

**SAMRAT ASHOK TECHNOLOGICAL INSTITUTE, VIDISHA (M. P.)**  
( COLLEGE OF ENGINEERING )

**PROGRAMMING PROJECTS**  
WITH  
**BASIC FORTRAN AND COBOL**



By  
**Rajeev Prakash Shrivastava**

FOR  
THE DEGREE OF  
**MASTER OF SCIENCE**  
( ENGG. FACULTY )

APPLIED MATHEMATICS  
SPECILISATION IN  
COMPUTER PROGRAMMING

OF  
THE BHOPAL UNIVERSITY  
BHOPAL

JUNE 1987, INDIA.

Smarat Ashok Technological Institute (Degree) Vidisha

Department of Applied Mathematics

This work has been carried out by me at Govt. Computer Center, Bhopal and S.A.T.I. Computer Centre, Vidisha (M.P.)

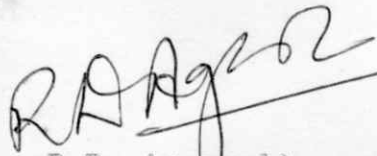
Dated.....1987

My sincere thanks are due to Prof. H.N. Silakari, Principal, S.A.T.I. for providing me with necessary facilities, help and encouragement.

C E R T I F I C A T E

This is to certify that Mr. Rajeev Prakash Shrivastava Student of M.Sc.(Applied) Mathematics and Computer Programming (Engg Faculty) has worked under the supervision of Dr. R.D. Agrawal during the session 1986-87 & accompanying Project in languages Basic, COBOL, FORTRAN, Programs attached are his original work.

I would like to thank Prof. H.N. Silakari, Principal, S.A.T.I. Computer Centre, Mr. Harank Shrivastava, Programmer, Vijay Chandra, operator, S.A.T.I. Computer Centre, for their whole hearted cooperation and constructive criticism in running this event.



(Dr. R.D. Agrawal)  
Prof. & Head of the  
Applied Mathematics  
Department.



(Prof. H.N. Silakari)  
Principal

RAJEEV PRAKASH SHRIVASTAVA

## ACKNOWLEDGEMENT

This work has been carried out by me at Govt. Computer Center, Bhopal and S.A.T.I. Computer Centre, Vidisha (M.P.)

### 1. INTRODUCTION

My sincere thanks are due to Prof. H.N. Silakari, Principal, S.A.T.I. for providing me with necessary facilities, help and encouragement.

I would like to thank Dr. R.D. Agrwal H.O.D. Maths & Computer Application Deptt. for his valuable guidance, it would not have been possible for me to accomplish this work.

I greatly thank Dr. D. P. Shukla, Reader, Dr. R. C. Jain, Dr. D. P. Goyal and Prof. Rajendra Dubey, for their valuable suggestions. I would like to thank Prof. S. N. Agrwal H.O.D. S.A.T.I. Computer Centre, Mr. Mayank Kanoongo, programmer, Mr. Vijay Sharma, operator, S.A.T.I. Computer Centre for their whole heartiest cooperation and constructive criticism in running the event.



May 1987, Vidisha

(RAJEEV PRAKASH SHRIVASTAVA)

# I N D E X

	<b>page</b>
1. INTRODUCTION	1-20
2. PROGRAMMING IN COBOL	21-50
3. PROGRAMMING IN FORTRAN	51-72
4. PROGRAMMING IN BASIC	73-117





## INTRODUCTION

### THE COMPUTER:

The modern computer is basically a machine capable of doing the following three things.

1. It can perform all the four basic arithmetic operation viz, addition, subtraction, multiplication and division all with a fantastic speed.
2. It can follow a list of instructions viz, what to read, where to write, what to subtract from what etc.,
3. It can read/write data from/ on punched cards, magnetic tape, magnetic disk etc. Also it can remember this information.

### IMPORTANCE OF COMPUTERS

Almost a century ago a spate of inventions ushered in the first Industrial Revolution. Within a short span of time many countries became industrialized. Now, we are in the beginning of another Industrial Revolution. The major cause of the second Industrial Revolution is the invention of Computers. Man has invented many electronic devices but the Computer has made a greater impact on society than any other single device. They have made a potentially significant contribution to the society during the last three decades.

The spate of innovations and invention in Computer technology during the last decade has lead to the development of mini, Micro and personal Computers. They are so versatile that they have become indispensable to engineers, scientists, business executives, managers, administrators, accountants, teachers and students. They have strengthened man's power in numerical computation and information processing and thereby have increased the effectiveness of organisations.

Modern computers possess certain characteristics and abilities that are peculiar to them. They can :

1. Perform complex and repetitive calculations rapidly and accurately.
2. Store large amounts of data and information for subsequent manipulations.
3. Hold a program of a model which can be explored in many different ways.
4. Make decisions.
5. Provide information to the user.
6. Automatically correct or modify by providing signals certain parameters, of a system under control.
7. Draw and print graphs, and
8. Converse with users through terminals.

## COMPUTER TYPES

Based on the operating principle, computer can be classified into one of the following types:

1. Digital Computers,
2. Analog Computers and
3. Hybrid Computers.

### DIGITAL COMPUTERS.

Digital computers are useful for evaluating arithmetic expressions and multiplications of data (such as preparation of bills, ledgers, solution of simultaneous equation, etc ).

### ANALOG COMPUTERS .

Operate by measuring rather than by counting. The name which is derived from the greek word analog denotes that the computers functions by establishing similarities between two quantities that are usually expressed as voltages or currents, analog computers are powerful tools to solve differential equations.

Computers which combine features of both analog and digital type are called hybrid computers. A majority of the computers used in the world today are digital. Modern Computers , depending upon their application are classified as under:

1. Special purpose computers, and
2. General purpose computers.



Special purpose Computers are designed and built solely to cater to the requirements of a particular task or application. A special purpose computer incorporates the instructions needed into the design of internal storage so that it can perform the given task on a simple command. It therefore does not possess unnecessary options and costs less.

On the otherhand, general purpose computers are designed to meet the needs of many different applications. In a general purpose computers the instructions needed to perform a particular task are not wired permanently into the internal storage. Rather, they are read from an input device and placed into the internal memory unit they are needed.

#### COMPUTER GENERATIONS.

The era of modern computers began in the 1950s when the UNIVAC developed by Remington Rand was put to commercial applications. The Vacuum tube was the main electronic component in the system. Later, such computers were termed as first generation computers. Since the tubes occupy large space and generate lot of heat, the first generation computers were large in size. In addition, they had the following limitations:

1. Slow operating speeds,
2. Restricted computing capacity.
3. Limited programming capabilities and
4. Short life-span.

The development of a device called transistor had a massive effect on the computer field and the second generation computer using transistors were introduced in the late 1950s. The transistors are fast operating devices and occupy considerably less space. They are comparatively more reliable than the vacuum tubes and generate almost no heat.

Due to the improved characteristics of transistors, the second generation computers were:

1. Faster in operations
2. Smaller in size
3. Lesser in cost.
4. Higher in reliability.

Third Generation - The growth of semi conductors technology continued and integrated circuits (ICs) which combine hundreds of components in a single module were developed.

The use of IC chips gave birth to the third generation computers in the mid 1960s with increased arithmetic capability, ability to perform operations in parallel and improved means of interaction. The size and cost reduced considerably. Mini Computers appeared in the market.

Fourth Generation - Continued developments and new techniques finally resulted in capabilities to pack thousands of components into extremely small assemblies known as LSI (large scale integration) assemblies. The largest child of the computer



family that users LSI has been named as fourth generation computers. These computers have increased the user-computer interaction capabilities and speed.

### BITS, WORDS AND MEMORY

The memory of a computer can be thought of as Cells is as shown in fig. 1.01 Each of these cells is further broken down into what are known as bits. The word bit is a contraction of the two words binary digit. These bits are represented by 0 and 1 and are used to store instructions and data by their combinations.

A group of bits representing data or instructions that form the basic information unit of the computer is called a word. Each cell represents a word and may have a length of 16 bits, 32 bits, 48 bits and so on.

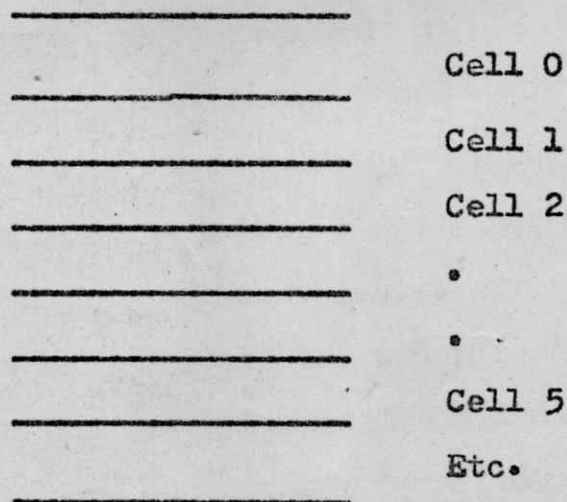


Fig 1.01 Computer Memory Cells

A computer word can also be defined in terms of what are known as bytes. A byte refers to a group of bits used to designate a single alphanumeric or special character. Each byte contains 8 bit. A word may be formed using two or more bytes, the most common being the four-byte combination. Such words, of course, contain 32 bits.

### Digital Computer Organization.

The classical block diagram of a computer has five units. These are the input unit, output unit memory or store, arithmetic unit and the control unit. The arithmetic unit and the control unit are normally combined and called the Arithmetic Logic Unit (ALU) or central processing unit. (CPU).

A block diagram showing the structure of the computer and the interconnections of the various units is drawn in the fig 1.02. The input unit is usually a teletypewriter, a video terminal with a key board , a punched card reader, magnetic tape, a magnetic disk or a paper tape reader. The memory consists usually of magnetic cores or semiconductor storage. The arithmetic unit consists of electronic registers , accumulators and related circuitry. The control unit has electronic circuitry to decode instructions and activate the other units. The output unit is normally a video display terminal , a teletypewriter, a high speed printer , a card punch paper tape punch or a magnetic disk or tape. A digital computer also has a console which

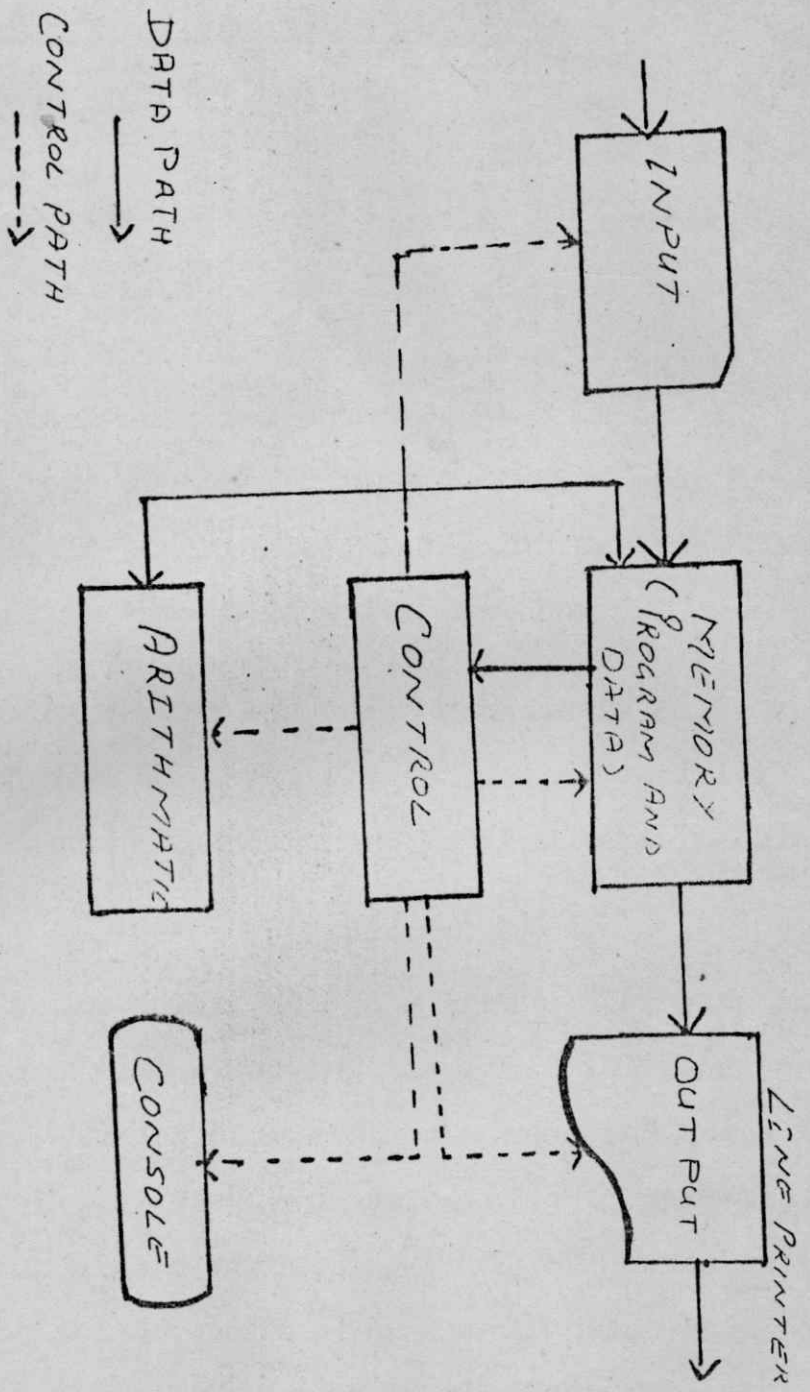


Fig.

BLOCK DIAGRAM OF A COMPUTER



provides a minor and supplementary link for human intervention.

