the file.

Additional information is provided in the track indexes to indicate when succeeding records are in an overflow area. Such additional information is not described here. At this point, you should note only that overflow areas are provided. When overflow areas become filled with records, the entire file must be reorganized and rewritten. Considerations pertaining to how much room should be provided in overflow areas for a particular data file and how often the file should be reorganized are beyond the scope of this book.

Direct Organization

A direct file organization is constructed by use of some specified mathematical rule of procedure.

The address of a particular record is derived from the key of that record. One way, by no means the only way, is to divide the key by some number and use the remainder as the address of the record. For example, suppose that a file has records with four-digit keys in the range 0001 to 3500. If we use 701 as a divisor, then the address of a record is determined by the following specified mathematical procedure:

1. Divide the record key by 701.
2. Discard the answer and keep the remainder.
3. Use the remainder as the address of the record.

Suppose that a record has a key of 1470. Using the preceding procedure:

1. $\frac{\text{Key} = 1470}{701} = 2$ with a remainder of 68.
2. Discard the answer 2.
3. The remainder 068 is the address of the record, as follows:

<u>Cylinder</u>	<u>Read/Write Head</u>
06	8

There is no need for you to learn the preceding example procedure. Randomizing, as this process is called, involves a number of considerations that are beyond the scope of this book.

After the file is constructed, the same rule of mathematical procedure can be used to find the records. For example, suppose that a change-transaction record is processed to update the record with a key of 1470. The address of that record is derived, by the program being executed in the CPU, in the same manner as above. A seek to cylinder 06 and then read/write head 8 is performed. The record is read from the DASD after its key is found by a search operation of the track. The record is updated according to the program and then written back in its original location. (Direct file organization can be used when the keys are not separately formatted on the track. The record address, however, is still derived from the key. The record can then be accessed according to its position on a track.)

Inline Processing

Processing records when they are received, without sorting or batching them, is called *inline processing*. Here, DASD's are particularly useful because of their direct access capabilities.

Inline processing is used, for example, in airline ticket reservation applications. A reservation-request record is processed by first accessing, from a DASD, the master record. (The key of the master record might be made up of the flight number and flight date.) The master record contains, among other things, data concerning whether the airplane flight in question still has seats available.

Accessing of such records is done frequently during the course of a day. The delays involved if batch processing were used (saving all reservation-request records for processing until the end of the day) would probably be unsatisfactory to most air travelers. A sequential file organization (rather than indexed sequential or direct), in which many master records must be searched in sequence, would also cause many delays in processing the reservation requests.

POST-TEST 9

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. In an indexed-sequential file organization, the index contains:
 - a. A random directory of record ID's
 - b. Identifiers and counters of intermediate records
 - c. The addresses of groups of records and the key of the last record in each group
 - d. Page numbers and symbolic entry names
 - e. I don't know
2. In a DASD, the cylinder is:
 - a. All the tracks across one disk surface
 - b. All of the tracks of all of the disks
 - c. The tracks that are available for reading or writing at each position of the access mechanism
 - d. The number of accesses performed during one movement of the access mechanism
 - e. I don't know
3. A seek operation is performed to:
 - a. Position the access mechanism at some specified cylinder or switch to another read/write head in the same cylinder.
 - b. Find a record in a data file that is organized on DASD by sequentially reading the keys.
 - c. Read a record from DASD and store that record in main storage.
 - d. Locate and identify one of the several records on a track.
 - e. Cycle the access mechanism piston chamber to one-half of the cylinder length.
 - f. I don't know.
4. A data file that is organized by the indexed-sequential method on a DASD contains records that:
 - a. Need not be arranged in sequence because of the addressing capabilities of the DASD
 - b. Must be arranged in ascending sequence by check character
 - c. Can be located from address information in the index when they are arranged in random order on the DASD
 - d. Are arranged in ascending sequence by key
 - e. I don't know
5. Considering that all files are on DASD, match the following two lists:
 - a. Rewrite the file whenever new records are added to the file.
 1. Indexed sequential
 2. Sequential
 - b. Reorganize the file when overflow areas are full and new records must be added to the file.
 - c. I don't know how to match the two lists.

6. In DASD, a direct file organization is one in which the address (or location) of a record is:
 - a. Derived from the value of the key of that record according to some specified mathematical rule of procedure
 - b. Identical to the main storage address where that record will be stored before processings.
 - c. Specified in ascending, collating sequence according to the high-order character of the key of that record
 - d. Determined by the amount of data in the record
 - e. I don't know
7. The operation of finding a specific record key on a track that contains several records, when that track is being read, is called a:
 - a. Seek operation
 - b. Search operation
 - c. Key-sequence access
 - d. Track flagging operation
 - e. I don't know
8. The word "inline" means:
 - a. The data on a DASD is exactly duplicated in main storage.
 - b. Record keys are aligned in ascending sequence throughout a cylinder.
 - c. Input records can be processed as they are received and need not be sorted and/or batched.
 - d. The DASD carries out operations specified by either of the two CPU's to which it is connected.
 - e. I don't know.

Answers are on the next page.

Answers to Post-Test 9

1. c
2. c
3. a
4. d
5. a. 2
b. 1
6. a
7. b
8. c

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 9. FILE ORGANIZATION AND DIRECT ACCESS STORAGE DEVICES (DASD)

1. The disk storage device, which is a DASD (direct access storage device), provides a way of writing data on or reading data from a disk pack. Such writing and reading operations are initiated by instruction execution in the CPU.

A disk pack is:

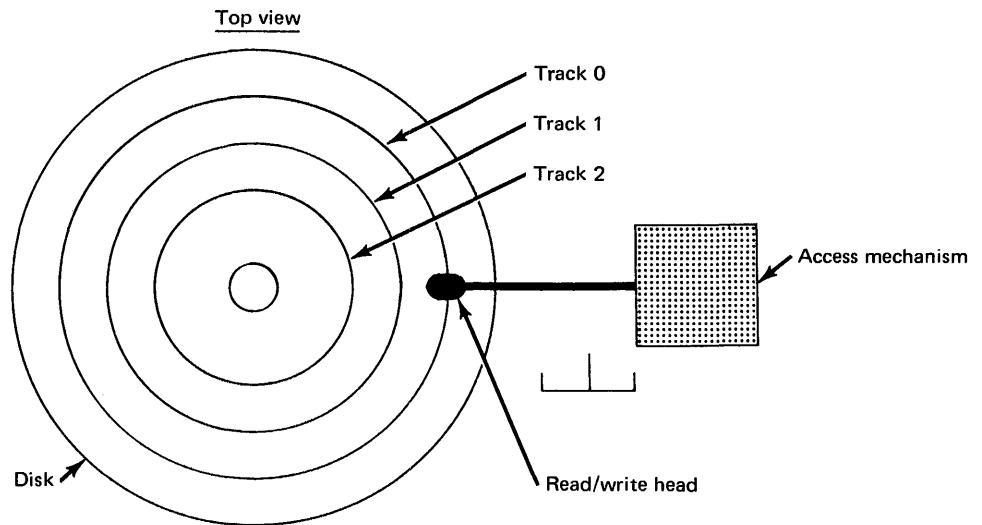
- a. An input medium
- b. An output medium

•••

Both

2. A disk pack is made up of several disks. Data is recorded on a disk surface in tracks. The tracks are circular and are in specific locations on a disk. The specific locations are determined by where a read/write head can be positioned (by an access mechanism) over the disk surface.

The following diagram shows that the access mechanism has positioned the (a) _____ directly over track (b) _____.



•••

- a. Read/write head
- b. 0

3. The access mechanism is designed to position the read/write head:

- a. At any point on the disk surface
- b. Only at specific addressable positions

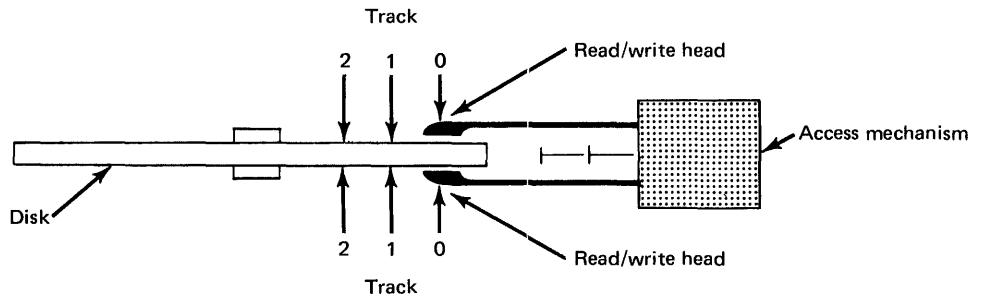
•••

- b. Only at specific addressable positions

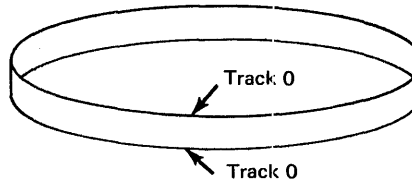
4. The access mechanism moves the read/write head either toward the center of the circular disk or away from the center. Because the disk rotates under the read/write head, data can be recorded in:
 - a. Specific circular paths on the disk surface
 - b. Tracks

• • •

Both
5. When a second read/write head is attached, a side view looks like this:



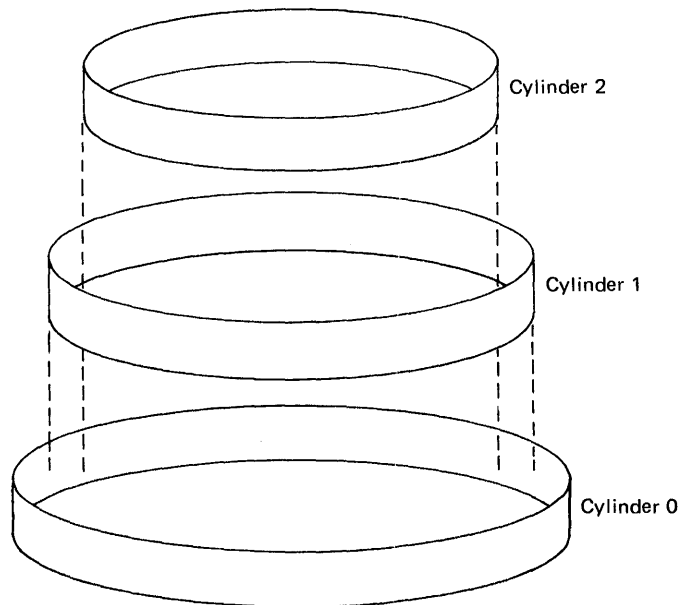
Cylinder 0 looks like this:



The second read/write head is directly under the first in the above diagram. Track 0 on the top surface of the disk and track 0 on the bottom surface of the disk:

- a. Are read by one read/write head
 - b. Make up the top and bottom of a cylinder
- • •
- b. Make up the top and bottom of a cylinder

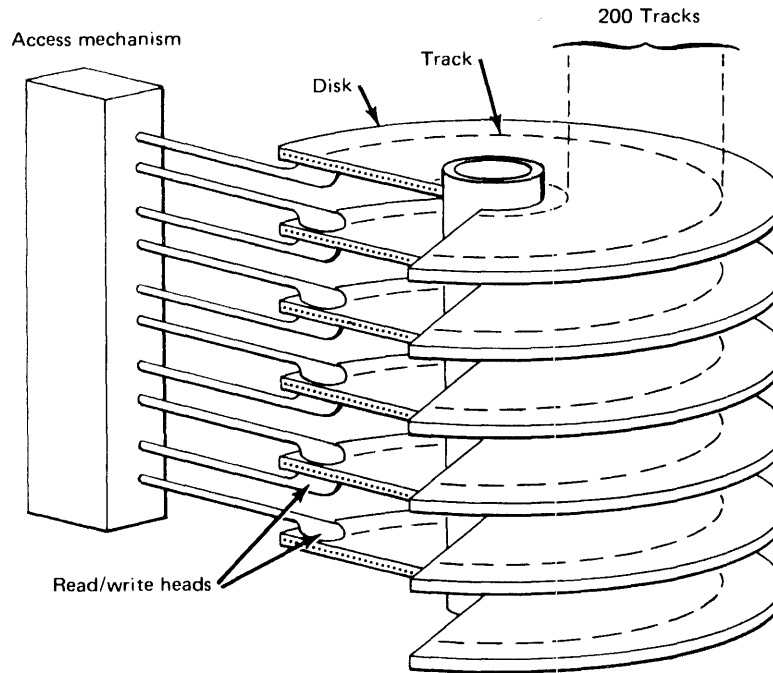
6. By imagining the three cylinders (from our single disk, two read/write head DASD) separated in space we can show how they fit inside of each other:



When the access mechanism moves the read/write heads to any one of these cylinders:

- a. Two tracks are accessed in the cylinder.
 - b. Six tracks are accessed in the cylinder.
 - • •
 - a. Two tracks are accessed in the cylinder.
7. In our single disk, two read/write head DASD there are a total of six tracks. At each position of the access mechanism, two of the six tracks are available for reading or writing data. This total number of available tracks at any one position of the access mechanism is called:
- a. A cylinder
 - b. A disk
 - • •
 - a. A cylinder
8. In our example DASD, the number of tracks available at each position of the access mechanism is called _____.
- • •
 - a cylinder
- 8.1 Just as in our example DASD, it is also true that in *any* DASD, the tracks that are available for reading or writing at each position of the access mechanism is called:
- a. A cylinder
 - b. A read/write head
 - • •
 - a. A cylinder

9. Adding more disks and more read/write heads:



Here, ten read/write heads access ten tracks. The ten tracks make up a cylinder. A cylinder in a DASD, then, is:

- a. All the tracks that are available for reading or writing at each position of the access mechanism
 - b. The areas between tracks
 - • •
 - a. All the tracks that are available for reading or writing at each position of the access mechanism
10. Moving all the read/write heads to another cylinder is called a *seek* (cylinder) operation, which must be initiated by a request from the CPU. In our example device, ten read/write heads are in each cylinder. A seek (cylinder) operation:
- a. Is performed as a result of program instruction execution requesting such an operation
 - b. Involves reading data from or writing data on a specific track
 - • •
 - a. Is performed as a result of program instruction execution requesting such an operation
11. A seek operation can also be performed within the current cylinder. This type of seek is performed to address (access) a specific track and is called:
- a. Seek cylinder
 - b. Seek read/write head
 - • •
 - b. Seek read/write head

12. Switching to a specific track in the current cylinder is called a _____ operation.

• • •

seek or seek read/write head

13. Moving all read/write heads to another cylinder is called a _____ operation.

• • •

seek or seek cylinder

14. Reading data from or writing data on a specific track occurs during:

- a. A seek cylinder operation
- b. A seek read/write head operation

• • •

Neither (Reading or writing occurs during read-record or write-record operations. The point made here is that reading or writing does *not* occur during seek operations.)

15. The important point is that a seek is a preparatory type of operation. It is performed so that a particular place is made available for reading or writing of data, even though no reading or writing of data occurs during the seek operation.

Moving all read/write heads to another position in the disk pack is called:

- a. A seek cylinder operation.
- b. A seek read/write head operation.

• • •

a. A seek cylinder operation. (In some DASD's, both the seek cylinder and the seek read/write head operations may be performed in a single operation. Such machine dependent characteristics are not important at this point.)

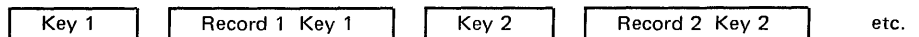
16. A seek operation is performed to:

- a. Position the access mechanism at some specified cylinder or switch to another read/write head in the same cylinder
- b. Locate and identify one of the several records on a track

• • •

a. Position the access mechanism at some specified cylinder or switch to another read/write head in the same cylinder. (A seek operation cannot locate and identify any record because a record is not read during a seek operation.)

17. There are many items, besides the data records, recorded on a track. We consider, however, only the following record format on a track:



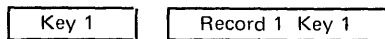
In the preceding diagram, keys:

- a. Are in each record on the track
- b. Are duplicated ahead of each record

• • •

Both

18. The disk-storage device is capable of *searching* for a key (or control field, see the descriptive material in Chapter 7) on a track when the following format is used:



The disk-storage device performs the *search* when requested by the CPU; the CPU sends the desired key to the disk-storage device, and then the disk storage device searches the currently accessed track for a matching key. A signal is sent from the disk-storage device to the CPU when the desired key is located.

Sometimes the search is for a key greater than the one received from the CPU. Here too, a signal is sent to the CPU when the disk-storage device finds such a key on the track being searched.

Assume that the operation is to search for an equal (matching) key or a key that is greater than the one sent from the CPU. The key sent from the CPU is 17. The disk-storage device sends a signal to the CPU when a key of:

- a. 15 is read from the track.
- b. 19 is read from the track.

• • •

- b. 19 is read from the track.

19. The disk-storage device operation of examining keys on a track is called a:
- a. Seek operation
 - b. Search operation

• • •

- b. Search operation

20. Before a search operation to find a specific key on a specific track can be performed:
- a. A seek to that track must be performed.
 - b. All records that are on that track must be read and then stored into main storage.

• • •

- a. A seek to that track must be performed.

21. A seek to a specific read/write head within a cylinder can be performed only after:
- a. A search of that cylinder has been performed.
 - b. A seek cylinder operation (to the specific cylinder) has been performed.

• • •

- b. A seek cylinder operation (to the specific cylinder) has been performed.

22. To find a specific record key on a specific track, first a seek cylinder operation is performed to:

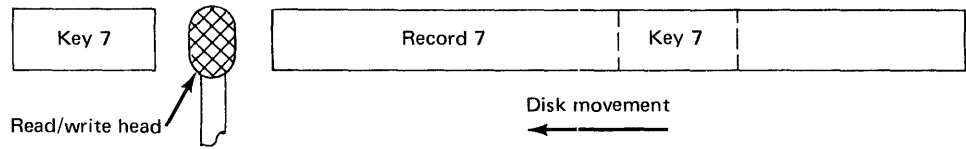
- a. Move all read/write heads to the desired cylinder.
- b. Perform a key search operation throughout the current cylinder.

• • •

- a. Move all read/write heads to the desired cylinder.

23. After the cylinder that contains the desired key (and record) is accessed, a seek read/write head operation is performed to:
- Access the track that contains the desired key.
 - Find the desired key on the specific track.
- • •
- Access the track that contains the desired key.
24. After a seek read/write head operation is performed, the desired track within the current cylinder can be searched to:
- Find the desired key on the track.
 - Move the access mechanism.
- • •
- Find the desired key on the track.
25. A particular record key can be located by a sequence of seek and search operations. Rearrange the following operations into the proper order (what should be done first, second, and third):
- Search key
 - Seek cylinder
 - Seek read/write head
- • •
- Seek cylinder
 - Seek read/write head
 - Search key
26. A seek operation is performed to:
- Find a specific key on a track.
 - Position the access mechanism at some specified cylinder or switch to another read/write head in the same cylinder.
- • •
- Position the access mechanism at some specified cylinder or switch to another read/write head in the same cylinder.
27. A search key operation involves:
- Reading the record keys from an accessed track
 - Switching to another read/write head in the current cylinder
- • •
- Reading the record keys from an accessed track
28. The operation of finding a specific record on a track, while that track is being read, is done:
- By searching the keys on the track until the desired record key is found
 - By seeking to another cylinder
- • •
- By searching the keys on the track until the desired record key is found

29. Assume that the disk-storage device sends a signal to the CPU indicating an equal key. (The key read from the track matches the one received from the CPU.) The physical relationship between the read/write head and the track (at the time the signal is sent to the CPU) can be shown as follows:



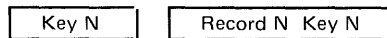
Now, note that read and write record operations are specified separately from search operations by the CPU. In order to cause the data record 7 to be stored into main storage (see preceding diagram):

- a. The CPU must immediately request a read-record operation because the disk is spinning and record 7 shortly passes the read/write head.
- b. The disk storage device must perform a seek operation to the track specified by the key.