### Description of units in the Computer.

#### Input Devices

Punched cards are used extensively to input informations to computers. A punch card is a rectangular thin card of size 8.8 cm x 8.3 cm . It is divided into 80 columns and 12 rows. Rows from bottom to top are numbered 9 to 0. Rows 11 and 12 punched holes in these rows are called zone punches. Cards are punched on machine called a punching machine. The punching machine has a key board similar to a type writer key board.

A companion machine to the key punch machine is called a verifier. The punched deck of cards are fed to the verifier and the same data from the source document are keyed on this machine by another operator.

Cards are read using a unit called a card reader attached to a computer. Cards are read by moving them past the reading station of the reader. The reading station senses the holes in card either mechanically by wire brushes passing through the holes and making contact with a conducting spot, or photoelectrically. Electrical impulses corresponding to the holes in a card are transmitted and stored in the memory of the computer.

Another data and program entry device coming into vogue

with the advent of time shared computers are video terminal with keyboards and teletypewriters. Video terminals are similar to television sets with attached Keyboards. As keys corresponding to characters are pressed on the keyboard they are display on the television screen. Simultaneously they are entered into the magnetic disk memory connected to the computer.

### Output devices

Output devices are required to present the answers computed by a computer in a readable form. The most popular output device is a line printer that can print one complete line at a time. The maximum number of characters that can be printed at a time is normally 132. Printing speed are usually between 300 lines per minute to 1800 lines per minute.

Video terminals and teletypewriters can also be used to display output information.

### Memory Devices.

The memory or storage device in a Computer is used to store the coded instructions corresponding to a program. It also temporarily stores data before it is processed and intermediate result and the answers before they are printed. Internally all informations is stored in a uniform manner using binary coding.



Beside the main memory a computer also has an auxillary memory made of magnetic tapes and disks to store data and programs when these cannot be accomodated in the main memory.

#### ARITHMETIC AND CONTROL UNITS.

The arithmetic unit does all the computation in a computer. It is made of fast electronic circuits which use semi conductor processing elements. The elements are integrated on a tiny chip of semiconductor. Speeds of addition are now in the range of tens of nanoseconds per digits where nano stands for one billionth ( $10^{-9}$ ). The control unit intergrets instruction and shedule schedules jobs to be performed by the various units of a computer. The control unit is also made of integrated circuits

Auxillary or secondary memory system is used in most computers. The secondary memory system stores more than 100 times the information that can be stored in the main memory. The cost of storing information in secondary memory is much lower but the time to access information from it is larger. In a balanced computer configuration such a memory is essential to store the main data files and programs and to provide a back up for the main memory the most common secondary memories are the magnetic tape and magnetic disk memories.

## Magnetic Tape

Magnetic tape memories are analogous to the familiar audio tape recorders. Basically a magnetic tape drive consists of a spool which a magnetic tape is wound. The tape is a 1/2 inch wide plastic tape finally coated with a magnetic oxide material. The tape is transported across a set of magnetic heads and is taken up on another spool. The heads are mounted between the two spools and are used to write and read information from the tape.

## Disk Drives

Magnetic disks are smooth metal plates coated with magnetic material. A set of such magnetic plates are stacked one below the other, to make up a disk pack. The disk pack is mounted on a spindle. A set of reading/writing heads are mounted on arms. The arm assembly is capable of moving in and out in a radial direction. Usually a diskette with 96 and 48 tracks per inch of single side and double side with single side and double side density are used for storing information. On this basis the storage is calculated. If it is a single side single density the maximum storage is 256 K. The upper surface of the top plate and the lower surface of the bottom plate are not used. Information is stored on tracks on each plate. A set of corresponding tracks on all the surfaces is called a cylinder. If disk pack with ten surfaces will

have ten tracks per cylinder. A track is further subdivided into sectors. A set of records is stored in a sector.

### SOFTWARE.

The set of programs that are run in a computer is called Software. A program is nothing but a set of sequenced instructions to the computer. The process of preparing the program is called programming and the man who does all this job is called a programmer. These are four major categories of software:

1. Operating systems,
2. Utility programs,
3. Language processors, and
4. Application programs.

1. Operating Systems: These are program usually written by computer manufacturers. These programs reside in the computer hardware components, such as processors, memory devices and input/output devices. They in fact act as an interface between the users programs and the computer components and facilitate in the execution of his programs. Operating system can handle several programs at a time in the time sharing mode and can optimize the use of computer hardware. The operating system of a computer is one of the main factors which contribute to its capabilities.



2. Utility Programs: These are also prewritten programs, usually by the manufacturers and supplied with the system. These are increasingly used in data-base management systems to manipulate data and print out results. A good range of such utility programs can make the task much simpler for the user. Other examples of utility, programs are:
  1. Program to load a program into a memory.
  2. Program to duplicate magnetic tapes.
3. Language Processors : This software is used to translate the programmer written instructions into machine code instructions. This is a machine dependent software and is known as assembler/ compiler.
4. Application Programs.: Application programs are user-written programs to do some specific jobs, They are unique in their construction and can be used only for identical jobs. Recently many such application programmes have been made commercially available under the name program packages. These can be changed marginally to fit individual needs and to meet the system requirements standard packages for application areas such as pay rolls, billing, PERT/CPM, inventory control and linear programming are available.

#### PROGRAM PROCESSING MODES.

A modern computer system can be used in three modes of operations:

1. Batch Processing mode,
2. Time sharing mode, and
3. Interactive time sharing mode.

Batch Mode : Early computers were all designed to be used in the batch-processing mode where programmes were presented to the system in batches. The system would process the programs one after the other. In this mode, one user has complete control of the machine until his program is completely executed.

Time shared- Computing: In the time sharing mode each user has a local input/output device called a terminal and he can use a distant computer by entering his programs at his terminals. Time-sharing is a means of making the resources of one computer system to serve the needs of many users simultaneously. We know that the computer does not do several things at once, but it can be made to jump from one task to another so rapidly that the individual user is not aware of any long delays. Large modern systems can serve a hundred or more terminals all using a variety of languages and in both batch and conversational modes.

Intractive Computer systems: An interactive computer systems put the user into direct conversation with the computers through a terminals usually a typewriter. In this mode the programmer directly enters his program at the terminal and begins to get answers only a few seconds after he tells the computer to



execute his job. He may examine the output and decide what he wants to do next. If any errors are made while typing in his program at the terminal, they will be immediately known to the programmer. In this mode of operations the programmer, can control the steps of the calculations as it progresses.

#### LANGUAGE OF THE COMPUTER

The functioning of a computer is controlled by a set of instructions. A complete set of such instructions is called a computer program. These instructions are written by a programmer to solve a particular problem. An instruction may tell the computer.

What operation to perform,  
Where to locate data,  
How to present results,  
When to make certain decisions,  
and so on.

The communication between two parties, whether they are machines, or human beings, always read a common language or terminology. The language which is used in the communication of Computer instructions is known as programming language. The computer has its language or translated into this language. Along with the developments of hardware components, the improvements in the programming language have been equally significant

## HIGH LEVEL LANGUAGES.

The limitation of machine and assembly language encouraged the search for alternative and gradually a set of new language called high-level languages emerged .

A high-level language consists of a set of words and symbols using which the programmer can write a program, in conjunction with certain rules, similar to the English language. The languages are oriented towards the problem to be solved or procedures for solution rather than computer instructions. These are more user centred than the machine-centred languages. These languages are therefore referred to as user-oriented or procedure-oriented languages.

The most important characteristics of a high level language is that it is machine-independent and a program written in a high - level language can be run on computers of different makes with little or no modifications.

Following are the list of some high- level languages used in general.

1. FORTRAN

FORmula TRANslation developed in 1957.

2. COBOL

Common Business Oriented Language developed in late 1950 s .

3. BASIC

Beginner's All - purpose Symbolic Instruction Code developed in late 1960 s .

4. RPG  
Report Program Generator

5. PL/I  
Programming Language I developed by IBM in the middle  
of 1960 s .

6. PASCAL

7. Medium Level Language C



## WIPRO Z-650

The system installed in S.A.T.I., Vidisha is WIPRO Z-650. Some features of the systems are as follows:

### Powerful 16-bit Micro-Processor.

The central processor of the WIPRO Z-650 is the Intel 1 APX 86/10 (8086) Microprocessor. The CPU architecture includes four 16-bit byte addressable data registers, two 16-bit memory base pointed registers and two 16 bit index registers. All accessed by a total of 24 operand addressing modes for Compre-hensive memory addressing and for support of the data structures.

The processor allows one megabyte of address space, 64 KB of input /output device address, space and 16 bit internal and external data paths.

The 8086 instruction set includes a variable length instruction format (including double operand instructions ) 8 bit and 16 bit signed and unsigned arithmetic operators for binary , BCD and unpacked ASC-II data , and interactive word and byte string manipulation functions.

A 6 -byte instruction queue provides pre-fetching of sequential instructions and can reduce the minimum instruction cycle time to one third.

## MULTIBUS

The electronic sub-systems of the WIPRO Z-650 are interconnected by Multibus (~~IEEE p 796 Bus~~). The multibus is an asynchronous bus accomdating multiple 16-bit Micro processor and intelligent peripheral controllers with various transfer rates while maintaining maximum throughput. This standard bus has mechanical and electrical interface specifications.

## SYSTEM FEATURES

### Hardware

- 16 bit processor
- 80 bit Numeric Data Processor
- 512 KB of main memory
- One winchester disk drives
- Printer with graphic options
- Four interactive VDU,
- Graphic Visual display Unit.
- One coloured Dot Matric Printer.

### Software

- Multi user operating systems, DBOS and UNIX.
- Choice of Languages-WIPRO FORTRAN ,WIPRO PASCAL,  
WIPRO BASIC ,WIPRO COBOL .
- Word Processor
- Scientific and Statistical Packages,
- Graphics.



## PROGRAMMING IN COBOL

### The COBOL Language.

COBOL stands for Common Business Oriented Language. This language was proposed in 1960 and is almost universally used as the programming language for business data processing.

Historically, COBOL grew out of the desire of the data processing professionals for a high level machine independent language for business data processing which would be accepted by any computer which has a COBOL compiler. This work began in 1959. The maintenance and further orderly growth of the language was handed over to a group called the Conference on Data Systems Languages abbreviated CODASYL. The first formal COBOL report was published in 1960 and revised in 1965. This version was gradually improved and on American National Standards Institute (ANSI) COBOL was standardized in 1968 and 1974.

### COBOL Divisions

A COBOL program is divided into four parts called DIVISIONS. These four divisions describe four main aspects of a program and are called :

IDENTIFICATION DIVISION.

ENVIRONMENT DIVISION.

DATA DIVISION and

PROCEDURE DIVISION.

Each COBOL program requires all the four divisions and they must appear in the above order. Each DIVISION may be further divided into several sections and the sections into paragraphs. Paragraphs may be broken into many sentences and sentences into statements. Statements may be divided into several phrases and phrases into words.

COBOL Coding form

COBOL programs are written on a coding sheet called COBOL coding form. A program written on such a form is punched one line per card on a key punch machine or a key board entry system, by a punch operator. The form has 80 columns. The 80n columns are divided into five distinct parts as shown in Table 2.00. The parts and their functions are :

- cols 1 to 6 sequence number field
- 7 continuation indicator or remark indicator.
- 8 to 12 Name field.
- 12 to 72 Text field.
- 73 to 80 Identification field.

1	6 : 7 : 8	12	72 : 73	80
:	:	:	:	:
:	:	:	:	:
:	:	:	:	:
:	:	:	:	:
:	:	:	:	:
Seq No.	A Margin	B Margin	Ident.	

Between columns 1 and 6 a sequence number for each line in the COBOL program is given. Such numbers are optional. If programs are entered through an interactive video terminal the editing program normally creates the sequence number automatically.

#### Continuation and Comments :

Column 7 has two uses. If writing a line column 72 is reached and a word is partially written the rest cannot be written on the same line as the information between columns 73 and 80 is ignored by the computer. The word may be continued from column 12 of the next line. This line is identified as a continuation line by placing a hyphen in column 7. To indicate that a line in a program text is a comment an asterisk (\*) is placed in column 7.

#### A and B Margins

The names of divisions, sections and paragraphs begin in column 8. Column 8 of the COBOL coding form is known as the A Margin.

All COBOL sentences start from column 12 and may extend up to column 72. Column 12 is known as the B margin.

#### COBOL Character set.

The set of characters available in COBOL language consists



of fifty-one characters. These are:-

Letters ( alphabetic characters)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Digits( Decimal numeric characters)

0 to 9

Special characters

- Blank (space) represented as  $\text{\textbackslash}$  in the back
- + Plus sign
- Minus sign as well as hyphen
- \* Asterisk also used as a multiplication symbol
- / Slash also used as division symbol
- = Equal sign
- \$ Dollar sign or currency sybol
- ,
- ;
- . Full stop, period or decimal point
- " Quote mark
- ( Left parenthesis
- ) Right parenthesis
- > Greater than symbol
- < Less than symbol

### DATA DIVISION

The DATA DIVISION is that part of a COBOL program where every data item processed by the program is described. It is important to note that unless a data item is described in the

DATA DIVISION, it cannot be used in the procedure division. The DATA DIVISION is divided into a number of sections and depending on the use of a data item. The main two sections of the DATA DIVISION are as follows:-

a) FILE SECTION

The FILE SECTION includes the descriptions of all data items that should be read from or written into some external file.

b) WORKING-STORAGE SECTION

The data items which are developed internally as intermediate results as well as the constants are described in this section of the DATA DIVISION.

The format of the DATA DIVISION is as follows:

DATA DIVISION.