• • •

- a. The CPU must immediately request a read-record operation because the disk is spinning and record 7 shortly passes the read/write head.

30. Assume that records are formatted as follows:



and the cylinder and track address of a specific record are available to the program being executed in the CPU. The specific record location can be accessed (in the disk pack) for reading or writing purposes by (arrange the following in the correct order):

- a. A search (key) operation
- b. A seek cylinder operation
- c. A seek read/write head operation

• • •

- b. A seek cylinder operation
- c. A seek read/write head operation
- a. A search (key) operation

31. Records are arranged in ascending sequence by key in the indexed-sequential file organization. You would expect a data file of such records to be sequentially arranged in:

- a. The tracks in a cylinder
- b. The tracks across a disk surface

• • •

- a. The tracks in a cylinder

32. The highest record key on each track of the cylinder and the address of that track (read/write head address) are used to form a *track index*. (Other levels of index may be used, such as a cylinder index, but we consider only the track index here.)

Using the description in this frame and the following information, specify both parts of entry 3 in the following index:

<u>Data File</u>			<u>Index</u>	
<u>Address</u>				
<i>Cylinder</i>	<i>Read/Write Head</i>	<i>Key</i>	<i>Key</i>	<i>(Read/Write Head)</i>
01	05	411		
		412		
		414	414	05
01	06	417		
		419		
		420	420	06
01	07	437		
		441		
		448	_____	_____ ← Entry 3
01	08	450		
		451		
		453	453	08
• • •				
448	07			← Entry 3

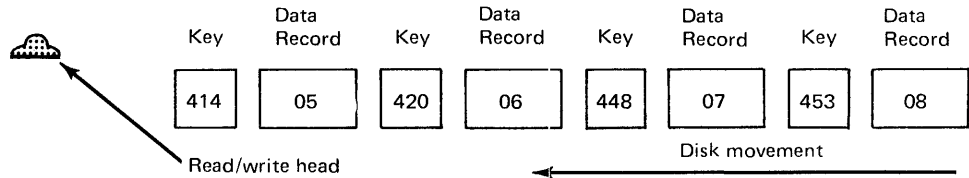
33. The track index, in an indexed-sequential file organization, is made up of:
- The addresses of groups of records
 - The key of the last record of each group

• • •
Both

34. A constructed track index for the file data on the left is shown on the right:

<u>Data File</u>			<u>Index</u>	
<u>Address</u>				
<i>Cylinder</i>	<i>Read/Write Head</i>	<i>Key</i>	<i>Key</i>	<i>(Read/Write Head)</i>
01	05	411		
		412		
		414	414	05
01	06	417		
		419		
		420	420	06
01	07	437		
		441		
		448	448	07
01	08	450		
		451		
		453	453	08

The preceding index is written on a track, say track 04 in cylinder 01, as follows:



To find the record with a key of 441, the CPU first instructs the DASD to search the index track. The search is for a key that is *equal to* or *greater than* 441. Assuming that the search starts at key 414 on the track, reading which key causes the search to be completed?

• • •

448 (Details concerning where on a track a particular disk-storage device starts searching are not important to this description. Such devices can, however, determine that searching is to start at the beginning of the track. For our example, we merely assumed that the search started at key 414.)

35. As soon as the signal is sent by the disk-storage device to the CPU, the CPU requests a read data record operation. The data following key 448 is then sent to main storage. What is the actual data in the data record that follows the key 448 on the index track (see preceding frame)?

• • •

07

36. The data 07 is the address of the track on which the record with a key of 441 is located. The CPU sends this address to the disk-storage device so that track 07 can be accessed. The operation performed to access track 07 is:

- a. A seek
- b. A search

• • •

- a. A seek

37. After the seek to track 07, the disk-storage device can perform a search, requested by the CPU, for a record key of 441. After key 441 is read from the track:

- a. A signal is sent to the CPU from the disk-storage device.
- b. A seek to cylinder 4, read/write head 41 is performed.

• • •

- a. A signal is sent to the CPU from the disk-storage device.

38. As soon as the CPU receives the signal, which indicates that key 441 is found, the CPU, in order to have the data record stored in main storage:

- a. Requests the disk-storage device to search the index again
- b. Initiates a read record request to the disk-storage device

• • •

- b. Initiates a read record request to the disk-storage device

39. In an indexed-sequential file organization:
- a. Data records are arranged in ascending sequence by key.
 - b. The index contains addresses of groups of records and the key of the last record in each group.
- • •
- Both
40. Without an index, a group of sequentially organized records on DASD is processed in a manner similar to a sequentially organized data file on magnetic tape. To find a specific record in a sequentially organized file on DASD, a sequential search of keys can be performed (when the keys are separately formatted on the track):
- a. By starting at the beginning of the file
 - b. By starting at the track address of the desired record
- • •
- a. By starting at the beginning of the file
41. In a file organized by the indexed-sequential method, there is no room for new records between old records that are on the track. Separate overflow areas are set aside when the file is constructed. New records to be added to the data file cause records to be stored in the *overflow area*. Information in the index specifies when records are in the overflow area. (Such information and the way it is used is not described here.)
- In a file organized by the indexed-sequential method, new records:
- a. Cause records to be put in the overflow area(s) as long as there is room in the overflow area(s)
 - b. Require rewriting and reorganizing of the entire file
- • •
- a. Cause records to be put in the overflow area(s) as long as there is room in the overflow area(s)
42. A file organized by the indexed-sequential method must be rewritten and reorganized when new records are to be added to the data file and:
- a. An old master record must be located.
 - b. The overflow area(s) is (are) already full.
- • •
- b. The overflow area(s) is (are) already full.
43. A data file without an index and in which records are arranged in sequence (either ascending or descending) by key is a *sequentially organized file*. This type of file must be rewritten and reorganized:
- a. Whenever new records are added to the file
 - b. When the overflow area is full
- • •
- a. Whenever new records are added to the file

44. Another type of file organization method is the *direct*. Here, the address of each record is derived from the key of that record. A specified mathematical rule of procedure is used to determine the address.

Records in a direct file organization are written on the DASD with their keys separated. In a direct file organization:

- a. The address of each record is derived from that record's key.
- b. A mathematical rule of procedure is used to determine the address of a record.

• • •

Both

45. A DASD contains a data file organized by the direct method. A record is written in the data file at the address derived (by means of some mathematical procedure) from the key of the record. Later, the CPU can request the disk-storage device to read that record. The CPU can do this by:

- a. Using the track index to locate the record
- b. Using the same mathematical procedure to determine the address of the record from its key

• • •

- b. Using the same mathematical procedure to determine the address of the record from its key.

46. Match the two lists:

- | | |
|--|-----------------------|
| a. Contains records arranged in either ascending or descending sequence by key. | 1. Indexed-sequential |
| b. Uses overflow areas when records are added to the file. | 2. Direct |
| c. Record addresses are determined by some mathematical rule of procedure utilizing the key of the record. | 3. Sequential |

• • •

- | | |
|--|-----------------------|
| a. Contains records arranged in either ascending or descending sequence by key. | 3. Sequential |
| b. Uses overflow areas when records are added to the file. | 1. Indexed-sequential |
| c. Record addresses are determined by some mathematical rule of procedure utilizing the key of the record. | 2. Direct |

47. DASD, in which data files are organized in a manner that allows direct-access of individual records, is useful for inline processing. The processing of records as they are received, without first batching and/or sorting them, is called *inline processing*.

Inline processing is not used with sequentially organized data files, such as those on magnetic tape, because:

- a. A record can be found only by sequentially examining each record in the file.
- b. It takes too long to locate each record to be updated in the file.

• • •

Both

Chapter 10. Output Printing; Levels of Control; Auditing

For directions, refer to the summary on the inside of the front cover.

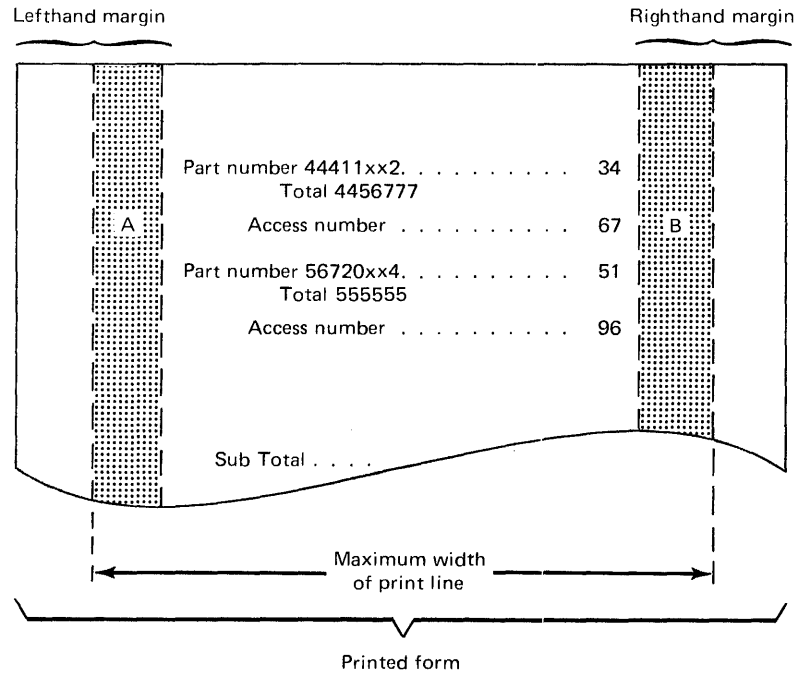
PRETEST 10

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. For a card-to-printer job, header information that is to be printed at the beginning of each page:
 - a. May be written in the source program or may be in header cards that are at the beginning of the input, transaction card file
 - b. Must be preprinted on each page before the job is run
 - c. Must be in every input card, so that whenever a page is started, the header information is available for printing
 - d. Can be printed only in a separate printer job run
 - e. I don't know
2. A printer skip operation, to the beginning of the next form (page), is normally performed:
 - a. Only after the operator presses the end-of-form key
 - b. Each time an input skip transaction record is stored into main storage
 - c. Under program control after the printer indicates that the last line for the current form has been reached
 - d. Only when an input transaction record that contains header information is stored into main storage
 - e. I don't know

3. With reference to the following figure, the shaded areas A and B and the blank spaces within a printed line are all:
- Determined by zeros in the input print record
 - Fixed locations
 - Determined by blanks in the output print record
 - Assigned according to where the operator places the print ribbons
 - Determined by asterisks in the input transaction record
 - I don't know



4. Answer parts I, II, and III of this question with respect to the following:

Job Description

A sales-total printed report is to be produced. The totals to be printed are: each salesman's total (within each office), each office total (within each district), each district total, and a final total of all sales.

- What total(s) should be printed when a change-of-district control break occurs?
 - Only salesman
 - Only salesman and office
 - Only salesman, office, and district
 - I don't know
- What total(s) should be printed after the last input record is read?
 - Only final
 - Salesman, office, district, and final
 - Only district and final
 - I don't know
- Forming the final total from district totals, district totals from office totals, and so on, rather than forming every total from each input record is called:
 - Total update
 - Reverse totaling
 - Rolling totals
 - Branching-record totals
 - I don't know

5. A total that is produced only for checking purposes, and is not part of the normal output data file, is called:
- a. Hybrid total
 - b. Hash total
 - c. Header total
 - d. Hits total
 - e. Hunting total
 - f. I don't know

Answers are on the next page.

Answers to Pretest 10

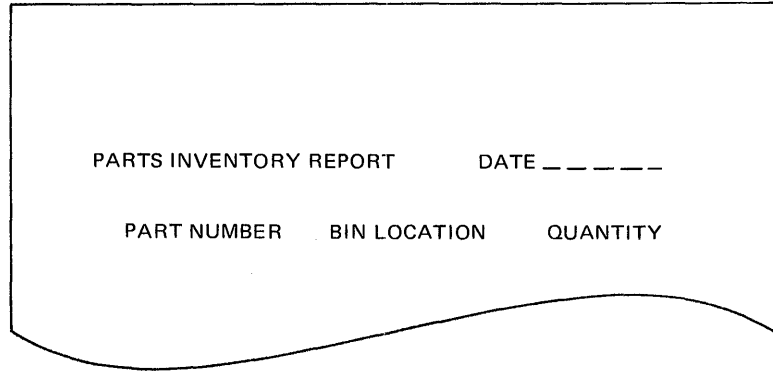
1. a
2. c
3. c
4. I. c
II. b
III. c
5. b

DESCRIPTIVE MATERIAL FOR CHAPTER 10. OUTPUT PRINTING; LEVELS OF CONTROL; AUDITING

Output Printing

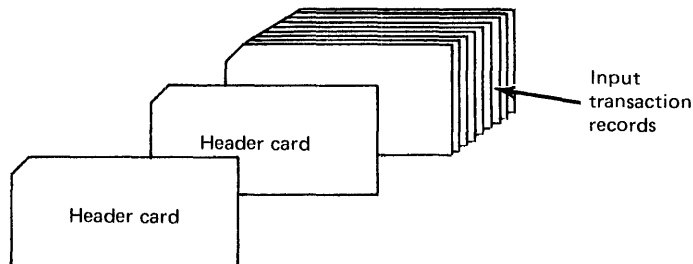
Header Information

Forms, on which output data is to be printed, are sometimes preprinted with header information. ("Preprinted" means that the header information is on the forms before they are put into the output device called a printer.) An example of header information on a printed form is:



On the other hand, header information need not be preprinted but can be printed as output data on blank forms as the job is being run. In this case, the needed header information can be in the input file.

Consider, for example, a card-to-printer parts inventory application. Punched cards contain input data for a parts inventory printed report. Two header cards are at the beginning of the input transaction card deck:



(Note that other jobs may require only one or several header cards. For this example we consider only two header cards.) The header cards for our example contain the following data:

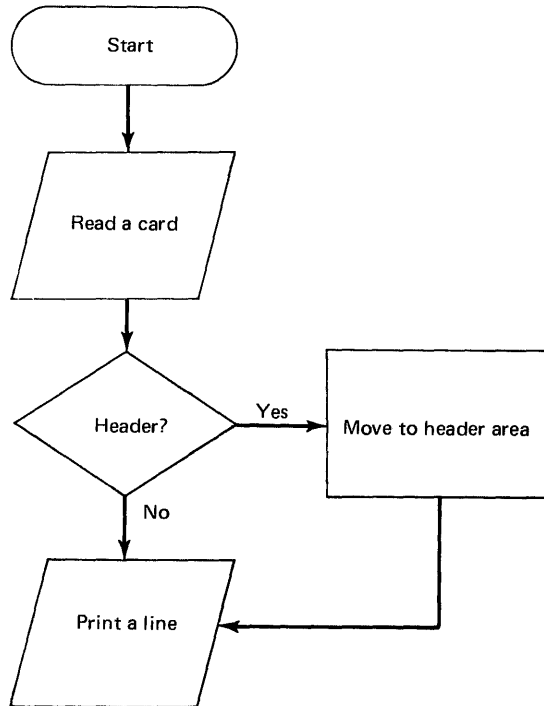
Card 1: PARTS INVENTORY REPORT FEBRUARY 20, 1998

Card 2: PART NUMBER BIN LOCATION QUANTITY

At the start of the job, the header cards are read at the card reader. Identifying information in these cards results in their being handled by the program as header cards. When a header card is read at the card reader, the data from that card is stored in the input area in main storage. Because the data is identified as header information, the program moves that data from the input area to a header area which is also in main storage. The need for such a header area must be established by the programmer when he writes the original source program.

The reason for assigning such a header area is related to the following: First, keep in mind that the job we are considering is a card-to-printer application. After a card is read, appropriate data from that card (according to the requirements of the job) is printed.

Next, recall that an input area (in main storage) must be assigned for the input records. Storing new data into a main storage location results in erasing of the data already in that same location. The header information in the input area, then, is erased when the next input record is stored in the input area. But the header information is to be used for every page of the report. Therefore, the input header information is moved, under program control, from the input area to some other place in main storage (a header area) before the next input record is stored in the input area. The header information can be kept in the header area and used for printing at the start of each new page. In flowchart form, the operation is:



If this procedure is used, only two header cards are needed at the beginning of the input deck. Whenever the header information is subsequently needed for a new page, it is read out of the header area and sent to the printer. (Recall that reading data from a main-storage location does *not* cause the data in that same location to be erased.)

Another way of providing for the availability of header information is to write it as a part of the original source program. (How this is done is related to the actual machines and to the programming language used and is beyond the scope of this book.)

Note, however, that even if the header information is a part of the program, the date must be specified separately (such as in an input header card) every time the job is run.

Planning Print-Record Layouts

Recall that the programmer usually plans input-record and output-record layouts before he produces a flowchart for a particular problem statement. Planning an output-record layout for a printer requires detailed knowledge of the specific characteristics of the printer to be used. We consider here only some basic considerations that are applicable to many printers.

The first two interrelated items to be considered are:

1. How many printing positions are available on the printer that is being used?
2. What is the maximum number of positions needed for the job under consideration?

These questions are important because each output record that is sent to the printer is printed on a single line. It does *not* carry over to the next line. If, therefore, an output record contains more characters than the printer can print on a single line, the “extra” characters are lost. Programmers do not plan to lose output data. Consequently they must specify output records to be smaller than or equal to the maximum number of available print positions in a line.

Note: Most output printers can print any character in each print position. The characters vary on different printers but are composed of alphabetic, numeric, and special characters.

In order to set up output records for a printer, the programmer uses a *printer spacing chart*. Such a chart is designed according to the characteristics of the printer being used. As an example, consider a printer that has a maximum of 45 printing positions per line. We might use the following example printer spacing chart for such a printer:

	Print positions																																																	
Print line	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45					
1																																																		
2																																																		
3																																																		
4																																																		
5																																																		
6																																																		
7																																																		
8																																																		

Note: Most printers have more than 45 positions in a print line. A job that is performed frequently is an “80-80 list.” This job is the printing of the data read from 80-column punched cards, a single line of print for each card. The print line in this case is at least 80 positions wide.

An example of header information laid out on a printer spacing chart is:

Print positions		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45						
Print line	1	P	A	R	T	S	␣	I	N	V	E	N	T	O	R	Y	␣	R	E	P	O	R	T	␣	␣	␣	␣	F	E	B	R	U	A	R	Y	␣	2	0	,	␣	1	9	9	8	␣	␣						
	2																																																			
	3																																																			
	4	␣	␣	P	A	R	T	␣	N	U	M	B	E	R	␣	␣	␣	␣	B	I	N	␣	L	O	C	A	T	I	O	N	␣	␣	␣	␣	Q	U	A	N	T	I	T	Y	␣	␣	␣	␣						
	5																																																			
	6																																																			

Note that the symbol ␣ is used in positions where no printing is desired. ␣ is the symbol for a space or blank. It, just like other data, can be represented in machine readable form, such as punched card code. When a ␣ is sent as part of an output record to a printer, nothing is printed in the corresponding print position in the print line.