[FILE SECTION.

File section entries.

.....

.....]

[WORKING-STORAGE SECTION.

Working - Storage entries.

.....

.....]

LEVEL STRUCTURE

The data to be processed are internally stored in a specific area in the memory of a computer. They are corresponding to

a particular data item is referenced by the data name used in the description of the said item. Data names are user - created words.

#### DATA ENTRIES

An entry is the description of a data item in the DATA DIVISION. An entry consists of the level number, data name and various data-description clauses. For the time being we shall consider only two of these data-description clauses-- the PICTURE clause and VALUE clause. It may be noted that in the data entry these clauses can appear in any order. The level number in an entry must begin from the margin A ( column 8 ) or any position after it. In the case of level 01 entries, The level number must begin from the margin A or any position before the margin B (column 12). The data name in an entry must begin from the margin B or any position after it, There must be at least one space between the level number and data name. An entry must end with a terminating period.

#### PICTURE Clause

The PICTURE clause describes the general characteristics of an elementary data item. These characteristics are described below:

a. Class

In COBOL a data item may be one of the three classes-- numeric, alphabetic or alphanumeric. As indicated by



these names, the numeric, items consist only of digits 0 to 9 and the alphabetic items consists only of the letters A to Z and the space (blank) character. The alphanumeric items may consist of digits, alphabets as well as special characters.

b) Sign

A numeric data item can be signed or unsigned .

c) Point Location

The position of the decimal point is another characteristic that can be specified in the case of numeric data items.

d) Size

Size is another characteristic which specify the number of characters or digits required to store the data item in the memory.

The four general characteristics described above can be specified through a PICTURE clause. This clause may also be used to describe other characteristics

The PICTURE Clause is to be followed by a picture character string as shown below.

```
{ PICTURE } IS character-string  
  PIC
```

Code character

Meaning

9	Each occurrence of this code in the picture string indicates that the corresponding character position in the data item contains a numeral.
x	Each occurrence of this code indicates that the corresponding character position in the data item contains any allowable character from the COBOL character set.
A	Each occurrence of this code indicates that the corresponding character position in the data item contains only a letter or space character.
V	The occurrence of this in a picture string indicates the position of the assumed decimal point.
P	The occurrence of this indicates the position of the assumed decimal point when the point lies outside the data item.
S	The occurrence of this indicates that the data item is signed.

The allowable combinations are governed by the following rules:

- i. In the case of an alphabetic item the picture may contain only the symbol A.
- ii. In the case of a numeric item the picture may contain only the symbols 9, V, P and S. These are called operational characters. It must contain at least one 9.
- iii. In the case of an alphanumeric item, the picture may contain all Xs or a combination of 9, A and X (except all 9 or all A).

The PICTURE clause is only to be specified for elementary items; it cannot be used for a group item. The size of a group item is equal to the total of the sizes of all subordinate elementary items.

#### VALUE Clause

The VALUE clause defines the initial value of a data item. Normally the initialization is done just before the first statement in the PROCEDURE DIVISION is executed. The syntax of the VALUE clause in its most simple form is

VALUE            IS literal

#### PROCEDURE DIVISION AND BASIC VERBS

The procedure division contains statements which specify the operations to be performed by the computer. Each of these statements is formed with COBOL words and literals.

#### DATA MOVEMENT VERB: MOVE

It frequently becomes necessary to move data from one place in the memory to another place. This is done with the help of the MOVE verb. The general form of the MOVE verb is as follows:-

MOVE        {    identifier-1  
              ;    literal-1        }    TO identifier-2 [, identifier-3]...



Data movement is governed by the following rules.

a) The contents of identifier-1 or the value of literal-1 is moved to identifier-2, identifier-3, etc.

b) When the sending field is numeric and the receiving field is numeric or numeric edited (i.e., picture contains edit symbols) the data movement is called numeric data transfer.

c) When both the sending and receiving fields are alphabetic alphanumeric or alphanumeric edited, the data movement is called alphanumeric data transfer.

#### ARITHMETIC VERBS

Most of the problems requires some computations to be performed on the input or intermediate data which are numeric in nature. Arithmetic verbs are used to perform these computations. All these verbs can contain either identifiers or numeric literals or both. Some arithmetic verbs are given below:

- i. ADD
- ii. SUBTRACT
- iii. MULTIPLY
- iv. DIVIDE

#### SEQUENCE CONTROL VERBS

Sequence control verbs are

- i. GO TO : This verb is used to unconditionally transfer the control to elsewhere in the program.

ii. STOP: This verb causes the termination of the execution of the object program. STOP RUN

INPUT AND OUTPUT VERBS

Examples of input-output operations are as follows:

- i. OPEN
  - ii. READ
  - iii. WRITE
  - iv. CLOSE
  - v. ACCEPT
  - vi. DISPLAY
-

PROBLEM No.1 : Program for tabulation of sales report of a number of items sold in a day by various sales girls at a Super Market.

ALGORITHM :

There are a number of shops and 5 Sales girls, various items are given code numbers, and sales girls are also given code numbers.

At the time of closing of shop, we want to know how many items have each sales girl sold and their total cost. When a salesgirl have sold an item, she enters in the input file her shop No., Sales Girl No. ITEM CODE AND PRICE.

Basic Steps of Programming :

STEP : 1 : Read the shop No., Sales Girl No., Item Code & Price from In -File.

STEP : 2 : Print heading.

STEP : 3 : Move all the items in working storage area write sales record from working storage area.

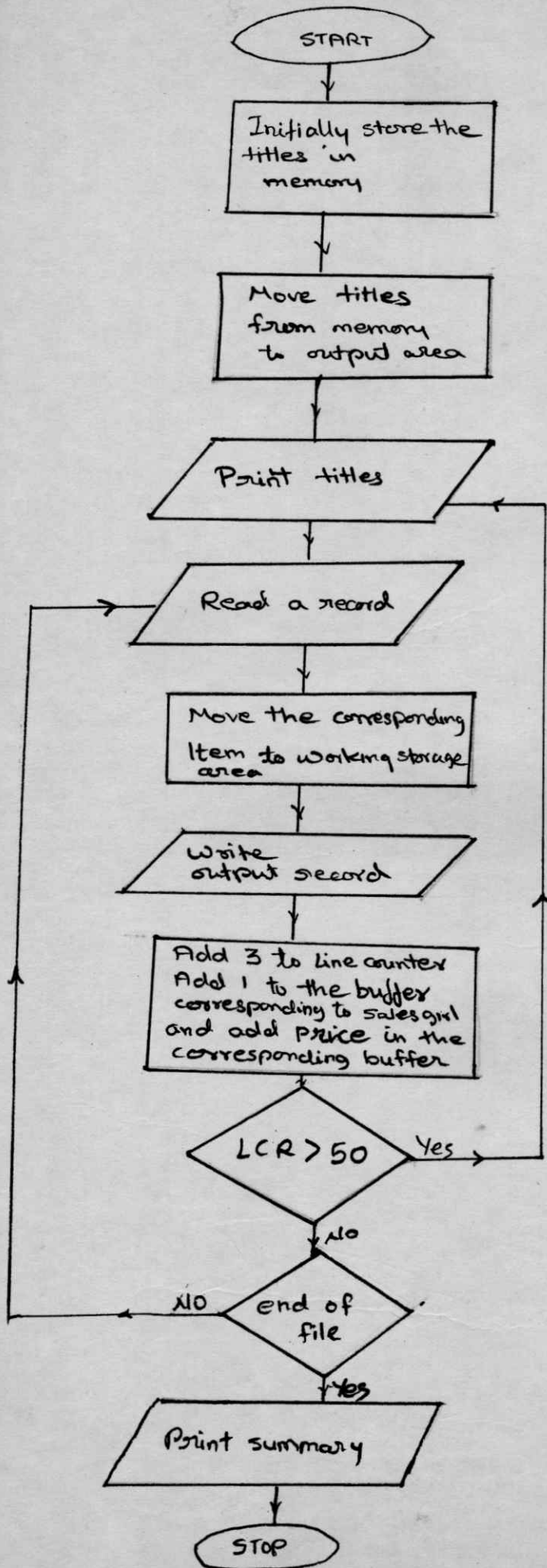
STEP : 4: Take five buffer areas for 5 sales girls. Add one in the buffer corresponding to the sales girls, every time.

STEP : 5: At the end of Input file, print summary giving total items and amounts sold by various sales girls.

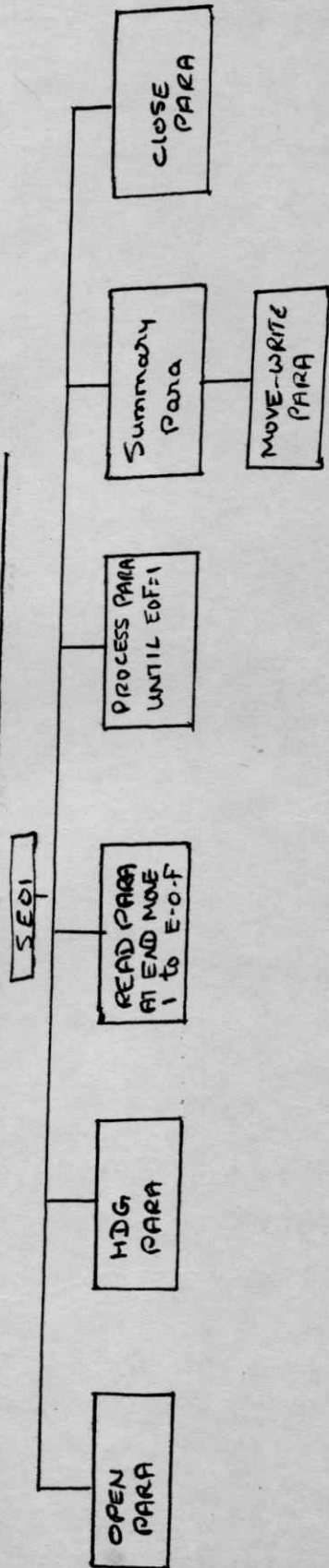
STEP : 6 : Stop.



# FLOWCHART FOR TABULATION OF SALES REPORT



## MODULAR STRUCTURE FOR PROGRAM SE01



QQQ  
 QQ  
 QQ FILE :TRG.SALE(16) UPDATED 13/01/87 A  
 QQ  
 QQ JOB \*TRG.JOB2572  
 QQ  
 QQ OUTPUT 14/01/87 AT 12.46.36 ON UNIT LP01  
 QQ  
 QQ STREAMS CENTRAL  
 QQ  
 QQQ

COB(LISTING = SOURCE, CODE = NE)  
 CONS(OUTPUT = PROGRAM BHEL)  
 \*\*\*\*

IDENTIFICATION DIVISION.  
 PROGRAM-ID. SE01.  
 ENVIRONMENT DIVISION.  
 CONFIGURATION SECTION.  
 SOURCE-COMPUTER. ICIM 6080.  
 OBJECT-COMPUTER. ICIM 6080.  
 INPUT-OUTPUT SECTION.  
 FILE-CONTROL.  
     SELECT IN-FILE ASSIGN TO CRO1.  
     SELECT SALES-FILE ASSIGN TO LP01.  
 DATA DIVISION.  
 FILE SECTION.

FD IN-FILE.  
 01 SALES-DATA.  
     02 SHOP-NO PIC 999.  
     02 SALES-GIRL-NO PIC 999.  
     02 ITEM-CODE PIC 999.  
     02 PRICE PIC 999V99.

FD SALES-FILE.  
 01 OUTPUT-SALES-RECORD.  
     02 F PIC X(120).  
     02 F PIC X(12).

WORKING-STORAGE SECTION.  
 01 E-O-F PIC X VALUE ZERO.  
 01 HD1.

    02 F PIC X(50) VALUE SPACES.  
     02 F PIC X(30) VALUE "S A L E S                      R E C O R D".

    02 F PIC X(52) VALUE SPACES.  
 01 HD2.  
     02 F PIC X(20) VALUE SPACES.  
     02 F PIC X(7) VALUE "SHOP NO".  
     02 F PIC X(5) VALUE SPACES.  
     02 F PIC X(13) VALUE "SALES GIRL NO".  
     02 F PIC X(5) VALUE SPACES.

    02 F PIC X(9) VALUE "ITEM-CODE".  
     02 F PIC X(5) VALUE SPACES.  
     02 F PIC X(5) VALUE "PRICE".  
     02 F PIC X(63) VALUE SPACES.

01 HD3.  
     02 F PIC X(22) VALUE SPACES.  
     02 W-SHOP-NO PIC ZZ9.

    02 F PIC X(12) VALUE SPACES.

02 W-SALES-GIRL-NO PIC ZZ9.  
 02 F PIC X(13) VALUE SPACES.  
 02 W-ITEM-CODE PIC ZZ9.  
 02 F PIC X(8) VALUE SPACES.  
 02 W-PRICE PIC ZZZ9VI.99.  
 02 F PIC X(62) VALUE SPACES.  
 01 HD4.  
 02 F PIC X(20) VALUE SPACES.  
 02 F PIC X(15) VALUE "SALES GIRL NO".  
 02 S-SALES-GIRL-NO PIC 9.  
 02 F PIC X(5) VALUE SPACES.  
 02 F PIC X(11) VALUE "ITEMS SOLD".  
 02 S-SUM-NO PIC ZZ9.  
 02 F PIC X(5) VALUE SPACES.  
 02 F PIC X(5) VALUE "WORTH".  
 02 S-SUM-VALUE PIC ZZZZ9VI.99.  
 02 F PIC XX VALUE "RS".  
 02 F PIC X(59) VALUE SPACES.  
 01 SUMMARY.  
 02 SUM OCCURS 5 TIMES.  
 03 SUM-NO PIC 999.  
 03 SUM-VAL PIC 99999V99.  
 01 LCTR PIC 99 VALUE 0.  
 01 SUB-NO PIC 9 VALUE ZERO.  
 PROCEDURE DIVISION.  
  