The programmer specifies the ␣'s on the spacing chart so that he can more easily plan where blanks are required in the output header records in main storage. Consequently, he can determine how the input header records should be formatted in order to have the data in main storage in the proper way—blanks in blank positions.

The programmer plans, by use of the spacing chart, not only for header but also for other output print data. Because such records result from processing input data, which is different in each input record, these records cannot be laid out character by character on the spacing chart. However, the maximum size of the output fields to be printed is known. Therefore, the spacing chart is used to set up for the locations of such output data. The blanks (␣) that are required for such records can be determined on the spacing chart. For our example, assume that each part number is a five-digit number, each bin location is a four-digit number, and the largest expected quantity can be printed in six positions. The output-record format for our example, as shown under the header information, is:

Print positions		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45							
Print line	1	P	A	R	T	S	␣	I	N	V	E	N	T	O	R	Y	␣	R	E	P	O	R	T	␣	␣	␣	␣	F	E	B	R	U	A	R	Y	␣	2	0	,	␣	1	9	9	8	␣	␣							
	2																																																				
	3																																																				
	4	␣	␣	P	A	R	T	␣	N	U	M	B	E	R	␣	␣	␣	␣	B	I	N	␣	L	O	C	A	T	I	O	N	␣	␣	␣	␣	Q	U	A	N	T	I	T	Y	␣	␣	␣	␣							
	5	␣	␣	␣	␣	␣																																															
	6																																																				

Note: Details concerning how a variable-size print field, such as quantity in the preceding example, is printed without high-order zeros (such as the printing of 975 instead of 000975) are not described in this book.

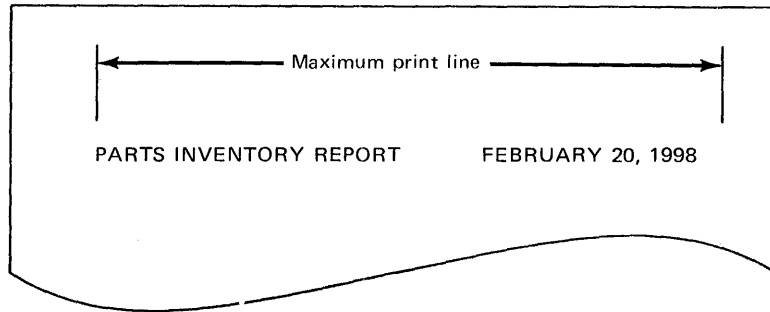
Printer Operation

When the job data is actually processed, an output-record area (in main storage) is set up in the same way as shown on the spacing chart. The output area that is reserved is, in our example printer, 45 main-storage positions long because the maximum print line is 45 positions wide. Reserving 45 main-storage positions and setting them according to the line to be printed assures that no data is lost and extra data is not printed. (Extra data would be printed if characters other than blanks are in the output record in positions that should contain blanks.)

In our example, suppose that the first header line is to be printed on a page at the printer. First, the output-record area in main storage is loaded with the header data (by the program) as follows:

PARTS INVENTORY REPORT FEBRUARY 20, 1998

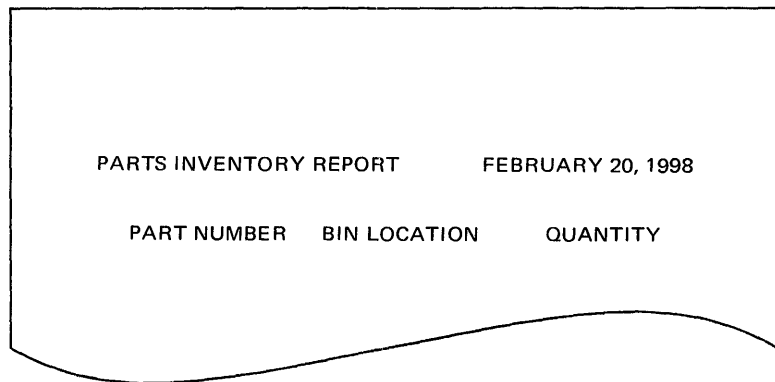
Next, the CPU initiates a print operation (according to a program instruction execution). As a result, the first line is printed on the form as follows:



Now the program initiates, via the CPU, a spacing operation (line spacing) so that the proper number of blank lines follow the first header line. (Some printers can combine the print and the space into a single operation. *How* this is done for any printer is relatively unimportant, however, as far as this description is concerned.)

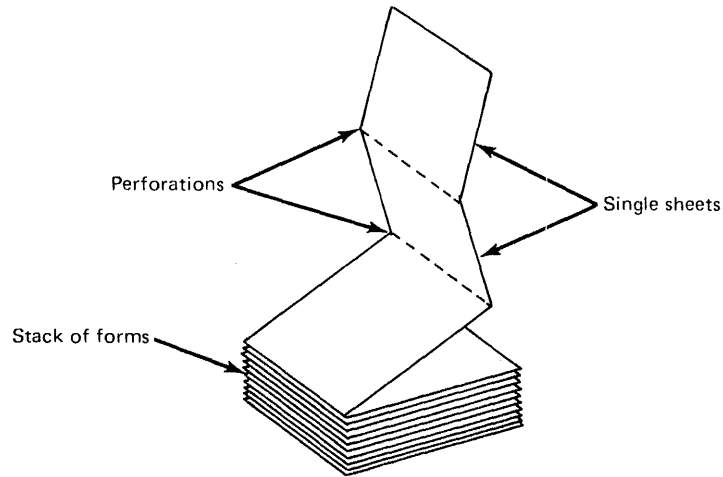
Next, the CPU (according to the requirements of the program) must set up the output area in main storage with the second header line. Moving the second header line data to the output area erases the first header line in the *output area*. Both header lines, however, are still available in the header area in main storage. The header information can be read from the header area any number of times without the header area being erased.

After the second header line data is in the output area, a program instruction executed by the CPU again initiates a print-a-line operation. The second header line is read from the output area and sent to the printer. The printer prints the line, and the page then looks like the following:

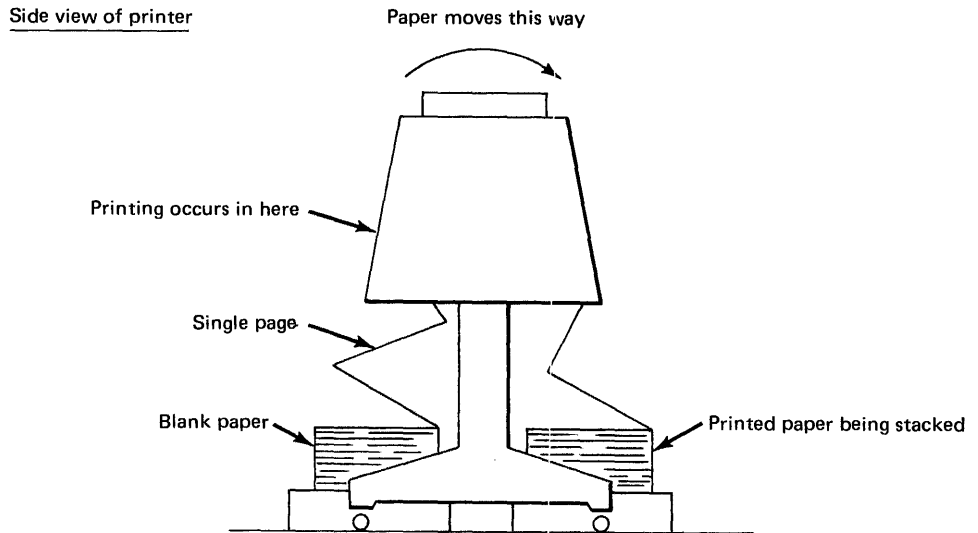


After the header lines are printed, the job continues: A record containing a part number, a bin location, and a quantity is read, processed (according to program requirements) and then moved to the output area. The operation proceeds in a manner similar to the header printing just described. The only basic difference is that the data for each output record is now obtained from processing a transaction input record instead of from the header area.

The forms (or pages or sheets) on which printing occurs are made from a long sheet of paper that has been perforated and then folded into flat sheets:



The paper is fed through the printer (under program control) in the following way:

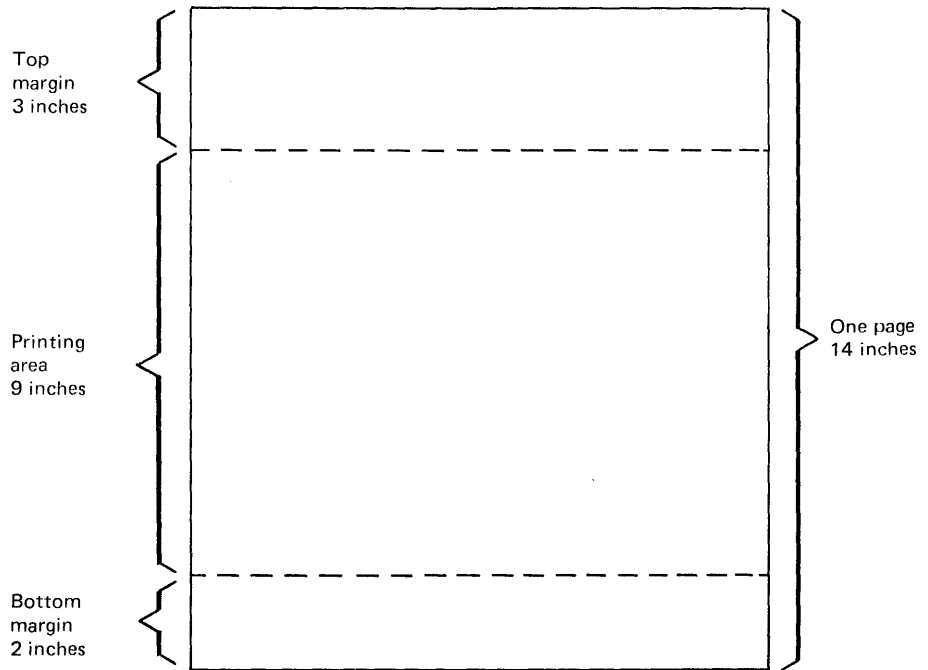


When paper is first loaded into a printer for a job, the operator sets up the first form at the first line to be printed. The procedure is somewhat similar to putting a piece of paper into a typewriter and lining it up at the first line to be printed. But in the printer, the first form is attached to the second, the second to the third, and so on.