 START1.  
 PERFORM OPEN-PARA.  
 PERFORM HDG-PARA.  
 PERFORM READ-PARA.  
 PERFORM PROCESS-PARA UNTIL E-O-F = 1.  
 PERFORM SUMMARY-PARA.  
 PERFORM CLOSE-PARA.  
 STOP RUN.  
 OPEN-PARA.  
 OPEN INPUT IN-FILE.  
 OPEN OUTPUT SALES-FILE.  
 HDG-PARA.  
 WRITE OUTPUT-SALES-RECORD FROM HD1 AFTER PAGE.  
  
 WRITE OUTPUT-SALES-RECORD FROM HD2 AFTER 3.  
 PROCESS-PARA.  
 MOVE SHOP-NO TO W-SHOP-NO.  
 MOVE SALES-GIRL-NO TO W-SALES-GIRL-NO.  
 MOVE ITEM-CODE TO W-ITEM-CODE.  
  
 MOVE PRICE TO W-PRICE.  
 WRITE OUTPUT-SALES-RECORD FROM HD3 AFTER 3.  
 ADD 3 TO LCTR.  
 IF LCTR > 50 GO TO HDG-PARA.  
 MOVE SALES-GIRL-NO TO SUB-NO.  
 ADD 1 TO SUM-NO(SUB-NO).  
 ADD PRICE TO SUM-VAL(SUB-NO).  
 PERFORM READ-PARA.  
 READ-PARA.  
 READ IN-FILE AT END MOVE 1 TO E-O-F.  
 SUMMARY-PARA.  
 PERFORM MOVE-WRITE-PARA VARYING SUB-NO FROM 1 BY 1 UNTIL  
 SUB-NO > 5.



S A L E S

R E C O R D

SHOP NO	SALES GIRL NO	ITEM-CODE	PRICE
1	1	1	396.32
1	2	2	856.32
1	3	2	739.65
1	4	6	785.42
1	5	4	452.13
1	1	2	785.32
1	3	1	785.23
1	2	3	785.23

SALES GIRL NO1 ITEMS SOLD 2 WORTH 1681.64RS

SALES GIRL NO2 ITEMS SOLD 2 WORTH 1641.55RS

SALES GIRL NO3 ITEMS SOLD 2 WORTH 1574.88RS

SALES GIRL NO4 ITEMS SOLD 1 WORTH 785.42RS

SALES GIRL NO5 ITEMS SOLD 1 WORTH 452.13RS

AA

NO. 002346

LEGAL DOCS. RS010UT(1)

UPDATED 13/01/87 AT 12.07.52

AA

AA

AA

PROBLEM : 1 : Program for sorting of a given file according to Name, Designation or Seniority (Age) :

ALGORITHM :

Let ( EMPLOYEE-NAME represents the name of employee, DESIGNATION represents, designation code of the employee, DATE OF JOINING represents the date when employee has joined. PAY represents the monthly earning of the employee.

EMPLOY-FILE contains records having all the above datas.

BASIC STEPS OF PROGRAMMING :

STEP : 1 : Read the a record from EMPLOY-FILE.

STEP : 2 : Write the record on SORTED-FILE which is disk file.

STEP : 3 : After writing all records in sorted file if switch is NAME then sort EMP-S-file which is on disk in ascending order according to SORT EMPLOY-NAME using sorted file and giving employee-S-file.

If Switch is Desg. then sort according to Designation code.

If switch is Senior sort according to Date of Joining.

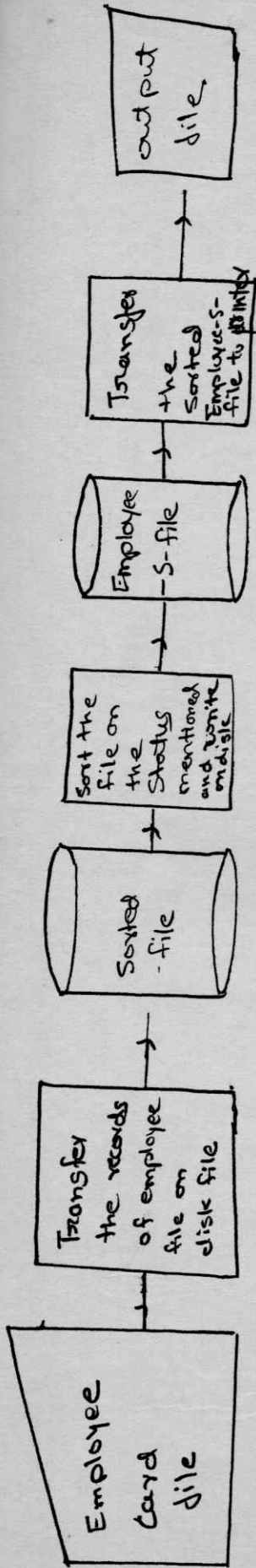
STEP : 4 : Move the records from Emp-S-file to input file.

STEP : 5 : WRITE heading.

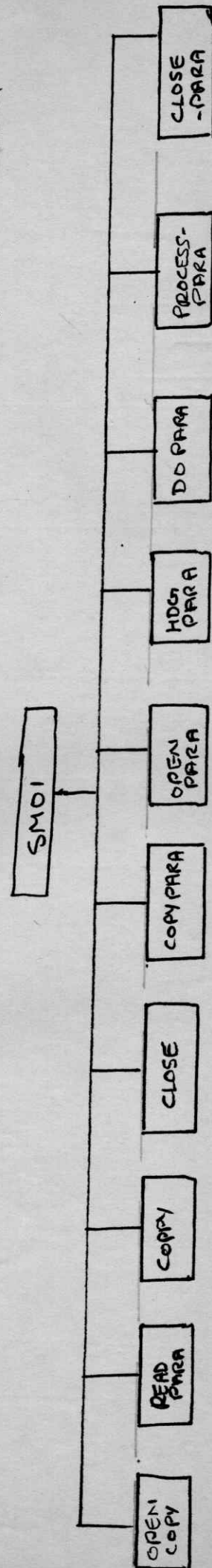
STEP : 6 : Write output file.

STEP: 7 : STOP.

### File Sorting



### Modular Structure for the above sorting of Employee file





PP  
PP FILE :TRG.SORT(42) UPDATED 10/01/  
PP  
PP JOB \*TRG.JOB2572  
PP  
PP OUTPUT 14/01/87 AT 12.42.18 ON UNIT LP01  
PP  
PP STREAMS CENTRAL  
PP  
PP

COBSORTDEF(,SORTWORKFIL1 & SORTWORKFIL2)  
COB(LISTING = SOURCE, CODE = NE)  
CONS(OUTPUT = PROGRAM BHEL)  
\*\*\*\*

IDENTIFICATION DIVISION.  
PROGRAM-ID. SM01.  
AUTHOR. SEEMA.  
DATE-WRITTEN. 061287.  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE-COMPUTER. ICIM 6080.  
OBJECT-COMPUTER. ICIM 6080.  
SPECIAL-NAMES.  
SW01 ON STATUS IS NAME  
SW02 ON STATUS IS DESG  
SW03 ON STATUS IS SINIOR.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
SELECT EMPLOY-FILE ASSIGN TO CRC01.  
SELECT OUTPUT-FILE ASSIGN TO LP01.  
SELECT EMP-S-FILE ASSIGN TO DA01.  
SELECT SORTED-FILE ASSIGN TO DA02.  
DATA DIVISION.  
FILE SECTION.  
FD EMPLOY-FILE.  
01 EMPLOYEE-REC.  
02 EMPLOYEE-NAME PIC X(15).  
02 DESIGNATION PIC XX.  
02 DATE-OF-JOINING PIC X(6).  
02 PAY PIC 9(5)V99.  
SD EMP-S-FILE  
BLOCK CONTAINS 2048 CHARACTERS.  
01 SORT-REC.  
02 SORT-EMPLY-NAME PIC X(15).  
02 SORT-DESG PIC XX.  
02 SORT-DATE-JOINING PIC X(6).  
02 SORT-PAY PIC 9(5)V99.  
02 F PIC XX.  
FD SORTED-FILE  
BLOCK CONTAINS 2048 CHARACTERS  
RECORDING MODE IS F  
LABEL RECORDS STANDARD  
VALUE OF ID "WORKFILE1"  
ACTIVE-TIME 0.  
01 SORTED-RECORD.  
02 WS-NAME PIC X(15).  
02 WS-DESG PIC XX.  
02 WS-DATE-JOINING PIC X(6).  
02 WS-PAY PIC 9(5)V99.  
02 F PIC XX.  
FD OUTPUT-FILE.

01 OUTPUT-REC.

02 F PIC X(120).

02 F PIC X(12).

WORKING-STORAGE SECTION.

01 HD1.

02 F PIC X(20) VALUE SPACES.

02 F PIC X(14) VALUE "EMPLOYEE NAME".

02 F PIC X(5) VALUE SPACES.

02 F PIC X(17) VALUE "DESIGNATION CODE".

02 F PIC X(5) VALUE SPACES.

02 F PIC X(15) VALUE "DATE OF JOINING".

02 F PIC X(5) VALUE SPACES.

02 F PIC X(3) VALUE "PAY".

02 F PIC X(48) VALUE SPACES.

01 HD2.

02 F PIC X(120) VALUE ALL "--".

02 F PIC X(12) VALUE ALL "--".

01 HD3.

02 F PIC X(20) VALUE SPACES.

02 PS-NAME PIC X(15).

02 F PIC X(10) VALUE SPACES.

02 PS-DESG PIC XX.

02 F PIC X(15) VALUE SPACES.

02 PS-DATE-JOINING PIC XX/XX/XX.

02 F PIC X(11) VALUE SPACES.

02 PS-PAY PIC Z(5).99.

02 F PIC X(43) VALUE SPACES.

01 E-O-F PIC X VALUE ZERO.

01 SW PIC X VALUE ZERO.

PROCEDURE DIVISION.

START1.

PERFORM OPENCOPY.

PERFORM READ-PARA.

PERFORM COPY UNTIL E-O-F = 1.

CLOSE EMPLOY-FILE

SORTED-FILE.

PERFORM COPY-PARA.

PERFORM OPEN-PARA.

PERFORM HDG-PARA.

PERFORM DO-PARA.

PERFORM PROCESS-PARA UNTIL SW = 1.

PERFORM CLOSE-PARA.

STOP RUN.

OPENCOPY.

OPEN INPUT EMPLOY-FILE

OUTPUT SORTED-FILE.

COPY.

MOVE EMPLOYEE-REC TO SORTED-RECORD.

WRITE SORTED-RECORD.

PERFORM READ-PARA.

COPY-PARA.

IF NAME SORT EMP-S-FILE ON ASCENDING KEY SORT-EMPLY-NAME  
USING SORTED-FILE GIVING SORTED-FILE.

IF DESG SORT EMP-S-FILE ON ASCENDING KEY SORT-DESG  
USING SORTED-FILE GIVING SORTED-FILE.

IF SINIOR

SORT EMP-S-FILE ON ASCENDING KEY

SORT-DATE-JOINING

ASCENDING KEY SORT-DESG

USING SORTED-FILE,

GIVING SORTED-FILE.

OPEN-PARA.

OPEN INPUT SORTED-FILE.

OPEN OUTPUT OUTPUT-FILE.

HDG-PARA.













### PROBLEM SOLVING : 3

Problem Program for tabulation of working  
time list of a Company.

#### Algorithm :

Let Working -Name represents the names of the employees in a Company. Working-Regular-Hours represents the total fixed working hours of an employee. Working-Overtime- Hours represents the total overtime hours during the total period.

#### BASIC STEPS OF PROGRAMMING :

STEP : 1: Read Working-Name, Working-Regular-Hours, Working-Overtime-Hour from the In-File i.e. input data file. Initialize S.No. to Zero.

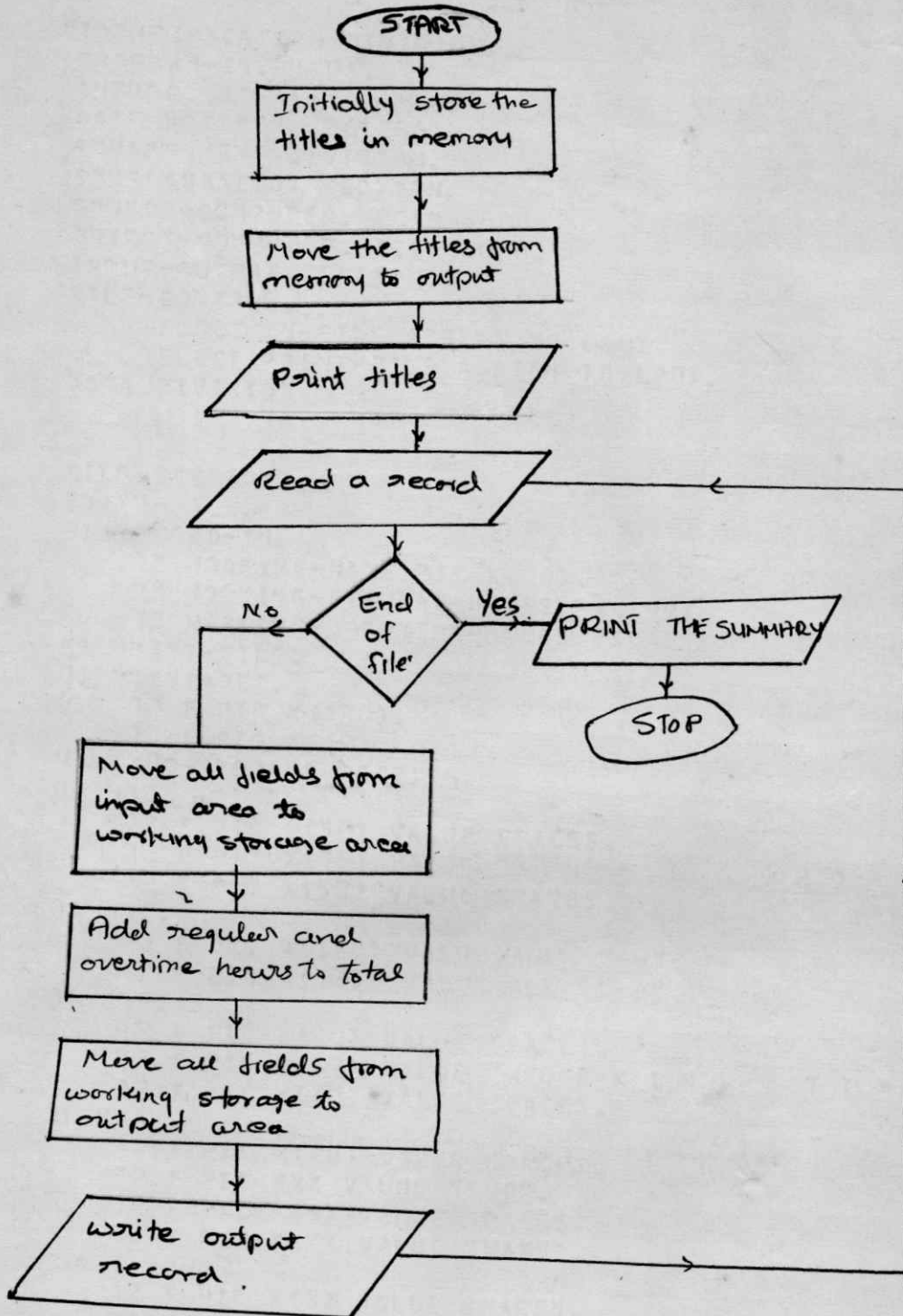
STEP : 2: Add Working-Regular-Hours and Working-Overtime-Hours to obtain total hours Add 1 to S.No.

STEP : 3: Moves the working -Name, Working-Regular-Hours, Working-Overtime-Hours and total Hours from input file to the output file i.e. Print-File.

STEP: 4: Print the heading and then all the records in Print-File.

STEP : 5: At the end of input file. Print a summary giving total number of employees.

STEP : 6 : Stop.



## SOURCE LISTING

IDENTIFICATION DIVISION.  
PROGRAM-ID. HR01.  
AUTHOR. SEEMA.  
DATE-WRITTEN. 261286.  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE-COMPUTER. ICIM 6080.  
OBJECT-COMPUTER. ICIM 6080.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
SELECT IN-FILE ASSIGN TO CRO1.  
SELECT PRINT-FILE ASSIGN TO LP01.  
DATA DIVISION.

## FILE SECTION.

FD IN-FILE.

01 RECORD-IN.

02 WORKING-NAME PIC X(20).  
02 WORKING-REGULAR-HOURS PIC 99.  
02 WORKING-OVERTIME-HOURS PIC 99.

FD PRINT-FILE.

01 PRINT-REC.

02 F PIC X(120).  
02 F PIC X(12).

## WORKING-STORAGE SECTION.

01 W01.

02 F PIC X(50) VALUE SPACES.  
02 F PIC X(25) VALUE "B H E L - 9 H O P A L".  
02 F PIC X(57) VALUE SPACES.

01 W02.

02 F PIC X(120) VALUE ALL "--".  
02 F PIC X(12) VALUE ALL "--".

01 W03.

02 F PIC X(40) VALUE SPACES.  
02 F PIC X(32) VALUE "W O R K I N G T I M E L I S T".  
02 F PIC X(60) VALUE SPACES.

01 W04.

02 F PIC X(20) VALUE SPACES.  
02 F PIC XXX VALUE "SNO".  
02 F PIC X(5) VALUE SPACES.  
02 F PIC XXXX VALUE "NAME".  
  
02 F PIC X(3) VALUE SPACES.  
02 F PIC X(20) VALUE "REGULAR HOURS WORKED".  
02 F PIC XXXX VALUE SPACES.  
02 F PIC X(21) VALUE "OVERTIME HOURS WORKED".  
02 F PIC XXXX VALUE SPACES.  
02 F PIC X(12) VALUE "TOTAL HOURS WORKED".  
02 F PIC X(25) VALUE SPACES.

01 W05.

02 F PIC X(20) VALUE SPACES.  
02 M-SNO PIC 999.  
02 F PIC X VALUE SPACES.  
02 M-NAME PIC X(15).  
02 F PIC X(12) VALUE SPACES.  
02 M-RHOURS PIC 99.  
02 F PIC X(21) VALUE SPACES.



## SOURCE LISTING

```

02 W-OHOURS PIC 99.
02 F PIC X(14) VALUE SPACES.
02 W-THOURS PIC 999.
02 F PIC X(39) VALUE SPACES.
01 E-O-F PIC X VALUE ZERO.
01 SNO PIC 999 VALUE ZERO.
01 EMPLOYEE-TOTAL-LINE.
02 F PIC X(52) VALUE SPACES.
02 F PIC X(25) VALUE "NUMBER OF EMPLOYEE IS".
02 NUMBER-OF-EMPLOYEES PIC 999.
02 F PIC X(52) VALUE SPACES.
01 TOTAL-HOURS-WORKED PIC 999 VALUE ZERO.
PROCEDURE DIVISION.

START1.
PERFORM OPEN-PARA.

PERFORM HDG-PARA.
PERFORM READ-PARA.
PERFORM PROCESS-PARA UNTIL E-O-F = 1.

WRITE PRINT-REC FROM EMPLOYEE-TOTAL-LINE.
PERFORM CLOSE-PARA.
OPEN-PARA.
OPEN INPUT IN-FILE.
OPEN OUTPUT PRINT-FILE.
READ-PARA.
READ IN-FILE AT END MOVE 1 TO E-O-F.
HDG-PARA.
WRITE PRINT-REC FROM HD1 AFTER PAGE.
WRITE PRINT-REC FROM HD2 AFTER 3.
WRITE PRINT-REC FROM HD3 AFTER 3.
WRITE PRINT-REC FROM HD2 AFTER 3.
WRITE PRINT-REC FROM HD4 AFTER 3.
WRITE PRINT-REC FROM HD2 AFTER 3.
PROCESS-PARA.
ADD WORKING-REGULAR-HOURS WORKING-OVERTIME-HOURS GIVING
TOTAL-HOURS-WORKED.
ADD 1 TO SNO.
ADD 1 TO NUMBER-OF-EMPLOYEES.
MOVE SNO TO W-SNO.
MOVE WORKING-NAME TO W-NAME.
MOVE WORKING-REGULAR-HOURS TO W-RHOURS.
MOVE WORKING-OVERTIME-HOURS TO W-OHOURS.
MOVE TOTAL-HOURS-WORKED TO W-THOURS.
WRITE PRINT-REC FROM HD5.
PERFORM READ-PARA.

CLOSE-PARA.
CLOSE IN-FILE.
CLOSE PRINT-FILE.

```

WORKING TIME LIST

EMP NO	NAME	REGULAR HOURS WORKED	OVERTIME HOURS WORKED	TOTAL HOURS WORKED
001	ABCDE	85	63	148
002	GHIJKL	85	64	149
003	MNOPQR	85	46	131
004	STUVW	89	56	145
005	XYZ	78	54	132
006	MNOPQR	78	54	132
007	CDEFGH	45	62	107

NUMBER OF EMPLOYEE IS 007



## PROGRAMMING IN FORTRAN

The name FORTRAN is a result of compressing the two words are FORMula TRANslation. FORTRAN began in 1954 a simpler more restricted language but after revision in 1958 and again in 1966 it became as FORTRAN IV, the most Common General purpose vehicle for data processing and numerical calculation in science and engineering. Many special purpose languages have sprung up ideally suited to a bewildering variety of tasks, but none presents any particular difficulty in learning after FORTRAN.

Experience in the use of FORTRAN brought an awareness of its shortcomings, particularly in relation to its handling of text, and because its control statement did not lend themselves very well to present ideas of well structured programs so in 1977 an updated version of FORTRAN IV incorporating widely adopted good features was announced and was formally standardised in 1978 by ANSI (ANSI standard x.3.9 1978) This version is popularly known as FORTRAN 77.

### Characters used in FORTRAN.

The following are the set of allowed characters in FORTRAN.

- i. Letters . : All 26 capital letters of English.
- ii. Decimal digits from 0 to 9
- iii. Special characters.  
+ - \* / ) % , ( = and blank



ank is a valid special character and is indicated by the  
ol  $\$$ . The colon character : is allowed in FORTRAN 77

### FORTRAN constants.

A number which does not change during the execution  
of a program is called a FORTRAN constant. A fortran variable,  
is the name given to a memory box in which data is stored.  
is a variable name may be assigned different values during  
execution of a program.

Fortran constant may be classified as:

Integer constants, and

) Real constants.

### FORTRAN Operations.

There are three types of Fortran operations illustrated  
under:

Arithmetic Operations		Illustration	
Symbols	For	Arithmetic	Fortran Equivalent.
	Exponentiation	$x^a$	X <del>^</del> A
	Multiplication	xy	X * Y
	Division	A/B	A/B
	Addition	a+b	A+B
	Substraction	x-5	X-5

## 2 Relational Operations

- .GT. Greater than(>)
- .GE. Greater than or equal to
- .LT. Less than(<)
- .LE. Less than or equal to ( ≤ )
- .EQ. Equal to
- .NE. Not equal to

## 3 Logical Operations

- .NOT. Not
- .AND. and
- .OR. or

Insequencing we determine an appropriate order for a series of jobs to be done on a finite number of service facilities in some preassigned order so as to optimise the total involved cost or (time) A particular situation may correspond to an industry producing a number of products each of which are to be processed through different machines a finite number of times.

SEQUENCING PROBLEM FOR M JOBS IN 2 MACHINES

let there are n jobs each of which is to be processed through two machines A and B in order ~~AB~~ i.e each job will go to machine M1 first then M2.

JOB 1 2 3 -----M=N/2

Machine A: P(1) p(2) P(3)-----P(N/2)

Machine B; P(N/2+1) P(N/2+2)-----P(N)

ALGORITHM-

STEP 1-Here we use one dimension arrays P(30) for time taken by the jobs on machine A & B . In array P(30) first half is the time taken by the job on machine A& second half is the time taken by job on machine B.respectively. In array Q(30) the value of P(30) are copied.

A(30) contains the value of time in order in which sequence is from MQ(15) contains the order of jobs

U(15) contains the time at which job enters machine A

V(15) contains time at which job comes out of machine A

W(15) contains time at which job enters B

X(15) contains time at which job comes out of machine B

STEP- 2 Read an N, Read P(I) varying I from 1 to N.

STEP- 3 Equate the array Q(I) and P(I).

STEP- 4 G=1000 comparing with G find the lowest value of P(I) and replace G= P(I) and KM=I ( the place at which the lowest value occurred.

STEP- 5 If KM is less than N/2 then replace the value in MQ(15) by KM from the beginning & P(KM)= P(KM+M)= 2000.

STEP- 6 If KM is greater than N/2 than replace the number KM-M in MQ(15) from the end ie MQ( M-(K-1) )= KM-M where K is initialized to 1. Replace P(KM) and P(KM-M) by 2000 & K=K+1.

STEP- 7 Repeat step 4,5,6 N/2 times.



- STEP- 8 Fill the array ~~in~~ A(30) in order in which jobs occurred in array MQ(15) by replacing  $A(I) = Q(MQ(I))$  &  
 $A(I+M) = Q(NS+M)$   
Increasing I from 1 to N/2. 0)
- STEP- 9 Initialize arrays U(I), V(I), W(I), X(I) to zero.
- STEP- 10 Replace the value of V(I) by U(I)+A(I) & U(I)+1 by V(I) repeating the process N/2 times.
- STEP- 11  $W(1) = V(1)$  &  $X(1) = W(1) + A(M+1)$ .
- STEP- 12 If X(I-1) less than V(I),  $W(I) = V(I)$  &  $X(I) = W(I) + A(M+1)$ .
- STEP- 13 If X(I-1) greater than V(I) then  $W(I) = X(I-1)$  &  $X(I) = W(I) + A(M+1)$ .
- STEP- 14 Repeat step-12&13 varying I from 2 to N/2.
- STEP- 15  $Telaps = X(M)$ .
- STEP- 16 Write MQ(I) varying I from 1 to M write Telaps.
- STEP- 17 Write U(I), V(I), W(I), X(I) varying I from 1 to N/2.
- STEP- 18 Stop.