Notice that it is *not* desirable to print on the perforations. Also, margins should be at the top and bottom of each page. The program is written so that the top and bottom margins are produced. When the last line on the first form is printed, the printer sends a signal to the CPU. The printer does this because it has been set up to do so after a specified number of lines have been printed. (How this is accomplished for any particular printer is unimportant to this description.)

The number of lines to be printed on a page is determined by the length of the page and the desired size of the margins. Suppose, for example, that the printer prints 8 lines per inch, a top margin of 3 inches and a bottom margin of 2 inches are desired, and each page is 14 inches from top-to-bottom.

On this example form, 9 inches remain for printing:



If the printer prints 8 lines per inch, 72 lines (9 inches times 8 lines per inch) are available for printing on each page. As you have seen, however, not all of these lines need necessarily be printed. Blank lines are between the two header lines in the example described previously.

However, the printer is set up so that when the last desired print-line position on a page is reached, the printer sends the CPU a signal indicating this fact. The CPU, as a result, requests (by executing an instruction) that the printer *skip* to the first print line on the next form. This skip is a distance of 40 lines (at 8 lines per inch):

<i>Bottom Margin</i>	+	<i>Top Margin</i>	=	<i>Skip Distance</i>
2 inches		3 inches		5 inches
16 lines		24 lines		40 lines

At the end of the *skip* operation, the first line (header line) of the next page is in position to be printed.

In summary:

1. Left and right margins and blank spaces within a print line are determined by blank characters in the output-record area in main storage.
2. Header information can be in special records at the beginning of the input file, written into the original source program, or preprinted on the forms.
3. A skip operation to the next page is initiated by a program instruction, which is executed as a result of the printer signalling the CPU that the last print line on a form has been reached.

Levels of Control

Records in a data file are frequently arranged according to groups. For example, in a sales-total job, total dollar-amounts are to be determined for:

1. Each salesman in each office
2. Each office in each district
3. Each district

In addition, a final dollar-total of all sales is to be determined. The "grouping" of records for this job is determined by control fields. Assume the input data file is in punched cards. Each input card contains a record of a single sale, including the dollar-amount of the sale. Each card also contains salesman-number, office-number, and district-number which are control fields.

Notice in the following example records that records are arranged in *ascending* sequence by salesman number within an office, and then office number within a district:

<i>District</i>	<i>Office</i>	<i>Salesman</i>	<i>Sales Amount Field</i>
1	2	4	\$3.00
1	2	4	\$6.00
1	2	6	\$4.00
1	4	6	\$3.00
1	4	7	\$4.00
1	4	7	\$7.00
2	1	2	\$2.00
2	1	2	\$4.00
2	1	2	\$5.00
2	2	4	\$3.00
2	2	5	\$4.00
2	2	9	\$7.00

Note: A data item in a control field is a control number. Control numbers allow the program to recognize specific control number changes.

Assume that the program, for this example, is written to print the totals for the previous records as follows:

Salesman 4	\$9.00			
Salesman 6	\$4.00			
		Office 2	\$13.00	
Salesman 6	\$3.00			
Salesman 7	\$11.00			
		Office 4	\$14.00	
			District 1	\$27.00
Salesman 2	\$11.00			
		Office 1	\$11.00	
Salesman 4	\$3.00			
Salesman 5	\$4.00			
Salesman 9	\$7.00			
		Office 2	\$14.00	
			District 2	\$25.00
				Final total \$52.00

The program examines each input record (after the record is in main storage) to determine when a control break occurs. A *control break* is a change from one control number to another, for example, from district 1 to district 2.

The program must check separate control fields in order to determine when a control break occurs. The important thing to notice, however, is which field is checked first. The rule is that the largest group is checked first, then the next largest, and so on until the smallest group is checked last. For example, a district (group of offices) is larger than an office (group of salesmen) such checking determines the order in which totals are formed by the program. If the proper order of checking control fields is not followed, incorrect totals are produced.

For example, consider that the salesman number is incorrectly checked by the program *before* the office number. We will use the following example data:

<i>Office</i>	<i>Salesman</i>	<i>Sales Amount</i>
2	4	\$3.00
2	4	\$6.00
2	6	\$4.00
4	6	\$3.00
4	7	\$4.00
4	7	\$7.00

The first change in salesman number is from 4 to 6 (salesman 6 in office 2, who is different than salesman 6 in office 4). The program checks this change and then prints the total:

Salesman 4 \$9.00

Next the program checks the office number (for salesman 6) which is still 2. No change in office has occurred, so the total for salesman 6 (office 2) is started.

A second card for salesman 6 is read. Notice, however, that this is salesman 6 in office 4. But the program (which is incorrectly written) then adds the amount for salesman 6 from office 4 to the amount for salesman 6 from office 2. It does this because it checks for change of salesman number *before* it checks that a new office number applies.

Consequently, a total of \$7.00 is formed (for salesman 6 from office 2) which, if you look at the example data is an incorrect total. Salesman 6 from office 2 should be credited only with \$4.00.

This type of error would not have occurred if the office number had been checked *before* the salesman number.

If control field checking is done properly, each control break results in printing of all lower level totals. For example, a change of district should cause printing of salesman total, office total, and district total.

If the district number does not change, but the office number does, a salesman total and then an office total should be printed.

The largest group of records, of course, is the entire input file. Therefore, after the last input record is read, all of the totals should be printed: salesman, office, district, and final.

Auditing

Procedures for checking and verifying output data (such as salesman totals, office totals, etc.) are used in most data processing installations. This type of procedure is called auditing.

For example, as each original sales record is received from a salesman, the sales amount from that record is totaled (to form an audit-check total) by the person receiving the original sales records. Later when the sales-total job is run, the final total, printed at the end of the job, is checked to see if it is equal to the audit-check total previously determined from each original record.

The method of forming the totals as the job is being run is as follows:

1. When a salesman's total is printed, that total is added to the total for his office.
2. When an office total is printed, that total is added to the district total.
3. When a district total is printed, that total is added to the final total.

In each case, the higher-level total is a result of adding lower-level totals. (This procedure is different than adding each salesman amount to every total as each input record is read.) The procedure of forming totals from previous totals (or totals of smaller groups) is called *rolling totals*. It can be shown conceptually as follows:

Print salesman total

—————→ Add salesman total to office total

Print next salesman total

—————→ Add salesman total to office total

Print office total

—————→ Add office total to district total.

And so on.

If the final total is not equal to the audit-check total, then all district totals can be added to see if they add up to the final total. If they do, something has gone wrong, during the job run, only with formation of the final total.

If, on the other hand, the district totals do not equal the audit-check total, then the office totals are added together. If this total equals the audit-check total, then only the district totals are in error.

Sometimes it is desirable to perform some sort of checking on an item that is not really output data. For example, suppose that a parts inventory job is run where some new part-number records are to be added to the master file.

Totals of the part-numbers are formed in this example, for checking purposes. Such totals are called *hash totals* and are usually not printed as output data.

The example data is:

<i>Old Master Part Numbers</i>	<i>New Part Numbers</i>
04	05
07	08
11	+12
+15	
37 (hash total)	25 (hash total)

The hash totals mean nothing except as they are used for checking. Here these two hash totals should equal the sum of the part numbers in the new master file, produced as a result of adding the new part-number records.

New Master File Part Numbers

04		
05		
07		
08		
11	Old Master	New Part
12	Part Numbers	Numbers
+15	<u>Hash Total</u>	<u>Hash Total</u>
62 (hash total) =	37	+ 25

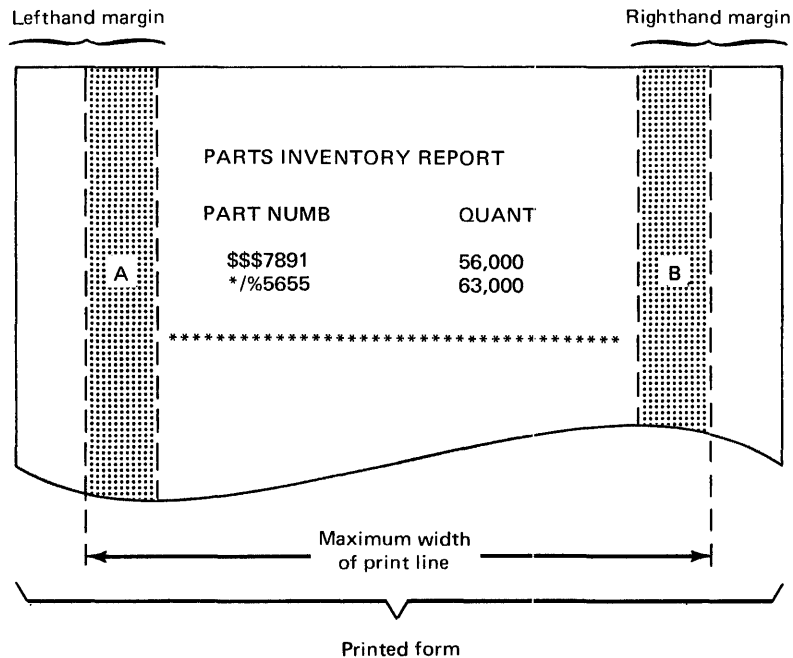
If the new master hash total does not agree with the sum of the hash totals from the old-master and the new record files, then a record was lost or added during the job run. In any case, such checking is an internal program operation, and normally hash totals are not printed. If a hash-total error is detected, however, the program is written so that the operator is notified and corrective measures are taken. What corrective measures are taken is dependent upon the job and the operating procedures at the data processing installation.