Flowchart program in fortran and output are given on ~~58-63~~ pages.

```
0001      DIMENSION P(30),MQ(15),U(15),V(15),W(15),X(15),A(30),G(30)
0002      READ (1,*) N
0003      M = N/2
0004      READ (1,*) (P(I),I = 1,N)
0005      K = 1
0006      TELAPS = 0.0
0007      DO 25 I = 1,N
0008      Q(I) = P(I)
0009 25      CONTINUE
0010      DO 70 J = 1,M
0011      KM = 0
0012      G = 1000.0
0013      DO 50 I = 1,N
0014      IF (P(I) .GT. G) GOTO 50
0015      G = P(I)
0016      KM = I
0017 50      CONTINUE
0018      IF (KM .GT. M) GOTO 60
0019      MQ(J) = KM
0020      P(KM) = 2000.0
0021      P(KM+M) = 2000.0
0022      GOTO 70
0023 60      L = M-(K-1)
0024      MQ(L) = KM-M
0025      P(KM) = 2000.0
0026      KM1 = KM-M
0027      P(KM1) = 2000.0
0028      K = K+1
0029 70      CONTINUE
0030      DO 80 I = 1,M
0031      NS = MQ(I)
0032      A(I) = Q(NS)
0033      IA = I+M
0034      IP = NS+M
0035      A(IA) = Q(IP)
0036 80      CONTINUE
0037      DO 90 I = 1,M
0038      U(I) = 0.0
0039      V(I) = 0.0
0040      W(I) = 0.0
0041      X(I) = 0.0
0042 90      CONTINUE
0043      DO 100 I = 1,M
0044      V(I) = U(I)+A(I)
0045      IF (I.EQ.M) GOTO 100
0046      IQ = I+1
0047      U(IQ) = V(I)
0048 100     CONTINUE
0049      W(1) = V(1)
0050      M2 = M+1
0051      X(1) = W(1) + A(M2)
0052      DO 110 I = 2,M
0053      I1 = I-1
0054      IF (X(I1) .LT. V(I)) GOTO 10
0055      W(I) = X(I1)
0056      M3 = M+I
0057      X(I) = W(I)+A(M3)
0058      GOTO 110
```

```
00059 10      W(I) = V(I)
00060      M3 = M + I
00061      X(I) = W(I) + A(M3)
00062 110     CONTINUE
00063      TELAPS = X(M)
00064      WRITE (1,*) (MQ(I) , I = 1,M)
00065      WRITE (1,*) TELAPS
00066      DO 130 I = 1,M
00067      WRITE (1,*) U(I),V(I),W(I),X(I)
00068 130     CONTINUE
00069      STOP
00070      END
```



Therefore  $X_2$  can replace  $X_0$  if  $f(X_2)$  has the same sign as  $f(X_0)$ ; otherwise  $X_2$  can replace  $X_1$ . The procedure can then be repeated to find better and better estimates until the program is satisfied with them.

It is desired that this should be made into a computer program. The steps in the program will be :

- i) Establish initial guesses  $X_0$  and  $X_1$
- ii) Calculate an improved guess  $X_2$
- iii) Replace either  $X_0$  or  $X_1$  by  $X_2$
- iv) Return to step (ii) if further improvement is desired.

The process of translating this procedure into a FORTRAN Program begins with an elaboration of the details.

1. Establish the initial guess.

It is apparently desirable that  $X_0$  and  $X_1$  should be chosen so that  $f(X_0)$  and  $f(X_1)$  are of opposite sign. The program could interact with the user who would provide values of  $X_0$  and  $X_1$  until this condition is met.

It is necessary to decide when  $f(X_0)$  and  $f(X_1)$  are of opposite sign. Casting around for a means of doing this, the SIGN function suggests itself. The relational expression.

$$f(X_0) \neq \text{EQ} \cdot \text{SIGN} (f(X_0) , f(X_1) )$$

is . TRUE , when the signs are the same and this will be used to return to the user for the next guess.

Calculate an improved guess  $X_2$ .

This simply follows the formula

$$X_2 = X_0 - \frac{X_0 - X_1}{f(X_0) - f(X_1)} f(X_0)$$

Replace either  $X_0$  or  $X_1$  by  $X_2$ .

To decide which of  $X_0$  or  $X_1$  is to be replaced, the sign of  $f(X_0)$  is compared to the sign of  $f(X_1)$ . As in step (1), this uses a relational expression.

$$f(X_2) .EQ. \text{SIGN} ( f(X_2), f(X_0) )$$

which is . TRUE. When the signs are the same, in which case  $X_2$  replaces  $X_0$ . Otherwise  $X_2$  replaces  $X_1$ , and similarly  $f(X_2)$  replaces either  $f(X_0)$  or  $f(X_1)$ .

Is further improvement desired ?

Such a simple question unfortunately has a complicated answer if it is done properly. The usual procedure is to see how much of a change is represented by the new estimate and stop if it is small enough. The complications arise in deciding what is small enough. When a new value  $X_2$  has been calculated the absolute value of the change is

$$\Delta = |X_2 - X_0| \quad \text{or} \quad \Delta = |X_2 - X_1|$$

depending on which side is chosen. Stopping of the process could be based on this. For a smallish root, less than 1 say, it might be sensible to say

IF (  $\Delta$  .GT.  $1E-5$  ) go back  
and the change of  $10^{-5}$  probably implies an error around  $10^{-4}$  in the answer.

It is better to normalize the step size as, for example,

IF ( ABS (  $\Delta / X_2$  ) .GT.  $1E-5$  ) go back

except that here the opposite problem could arise for tiny roots. As a compromise use.

IF (  $\Delta$  .GT.  $1E-5$  ) go back

if the absolute value of the root  $X$  is apparently less than 1 ( $X_2$  is the best guess for it ) and

IF ( ABS (  $\Delta / X_2$  ) .GT.  $1E-5$  ) go back

if the absolute value of  $X_2$  is greater than 1.

Coding the False Position algorithm into FORTRAN. As a first step, suitable variable names are chosen for the important

quantities . LET

XO for  $X_0$

X1 for  $X_1$

X2 for  $X_2$

FXO for  $f(X_0)$



FX1            for     $f(x_1)$   
FX2            for     $f(x_2)$   
D                for     $\Delta$

Suppose the problem is

$$f(x) = x^3 - 7.8 x^2 + 18.5 x - 9.1$$

Flow chart is shown as in fig. (3.3)

Program listing with output result is also enclosed.

-----

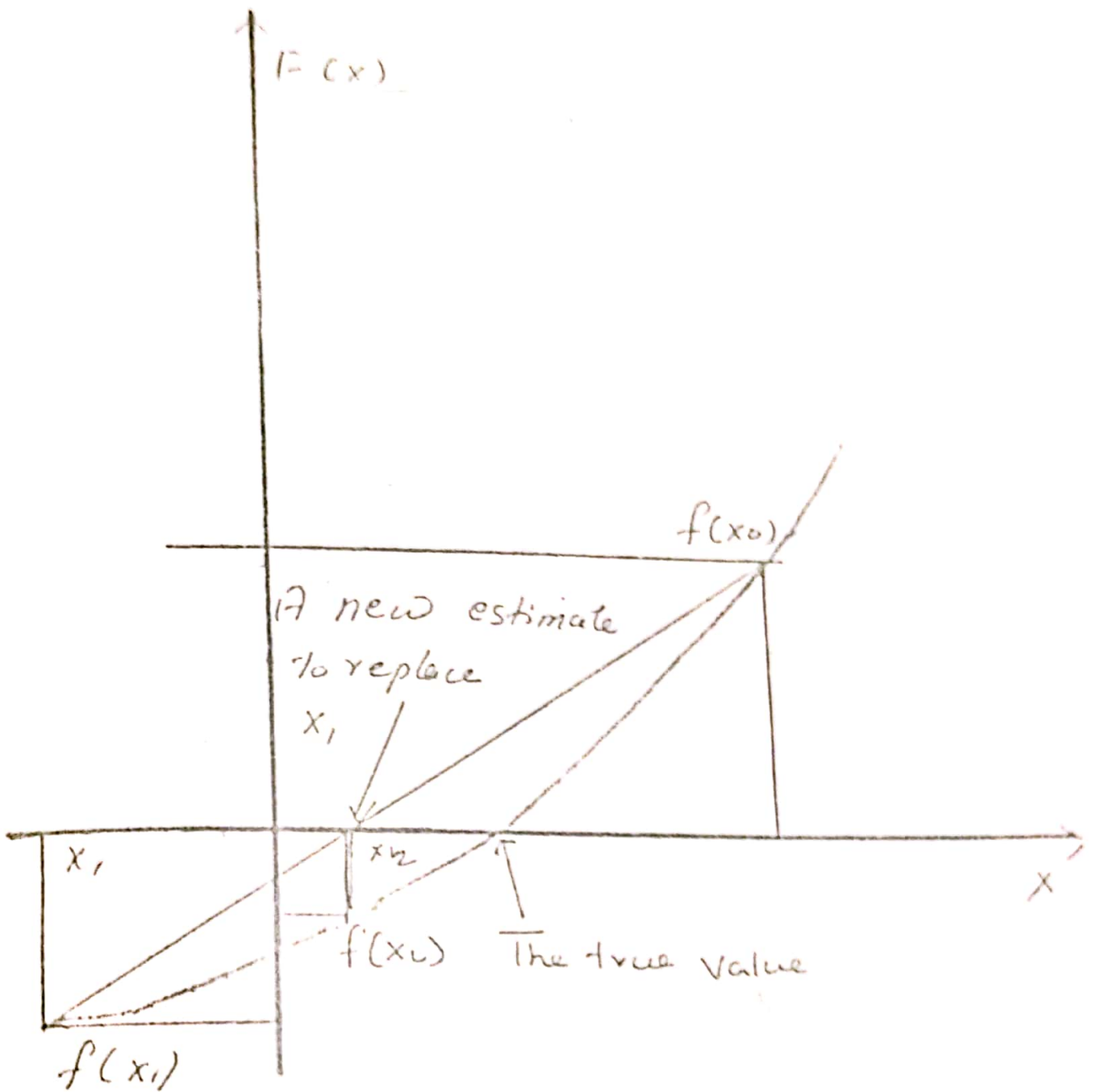


Fig 3.2 THE METHOD OF FALSE POSITION





S.A.T.1 COMPUTER CENTRE VICTORIA

06/17/87

IVE TWO BULBS 3E J  
J,FX0 ARE 2.50000 4.55300  
1,FX1 ARE 2.50000 4.32300  
RY AGAIN  
J,FX0 ARE 2.00000 5.59300  
1,FX1 ARE 2.00000 5.34300  
RY AGAIN  
J,FX0 ARE 1.90000 4.76300  
1,FX1 ARE 1.90000 4.75100  
RY AGAIN  
J,FX0 ARE 4.00000 4.62300  
1,FX1 ARE 4.00000 10.52001  
RY AGAIN  
J,FX0 ARE 1.30000 16.79601  
1,FX1 ARE 1.30000 3.96500  
RY AGAIN  
J,FX0 ARE 0.50000 4.35500  
1,FX1 ARE 0.50000 -1.67500  
2,FX2 ARE 1.00000 2.60300  
2 REPLACES X0  
ELTA= 1.30000  
2,FX2 ARE 0.69571 0.33386  
2 REPLACES X0  
ELTA= 0.30409  
2,FX2 ARE 0.66335 0.03159  
2 REPLACES X0  
ELTA= 0.03256  
2,FX2 ARE 0.65032 0.00290  
2 REPLACES X0  
ELTA= 0.00332  
2,FX2 ARE 0.66035 0.00026  
2 REPLACES X0  
ELTA= 0.00025  
2,FX2 ARE 0.65002 0.00002  
2 REPLACES X0  
ELTA= 0.00033  
2,FX2 ARE 0.66032 0.00000  
2 REPLACES X0  
ELTA= 0.00000  
UITABLE ROOT.66001950

## PROGRAMMING IN BASIC

### HISTORY OF BASIC

BASIC was first invented by Professors J.G. Kemeny and T.E. Kurtz of Dartmouth College, New Hampshire, U.S.A., as a language for beginners and was implemented in 1964. BASIC stands for Beginner's All-purpose Symbolic Instruction Code, it is an extremely powerful and useful language.

The BASIC language was designed to be conversational right from the start. This can put the programmer or user into direct communication with the computer, usually through a teletype terminal. In BASIC, the instructional statements are almost similar to normal algebra and therefore it is easy to understand and quick to learn. BASIC contains a relatively less number of statements and its grammar is simple. It also has a set of matrix operations that are more powerful than those available in FORTRAN.

The important features of BASIC can be summarized as under:

1. Suitable for conversational programming,
2. Facility for manipulation of character strings,
3. Dynamic program debugging
4. Ability to carry out arithmetic operations on matrices,
5. Provision for filing of programs and data,
6. Simplified grammar,

Facility for allowing more sophisticated formats for variables.

Facility for real-time execution and task scheduling,

Adaptability to both mathematical and business problems, and

Small interpreters and compilers are needed, making BASIC particularly suitable for microprocessor-based systems in which memory is limited.

### CHARACTER SET

BASIC has the same 51 character as in FORTRAN. The exponential symbol in BASIC vary with FORTRAN, in BASIC it is denoted as .

### CONSTANTS

BASIC deals with two types of constants :

- i. Numeric constants
- ii. String constants.

### NUMERIC CONSTANTS

The numbers are called numeric constants

#### Rules:

1. Numeric constants are numeric digits containing up to a maximum of 8 digits.
2. Commas are not allowed in constant.
3. The negative sign with negative number while + sign is optional.



4. The length of the exponent should not be more than two digits and it can contain - or + sign. + sign is optional.
5. BASIC does not make any distinction between integer and fractional numbers.

### STRING CONSTANTS

A string constant is a sequence of valid BASIC characters enclosed between quote mark. The quote mark do not form a part of string. String constt. represents non numeric information such as names, address, days, months, Years etc.

### VARIABLES

The quantity which changes during the execution of programme is called variable. One of the important task of preparing a computer programme is the selection of appropriate variables and putting their data structure. BASIC has two types of variables. They are :

- i. Numeric variable,
- ii. String variable.

#### Numeric variable

Numerical variable create the location in computer memory for storing numeric constant. Variables or data names or identifier which are used for numeric constt are called numeric variable.

## STRING Variable

The variable which locate the position for string constant are called string variables. These are named by numerical variable followed by \$ , ( Dollar sign ).

## ARITHMETIC EXPRESSIONS

A BASIC system uses the following five arithmetic operators:

1. + for addition
2. - for subtraction
3. \* for multiplication
4. / for division
5. for exponentiation

## HIERARCHY OF OPERATIONS

During the evaluation of expressions, the operations are executed one after the other and, in doing so, the computer assigns an order of priority to operators. The operations of high priority are performed before those of low priority. The order of heirarchy is:

Operation	Order of priority
( ) quantities inside the brackets	1
exponentiation	2

/ and *	division and multiplication	3
+ and -	addition and subtraction	4

---

### RULES OF ARITHMETIC

While writing expressions the following rules must be obeyed:

1. Two operators should never appear together,
2. No attempt should be made to raise zero to a negative power.
3. A negative value should not be raised to real number (fractional number),
4. Denominator of an expression should never be zero.
5. String constants or string variables are not allowed in expressions.
6. When variables are used in an expression we should make sure that they are assigned values before attempting to evaluate the expression.
7. Every left bracket must be matched by a right bracket.

Any violation of these rules will be a syntax error and a corresponding error message will be provided by the computer.

### LOGICAL EXPRESSIONS

#### GENERAL FORMAT

Constant or Variable or Expression	Relational operator	Constant or Variable or Expression
--	------------------------	--



## LIBRARY FUNCTIONS

Java provides some built-in function or Library functions

Function	Meaning
<del>Math.abs(x)</del>	Absolute value of x
<del>Math.atan(x)</del>	Arctangent of x(x in radians)
<del>Math.cos(x)</del>	Cosine of x(x in radians)
<del>Math.E</del>	$e^x$ the base of natural logarithm
<del>Math.floor(x)</del>	Largest integer not greater than x
<del>Math.log(x)</del>	Natural logarithm of x, where $x > 0$
<del>Math.random()</del>	Generates random number between 0 and 1
<del>Math.sign(x)</del>	Sign of x.
<del>Math.sin(x)</del>	Sine of x(x in radians)
<del>Math.sqrt(x)</del>	Square root of x, $x \geq 0$
<del>Math spaces(x)</del>	Carriage control to provide x spaces
<del>Math.tan(x)</del>	Tangent of x(x in radians)

## 1 - LINEAR PROGRAMMING PROBLEM

Many business and economic situations are concerned with a problem of planning activity. In each case there are limited resources at our disposal and our problem is to make such a use of these resources so as to yield the maximum production or to minimize the cost of production or to give maximum profit etc. Such problems are referred to as the problem of constrained optimisation. Linear programming is a technique for determining an optimum schedule of interdependent activities in view of the available resources.

LPP are mathematically formulated then their solution can be obtained by simplex method. Simplex method is an iterative procedure which either solves a LPP in a finite number of steps or gives an indication that there is an unbounded solution to the LPP.

### GENERAL LPP

Let  $Z$  be a linear functional defined by  
$$Z = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$$

Where  $c_j$ 's are constants let  $a_{ij}$  be an  $m \times n$  real matrix and let  $b_1, b_2, \dots, b_m$  be constants such that

$$a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n \leq b_1$$

$$a_{m1} x_1 + a_{m2} x_2 + \dots + a_{mn} x_n \leq b_m$$

$$\text{and } x_j \geq 0 \text{ for } j = 1, 2, \dots, n$$

The problem of determining  $n$  tuples  $(x_1, \dots, x_n)$  which makes  $Z$  a minimum or maximum and which satisfies  $b$  and  $c$  is called general LPP.

### ALGORITHM:

STEP 1 We use two dimensional array  $A (M, N)$  to represent coefficients of unknowns  $B(M)$  for constants,  $D(M)$  for basic variables,  $C(N)$  for objective function which is to be optimized  $L(N)$  for net evaluation which is calculated to determine the departing column.  $M$  is given value which is greater than total number of equations and  $N = 2M$ .

Step 1: The number of equations and G are given values as follows. The value of Q or R according to the value of G is calculated or minimize respectively.

Step 2: The values of  $L(1), L(2), L(3), \dots, L(N)$  are given values as follows.

Step 3: The value of  $L(1)$  is printed. It is the value of  $L(1)$ .

Step 4: The value of  $L(1)$  is printed and value of  $L(1)$  is calculated as follows.

$$L(1) = L(1) + L(2) \\ L(2) = L(2) + L(3) \\ \dots \\ L(N) = L(N) + L(N+1)$$

Step 5: The value of  $L(1)$  is printed for  $L(1)$  varying I from 1 to C. The value of  $L(1)$  is calculated by calculating value of  $L(1)$ .

Step 6: The value of  $L(1)$  is calculated by formula  $L(1) = L(1) + L(2)$ .

Step 7: The value of  $L(1)$  is calculated by the program field specification.

Step 8: The value of  $L(1)$  is calculated by a value such that it is less than or equal to the value of  $L(N)$  and the value of  $L(N)$  is the value of  $L(N)$  which is occur. R represent the value of  $L(N)$ . If all the values of  $L(N)$  are less than or equal to the value of  $L(N)$  then the solution has reached and the value of  $L(N)$  is calculated.

Step 9: If the value of  $L(N)$  is greater than zero then the value of  $L(N)$  is calculated. For this we check whether the value of  $L(N)$  is less than or equal to zero. If they are then "UNBOUNDED SOLUTION" is calculated.

Step 10: If the value of  $L(N)$  are less than or equal to zero then the value of  $L(N)$  is calculated. The value of  $L(N)$  is assigned a value of  $L(N)$ . The value of  $L(N)$  is calculated varying I from 1 to C and the value of I is found and the row is calculated by the value of I. The value of the corresponding variable is changed to R. The value of  $L(N)$  is R.



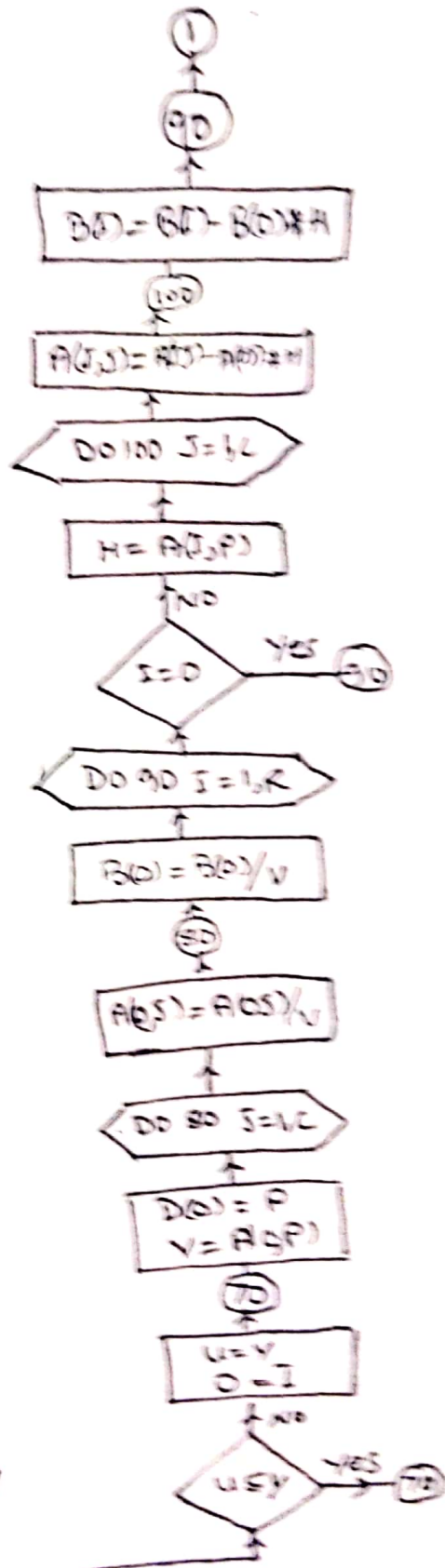
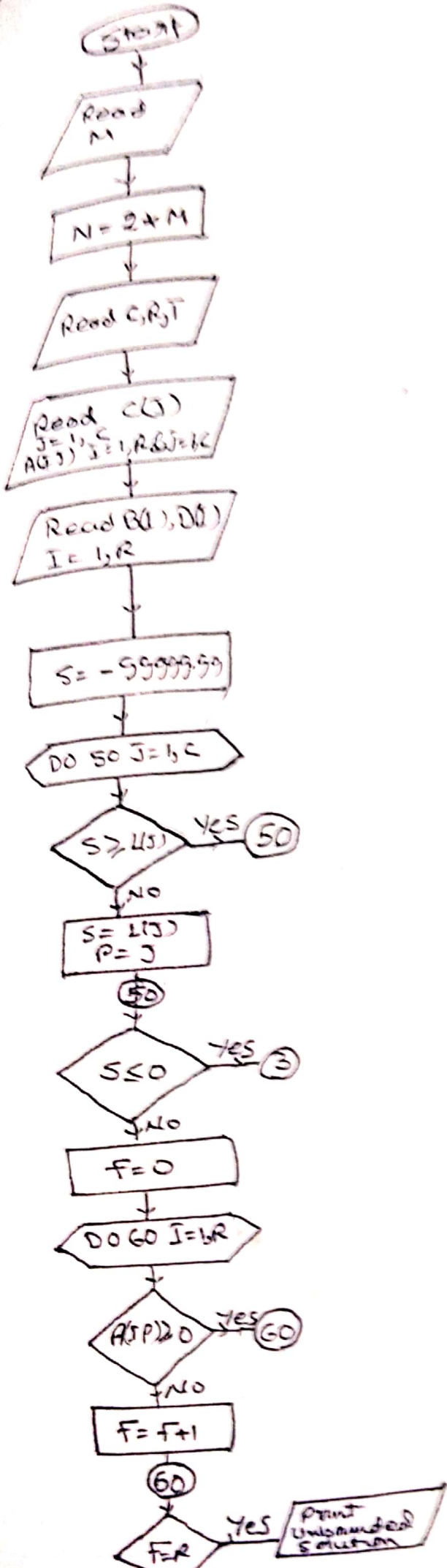
(3)

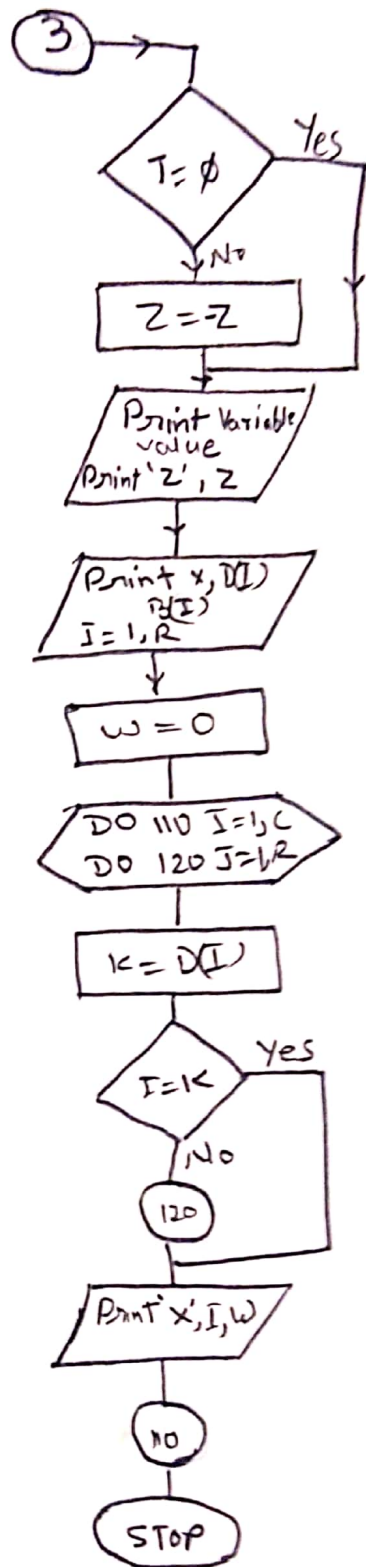
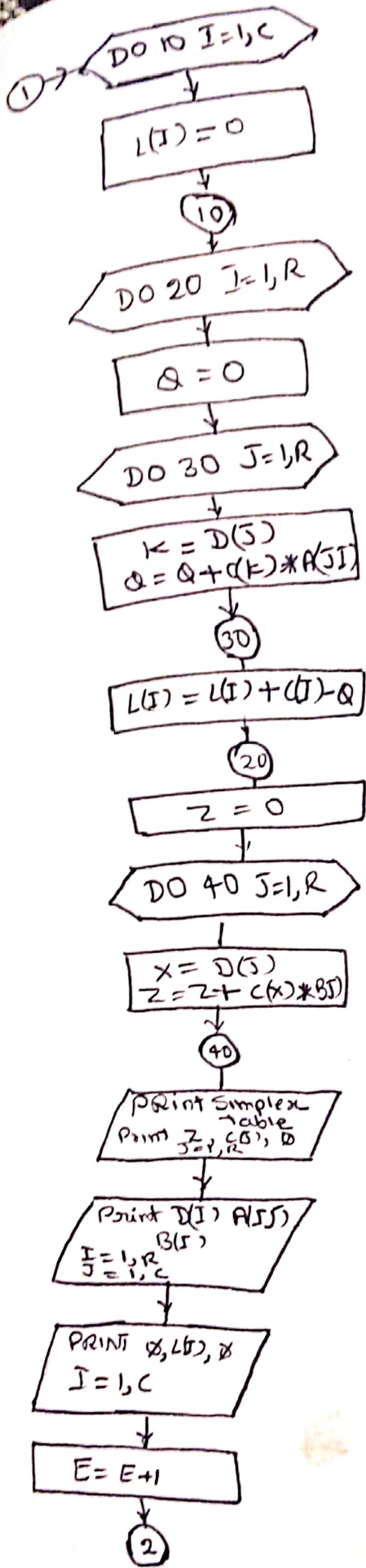
2

- STEP- 11 V is given value  $A(0,P)$  we divide the whole row 0 & B(0) by V.
- STEP- 12 For the rows other than 0 we make the changes as follows varying I from 1 to R
- $$A(I,J) = A(I,J) - A(0,J) * A(I,P)$$
- varying J from 1 to C  
and  $B(I) = B(I) - B(0) * A(I,P)$
- STEP- 13 Increase the value of E by 1 and repeat from step-5 till optimal solution is reached.
- STEP- 14 If  $T=1$  then  $Z = -Z$  otherwise the value of Z, variables and values are printed.
- STEP- 15 End.