POST-TEST 10

Please write your answers on a separate sheet of paper. Please do not guess. Specify the "I don't know" answer when appropriate.

Questions

1. Header information that is to be printed at the beginning of each page, during a card-to-printer job:
 - a. Must be preprinted on each page before the job is run
 - b. Can be printed only in a separate printer job run
 - c. May be written in the source program or may be in header cards that are at the beginning of the input, transaction card file
 - d. Must be in every input card, so that whenever a page is started, the header information is available for printing
 - e. I don't know
2. Normally, a printer skip operation, to the beginning of the next page, is performed:
 - a. Under program control after the printer indicates that the last line for the current form has been reached
 - b. Each time an input skip transaction record is stored into main storage
 - c. Only when an input transaction record that contains header information is stored into main storage
 - d. Only after the operator presses the end-of-form key
 - e. I don't know
3. The shaded areas A and B (see the following figure) and the blank spaces within a printed line are all:
 - a. Determined by zeros in the input print record
 - b. Determined by asterisks in the input transaction records
 - c. Fixed locations
 - d. Determined by blank characters in the output print record
 - e. Assigned according to where the operator places the print ribbons
 - f. I don't know



4. Answer parts I, II, and III of this question with respect to the following:

Job Description

A sales-total printed report is to be produced. The totals to be printed are: each salesman's total (within each office), each office total (within each district), each district total, and a final total of all sales.

- I. What total(s) should be printed when a change-of-district control break occurs?
 - a. Only salesman, office, and district
 - b. Only salesman
 - c. Only salesman and office
 - d. I don't know
 - II. What total(s) should be printed after the last input record is read?
 - a. Only district and final
 - b. Only final
 - c. Sales, office, district, and final
 - d. I don't know
 - III. Forming the final total from district totals, district totals from office totals, and so on, rather than forming every total from each input record is called:
 - a. Rolling totals
 - b. Branching-record totals
 - c. Reverse totaling
 - d. Total update
 - e. I don't know
5. A total that is produced only for checking purposes, and is not part of the normal output data file, is called:
- a. Header total
 - b. Hunting total
 - c. Hybrid total
 - d. Hash total
 - e. Hits total
 - f. I don't know

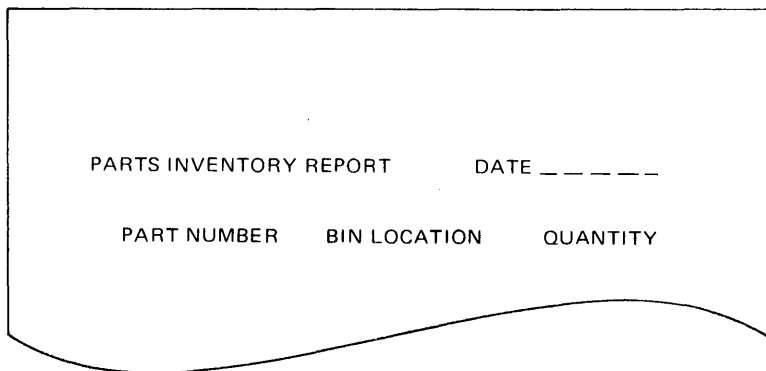
Answers are on the next page.

Answers to Post-Test 10

1. c
2. a
3. d
4. I. a
II. c
III. a
5. d

PROGRAMMED INSTRUCTION SEQUENCE FOR CHAPTER 10. OUTPUT PRINTING; LEVELS OF CONTROL; AUDITING

1. The header information shown in the following figure can be either printed on the page as the job is being run, or preprinted on the page before that page is put into an output printer.



Header data that is printed on a page as the job is being run must be:

- a. Sent to the printer as output data
- b. Available in main storage

• • •

Both

2. Header data can be stored in main storage, under program control, when that header data:

- a. Is in the input data file
- b. Has been preprinted on each page

• • •

- a. Is in the input data file

3. Header information may be in input cards in the input data file. Or, header data may be specified by the programmer when he writes the program. Header information to be printed while the job is being run:

- a. May be read as input data
- b. May be written as part of the source program

• • •

Both

4. Program instructions and header data in the source program are compiled to produce:

- a. Punched cards to be used in the job input data file
- b. Program instructions and header data in the object program

• • •

- b. Program instructions and header data in the object program

5. Header data can be in main storage before the reading of the job input data file is started when:
 - a. The header data is part of the object program.
 - b. Each job input transaction record contains header data.

• • •

 - a. The header data is part of the object program.
6. When header data is printed while a job is being run, that header data is kept in a header area in main storage. On the other hand, a record to be sent to a printer must first be set up in an output area (in main storage). Before header data can be printed, the CPU (by means of program-instruction execution):
 - a. Moves the data from the header area to the output area
 - b. Stores blanks in the header area in main storage

• • •

 - a. Moves the data from the header area to the output area
7. After header data is in a header area in main storage, no other data is stored into that same location during execution of the program. (Recall that storing new data into a main-storage location first erases the data in that location. The data in a main-storage location is undisturbed, however, during a read operation to read data from that location.)

Data in a header area in main storage:

 - a. Is erased each time a read operation occurs for that header area
 - b. Must be renewed after each header line is printed. It is renewed by reading duplicate header records from the input data file.

• • •

Neither
8. Note that when we speak of a header area and an output area in main storage, reference is to a place reserved according to the requirements specified in the source program. In some data processing systems, specific addresses in main storage must be used for certain areas, such as an output area. Usually, however, the locations of such areas are determined according to specification in the source program, and how the compiler assigns such areas during compilation.

No response required; go on to the next frame.
9. Because a separate header area is provided in main storage:
 - a. Only one set of header records is needed at the beginning of the input data file.
 - b. Duplicate header records must be in the input data file. Then, each time a header line is to be printed, a new header record can be read from the input data file.

• • •

 - a. (Only *one* set is required, because as soon as that data is read and then stored into the header area, it is available in the header area as long as the job is being run.)

10. A printer spacing chart is used to plan the format of data to be printed. Generally, each output record that is sent to a printer is the same size as the maximum print line in that printer. If the print line in a printer contains 100 character positions, then each output record sent to that printer should be:
- 90 characters long
 - 132 characters long

• • •

Neither

11. Margins and other blank spaces in a printed line (such as the blank spaces between words) are specified by blank characters in the output record in main storage that is sent to the printer. Assume that a printer has 20 print positions in each line and that the record to be printed is:

OUTPUT RECORD

Which positions of this record (shown below on the spacing chart) should contain blanks (␣)?

Portion of a printer spacing chart

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
O	U	T	P	U	T		R	E	C	O	R	D								

← Print positions

• • •

Position 7 (a blank between words) and positions 14 through 20.

12. After a printer spacing chart has been used to arrange header data and blanks in an output print record, that same chart can be used to determine the layout of data in:
- The header area in main storage
 - The input header records, when they are used

• • •

Both

13. The layout of blanks in output records other than header records can also be specified by use of the printer spacing chart. Here, however, because each output record is the result of processing a different input record:

- Actual data can be specified on the printer spacing chart.
- The programmer can safely assume that all output records will be identical.

• • •

Neither

14. Each paper page that is fed through a printer is attached to the next page. Perforations between pages provide for easy separation of the pages after the job is completed. Printing, of course, should not be done on the perforations.

Relatively short distances may separate printed lines on a page. For short line separations (such as one, two, or three lines) a *space-line* operation is used. Some printers can combine a print-and-space operation (by printing a line and then spacing the number of lines specified by the program).

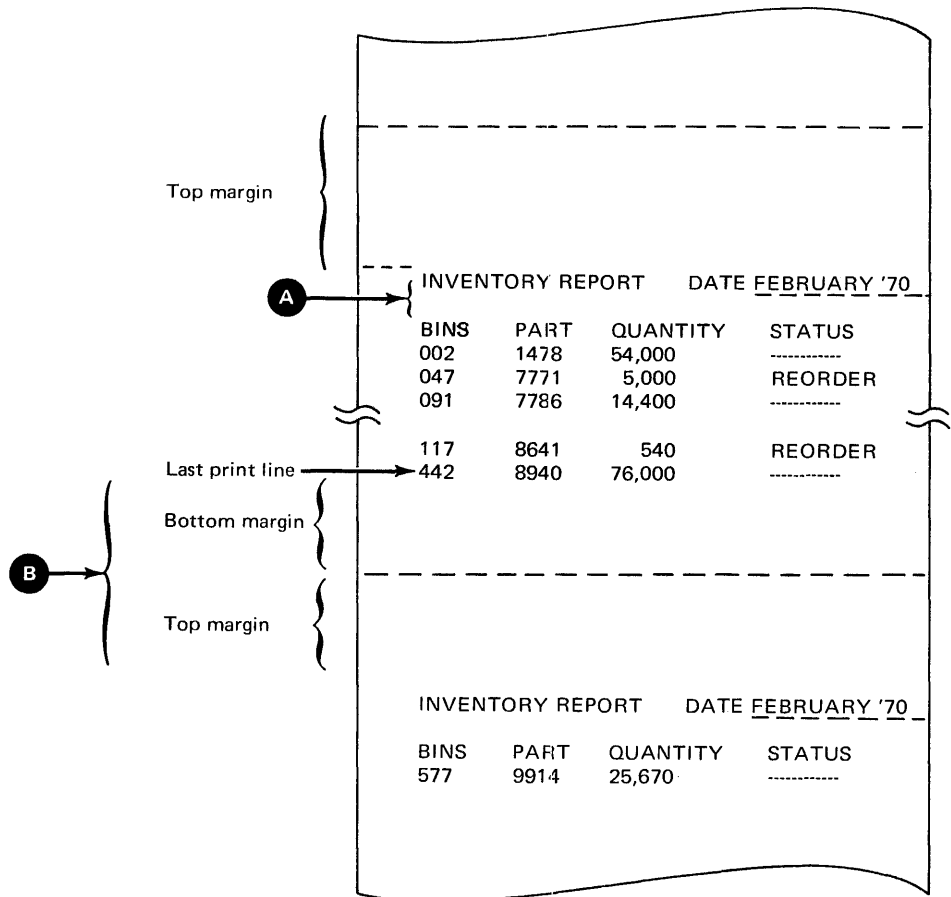
For longer distances, a *skip* operation is usually used. Skipping (when it is an available function in the printer being used) usually moves the paper faster than a spacing operation and is therefore preferred for long-distance, paper-moving operations.

Match the letter (that shows an area in the following figure) to the operation:

Operation

Space

Skip



• • •

A. Space

B. Skip

15. When the operator sets up a printer for a job run, he lines up the first page so that the first line is in position for printing. When the *last* line on the first page is reached, the printer signals the CPU of this fact. (How the lines on a page are counted, and how the CPU is signalled is dependent upon the specific printer being used. Such machine-dependent information is not described here.) After the printer signals the CPU, an instruction in the program requests the printer to move the paper to the first print line on the next page. Because the paper is moved a relatively large distance, the operation is called a:
- Skip
 - Space
- • •
- Skip
16. A skip operation is initiated by program-instruction execution as a result of:
- A header card being read in the input file signalling that it is time for the next page
 - A signal (to the CPU from the printer) that indicates that a specific print-line position (such as the last line on a page) has been reached
- • •
- A signal (to the CPU from the printer) that indicates that a specific print-line position (such as the last line on a page) has been reached
17. Recall that the program identifies records by the data item in their control field. Records may contain more than one control field. The following sales records, for example, contain three control fields:

Control Fields

<i>District Number</i>	<i>Office Number</i>	<i>Salesman Number</i>	<i>Amount</i>
3	8	1	\$500.00
3	8	1	\$400.00
3	8	2	\$600.00
3	9	2	\$400.00
3	9	3	\$200.00
3	9	4	\$300.00
4	3	1	\$900.00
4	3	2	\$300.00

There are three distinct groups in these records: salesman, office, and district. Notice that all of these records are arranged in ascending sequence according to control field within a group, such as:

- By salesman number within each office
- By district number within each office

• • •

- By salesman number within each office

18. Because members of each group are arranged in ascending sequence within the next higher group, sales-amount dollar totals can be determined and then printed:
- For each salesman (in ascending sequence by salesman number) within each office
 - For each office (in ascending sequence by office number) within each district

• • •

Both

19. A portion of the example data is:

Control Fields

<i>District Number</i>	<i>Office Number</i>	<i>Salesman Number</i>	<i>Amount</i>
3	8	1	\$500.00
3	8	1	\$400.00
3	8	2	\$600.00
3	9	2	\$400.00
3	9	3	\$200.00
3	9	4	\$300.00

Notice in this data that there are two salesman with the number 2: one of them is in office 8, the other is in office 9.

The sales amount total that should be credited to salesman 2 in office 8 is:

- a. \$1000.00
- b. \$600.00

• • •

- b. \$600.00

20. In order to total only \$600.00 for salesman 2 in office 8 (see preceding frame) the program should:
- a. Add each new sales amount to the current salesman's total when salesman number does not change.
 - b. Prevent addition of a sales amount to the current salesman's total when a change of office number occurs.
- • •
- b. Prevent addition of a sales amount to the current salesman's total when a change of office number occurs.
21. A change of control field number from one record to the next record is called a control break. A check for a possible control break in a higher-level (or larger) group, such as office, should be performed before a check for a change in a lower-level group, such as salesman. Checking for control breaks in this manner:
- a. Prevents incorrect totals from being formed
 - b. Must be planned by the programmer when he produces a flowchart and when he writes a source program

• • •

Both

22. When a control break occurs for a specific group, totals are printed for that group and for all lower-level groups. The lowest-level group total, however, is formed and printed first. These lowest-level group totals are the sums of individual amounts in the input records (individual sales amounts for a salesman, in the example we have been considering). The next higher level (office, in our example) total is then formed by adding salesman totals. Consider, for example, the following data:

<i>Office Number</i>	<i>Salesman Number</i>	<i>Amount</i>
8	1	\$500.00
8	1	\$400.00
8	2	\$600.00
9	2	\$400.00
9	3	\$200.00
9	4	\$300.00

In this data, the total for office 8 is determined by:

- Adding the total for salesman 1 and the total for salesman 2 (both in office 8) together
- Adding each input record sales amount to the office total until a control break for change-of-office occurs

• • •

- Adding the total for salesman 1 and the total for salesman 2 (both in office 8) together. (Lower-level group totals are added to get the next higher-level group totals. Individual input-record amounts are added together to obtain totals for the lowest-level group *only*.)

23. The procedure of producing higher-level group totals from lower-level totals is called *rolling totals*. "Rolling" the totals facilitates checking (or auditing) procedures. If, for example, *all* district totals do not add up to equal the sum of all office totals, then an error has occurred in the forming of the totals. This type of error is not the same as an error that might occur during reading of the input records.

Usually, some audit-check total is developed independently from the actual job. An example here is the manual totaling of the amounts in original documents that are received by a record-receiving clerk.

The audit-check total can be used to check the final total of the sales-total job run. If the audit-check and final-job totals are not the same value, the next lower-level (district totals in this example) totals can be manually added together and then checked against the audit-check total. (In most cases, the final total would equal the audit-check total. Further checking, then, would not have to be done.)

If the sum of all the district totals then *does* equal the audit-check total, an error was made only in forming the *final* total during the job run. Usually there is no need to go back and total each original input record because:

- A rolling totals procedure is used during the job run.
- Higher-level totals are formed from lower-level totals instead of from data in each input record.

• • •

Both

24. Notice also that when a control break occurs, the totals for that control level and for all lower-levels, are printed. In the sales-total job, a control-break in the lowest-level group (salesman) results in printing that salesman's total (the current salesman when the control break occurs) and adding his total to the current office total. With reference to the sales-total job, then, match the two lists:

Control Break

Totals Printed

1. Office
2. District

- a. Salesman, office, district
- b. Salesman, office

• • •

1. b. Salesman, office
2. a. Salesman, office, district

25. District is a higher control level (in the sales-total job) than either office or salesman. When a control break such as for a new district occurs, the totals for:
- a. That level and all lower levels are printed
 - b. That level and all higher levels are printed

• • •

- a. That level and all lower levels are printed

26. List the total(s) that should be printed (in the sales total job) when a change of district occurs.

• • •

Salesman
Office
District (the highest level and all lower levels)

27. List the total(s) that should be printed (in the sales-total job) when a change of salesman occurs.

• • •

Salesman (there is no lower level of control)

28. List all the total(s) that should be printed (in the sales-total job) after the last input record is processed.

• • •

Salesman
Office
District
Final (the highest level of control)

29. The largest group in our sales-total job is the entire input file of all sales records. After the last record is read and processed, a final total should be printed. Reading of the last card should cause printing of:

- a. A final total only
- b. The current salesman's total, the current office total, the current district total, and a final total

• • •

- b. The current salesman's total, the current office total, the current district total, and a final total

30. A total that is formed for checking purposes only and is not normally printed is called a *hash total*. For example, adding all employee numbers together can form a hash total. If a new employee is hired, his number can be added to the total. Note that this total has no value other than that it is the total of all the employee numbers.

Then, during a payroll job, the sum of the employee numbers for which payroll checks are printed can be checked against the hash total for the employees whose checks *should* be printed. A discrepancy indicates that either an extra check(s) has been printed, or a check(s) has not been printed.

Which of the following would you consider to be a hash total?

- a. The total of all part numbers manufactured by a company
- b. The total of the number of parts in a bin location in a parts warehouse.

• • •

- a. (The total number of parts in a bin location is a useful total that is not limited to checking purposes.)



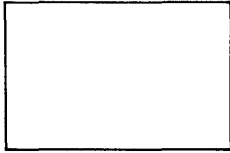
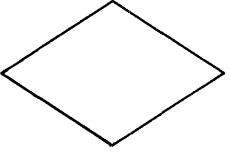

31. Match the two lists:

- | | |
|-------------------|--|
| 1. Hash totals | a. Forming a higher-level group total from the sum of the totals of a lower-level group. |
| 2. Rolling totals | b. Forming a total (or totals) that is normally useful only for checking purposes. |

• • •

- | | |
|----|--|
| 1. | b. Forming a total (or totals) that is normally useful only for checking purposes. |
| 2. | a. Forming a higher-level group total from the sum of the totals of a lower-level group. |

Flowcharting symbols

Symbol	Name	Use
	Terminal	Any start or stop point in the problem
	Input/Output	Any function of an input or output device
	Processing	Operation(s) to be performed
	Decision	A choice is to be made from two or more options
	Flowline	Shows direction



**International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)**

**IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)**

ZZ34-0001-0

Data Processing Systems Basic Concepts Printed in USA ZZ34-0001-0