Flowchart, program in Basic and output are given on ~~82-92~~ page.

\*\*\*







```

10 REM BASIC PROGRAM FOR GAUSS-JORDAN METHOD
20 INPUT M
30 N=2*M
40 DIM K(M,N),Z(M),X(M),Y(M)
50 INPUT "INPUT A(I,J)";A(I,J)
60 REM F=NO. OF ROWS, C=NO. OF COLUMNS
70 INPUT "INPUT F";F
80 FOR J=1 TO N
90 INPUT "INPUT C(J)";C(J)
100 NEXT J
110 FOR I=1 TO F
120 FOR J=1 TO N
130 INPUT "INPUT A(I,J)";A(I,J)
140 NEXT J
150 NEXT I
160 REM T=0 FOR ROW 1 FOR ROW
170 FOR I=1 TO F
180 INPUT "INPUT T(I)";T(I)
190 NEXT I
200 FOR I=1 TO F
210 INPUT "INPUT S(I)";S(I)
220 NEXT I
230 I=1
240 GOSUB 300
250 REM DETERMINATION OF PIVOT
260 S=99999.99
270 FOR J=1 TO N
280 IF A(I,J)>S THEN GOTO 290
290 S=A(I,J)
300 P=J
310 NEXT J
320 IF S=0 THEN GOTO 330
330 F=0
340 FOR I=1 TO F
350 IF A(I,P)≠0 THEN GOTO 360
360 F=F+1
370 NEXT I
380 IF F=0 THEN GOTO 390
390 REM DETERMINATION OF ROW
400 U=99999.99
410 FOR I=1 TO F
420 IF A(I,P)>U THEN GOTO 430
430 U=A(I,P)
440 IF U<0 THEN GOTO 450
450 U=Y
460 O=I
470 NEXT I
480 D(O)=F
490 V=A(O,P)
500 FOR J=1 TO N
510 A(O,J)=A(O,J)/V
520 NEXT J
530 B(O)=B(O)/V
540 FOR I=1 TO F
550 IF I=O THEN GOTO 560
560 W=A(I,P)
570 FOR J=1 TO N
580 A(I,J)=A(I,J)-W*A(O,J)+W*B(O)

```

```
590 NEXT J
600 B(I)=B(I)-B(O)*H
610 NEXT I
620 REM
630 E=E+1
640 GOTO 240
650 IF T=0 THEN 670
660 Z=-Z
670 LPRINT "VARIABLE", "VALUE"
680 LPRINT "Z",
690 LPRINT Z
700 FOR I=1 TO R
710 LPRINT "X";
720 LPRINT D(I), B(I)
730 NEXT I
740 W=0.0
750 FOR I= 1 TO C
760 FOR J=1 TO R
770 K=D(J)
780 IF I=K THEN 820
790 NEXT J
800 LPRINT "X"; I,
810 LPRINT W
820 NEXT I
830 GOTO 1220
840 LPRINT "UNBOUNDED SOLUTION"
850 GOTO 1220
860 REM CAL OF L(J)
870 FOR I=1 TO C
880 L(I)=0
890 NEXT I
900 FOR I=1 TO C
910 Q=0
920 FOR J=1 TO R
930 K=D(J)
940 Q=Q+C(K)*A(J, I)
950 NEXT J
960 L(I)=L(I)+C(I)-Q
970 NEXT I
980 Z=0
990 FOR J=1 TO R
1000 X=D(J)
1010 Z=Z+C(X)*B(J)
1020 NEXT J
1030 LPRINT "SIMPLEX TABLE"; E
1040 LPRINT USING "+####.##"; Z;
1050 FOR J=1 TO C
1060 LPRINT USING "+####.##"; C(J);
1070 NEXT J
1080 LPRINT "0"
1090 FOR I=1 TO R
1100 LPRINT USING "+####.##"; D(I);
1110 FOR J=1 TO C
1120 LPRINT USING "+####.##"; A(I, J);
1130 NEXT J
1140 LPRINT USING "+####.##"; B(I)
1150 NEXT I
1160 LPRINT TAB(3) "0";
```

```
1170 FOR I= 1 TO C
1180 LPRINT USING "+###.#";L(I);
1190 NEXT I
1200 LPRINT "0"
1210 RETURN
1220 END
```



DATA:

(1) 10  
3, 6  
0  
-1, 3, -2, 0, 0, 0  
3, -1, 2, 1, 0, 0  
-2, 4, 0, 0, 1, 0  
4, 3, 8, 0, 0, 1  
7, 12, 10,  
4, 5, 6

(2) 10  
3, 5  
0  
4, 10, 0, 0, 0  
2, 1, 1, 0, 0  
2, 5, 0, 1, 0  
2, 3, 0, 0, 1  
50, 100, 90,  
3, 4, 5

DATA:

(3) 10  
3, 6  
0  
107, 1, 2, 0, 0, 0  
4.66, .99, -2, 1, 0, 0  
16, .5, -6, 0, 2, 0  
3, -1, -1, 0, 0, 1  
2.33, 8, 0  
4, 5, 6

(4) 20  
3, 8  
0  
-4, -2, 0, 0, 0, -1000, -1000, -1000  
3, 1, -1, 0, 0, 1, 0, 0  
1, 1, 0, -1, 0, 0, 1, 0  
1, 2, 0, 0, -1, 0, 0, 1  
27, 21, 30  
6, 7, 8

SIMPLEX TABLE 1

+0.00	-1.00	+3.00	-2.00	+0.00	+0.00	+0.000	
+4.00	+3.00	-1.00	+2.00	+1.00	+0.00	+0.00	+7.00
+5.00	-2.00	+4.00	+0.00	+0.00	+1.00	+0.00	+12.00
+6.00	-4.00	+3.00	+8.00	+0.00	+0.00	+1.00	+10.00
0	-1.00	+3.00	-2.00	+0.00	+0.00	+0.000	

SIMPLEX TABLE 2

+9.00	-1.00	+3.00	-2.00	+0.00	+0.00	+0.000	
+4.00	+2.50	+0.00	+2.00	+1.00	+0.25	+0.00	+10.00
+2.00	-0.50	+1.00	+0.00	+0.00	+0.25	+0.00	+3.00
+6.00	-2.50	+0.00	+8.00	+0.00	-0.75	+1.00	+1.00
0	+0.50	+0.00	-2.00	+0.00	-0.75	+0.000	

SIMPLEX TABLE 3

+11.00	-1.00	+3.00	-2.00	+0.00	+0.00	+0.000	
+1.00	+1.00	+0.00	+0.80	+0.40	+0.10	+0.00	+4.00
+2.00	+0.00	+1.00	+0.40	+0.20	+0.30	+0.00	+5.00
+6.00	+0.00	+0.00	+10.00	+1.00	-0.50	+1.00	+11.00
0	+0.00	+0.00	-2.40	-0.20	-0.80	+0.000	

VARIABLE	VALUE
Z	-11
X 1	4
X 2	5
X 6	11
X 3	0
X 4	0
X 5	0



SIMPLEX TABLE 1

0	+0.00	+107.00	+1.00	+2.00	+0.00	+0.00	+0.000	
	+4.00	+4.66	+1.00	-2.00	+1.00	+0.00	+0.00	+2.33
	+5.00	+16.00	+0.50	-6.00	+0.00	+1.00	+0.00	+8.00
	+6.00	+3.00	-1.00	-1.00	+0.00	+0.00	+1.00	+0.00
0	+107.00	+1.00	+2.00	+0.00	+0.00	+0.00	+0.000	

SIMPLEX TABLE 2

0	+0.00	+107.00	+1.00	+2.00	+0.00	+0.00	+0.000	
	+4.00	+0.00	+2.55	-0.45	+1.00	+0.00	-1.55	+2.33
	+5.00	+0.00	+5.83	-0.67	+0.00	+1.00	-5.33	+8.00
	+1.00	+1.00	-0.33	-0.33	+0.00	+0.00	+0.33	+0.00
0	+0.00	+36.67	+37.67	+0.00	+0.00	-35.670		

UNBOUNDED SOLUTION

Subtotal for SAMPLES (TOTALS) 444,444

SAMPLE TABLE 0

1	2	3	4	5	6	7	8	9	10
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100

SAMPLE TABLE 1

1	2	3	4	5	6	7	8	9	10
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100

TABLE

100  
100  
100  
100  
100  
100  
100

TABLE

100  
100  
100  
100  
100  
100  
100





## 2 - REPLACEMENT PROBLEM

Replacement is concerned with situations that arise when some items such as machine, men, electric bulbs etc need replacement due to their deteriorating efficiency, failure or breakdown. The deteriorating efficiency or complete breakdown may be either gradual or sudden.

Following are the situations when the replacement of certain item is to be done:

- (1) Old item has failed and does not work at all or expected to fail shortly.
- (2) The old item has deteriorated and works badly or requires expensive maintenance.
- (3) Better design of equipment has been developed.

The problem is to decide the best policy to adopt with regard to replacement for eg.

- (a) It may be necessary to decide whether to wait for certain item to fail which may cause some loss or to replace earlier at the expense of higher cost of the item
- (b) The expensive item can be considered individually to decide whether to replace now or when to reconsider the item in question.
- (c) It is necessary to decide whether to replace by the same item or by a different type of item.

Replacement problems in general are of three types viz.

- (1) Replacement of item that deteriorate with time.
- (2) Replacement of item that breakdown completely.
- (3) Replacement of item that becomes out of date due to new developments.

## 2. REPLACEMENT OF ITEM THAT DETERIORATE WITH TIME

Generally the maintenance cost of certain items e.g. machine always increases gradually with time & stage comes when the maintenance cost becomes too large that it is better and economical to replace the item with a new one. There may be a number of choices and in each choice we make a comparison between various alternatives by considering cost due to waste scrap loss of output damage to equipment and safety risks etc.

Thus the average annual cost incurred on a machine per year during  $n$  years is given by-

$$T_n = \frac{1}{n} ( C - S(t) + \int_0^t f(t) dt )$$

Where --  $C$  = The capital cost of item.

$S(t)$  = Selling or scrap value after  $t$  years.

$f(t)$  = operating cost at time  $t$ .

$n$  = optimal replacement period.

In this type of problem item should be replaced when the average cost to date becomes equal to current maintenance cost thus we can decide the time at which it is profitable to replace an item.

### ALGORITHM

STEP- 1  $C_1$  = cost of new item.  
 $C(I)$  = cost of maintenance.  
 $S(I)$  = scrap value.

We take dimensions  $C(10)$  &  $S(10)$  for above cost.

STEP- 2 Read & print  $C_1$ ,  $K$ .  $K$  = no. of years.

STEP- 3  $C = 99999$  &  $TR = 0$

STEP- 4 Read  $C(I)$ ,  $S(I)$  varying  $I$  from 1 to  $K$ .

- STEP- 5     $TR = TR + C(I)$   
             $DC = C1 - S(I)$   
             $TC = TR + DC$   
             $AC = TC / I$
- STEP- 6    print I, TR, DC, TC, AC.
- STEP- 7    If AC less than or equal to C,  $C = AC$  &  $I1 = I$
- STEP- 8    Repeat step 5,6,7 K times.
- STEP- 9    Print I1, C.
- STEP- 10   Repeat step 1 to 9 K times.
- STEP- 11   Stop

Flowchart program in basic and output are given on ~~7,8,9,10~~<sup>9,10,11,12</sup> pages

REPLACEMENT OF ITEMS THAT FAIL COMPLETELY

GROUP REPLACEMENT POLICY

We always come across practical situation in real life where the failure of certain item causes all of a sudden breakdown of parts of the system. The failure of item may result in complete breakdown of the system in part from the actual cost of replacement of that part is a substantial cost involved in loss of production, loss of maintenance and other damages. However if it would have been known in advance as is when the item is going to fail it could have been replaced before its sudden breakdown. It is quite possible that in such a case the influence of indirect cost may be almost completely eliminated or reduced to great extent.

BENEFIT ANALYSIS:

Let us assume that there are  $N$  items in a system. Let  $P$  be the probability that an item which was new when placed in position for use fails during the period of life.

The no of items failed and replaced during the first month would be  $N \cdot P$ . If we denote this by  $N_1$  then the no of items failed and replaced during the second month would be

$$N_2 = N(1-P)P$$

In this  $N$  items are two period old and  $N_2$  are one period in general if

$$\begin{aligned} N_3 &= N(1-P)^2 P \\ P &= \text{probability of failure during time } t, \text{ then} \\ N_3 &= N(1-P)^2 P \end{aligned}$$

If we put  $t = 1, 2, 3$  then

$$N_1 = N \cdot P \quad N_2 = N(1-P)P \quad N_3 = N(1-P)^2 P$$

The total cost of replacement at the end of time  $t$  is

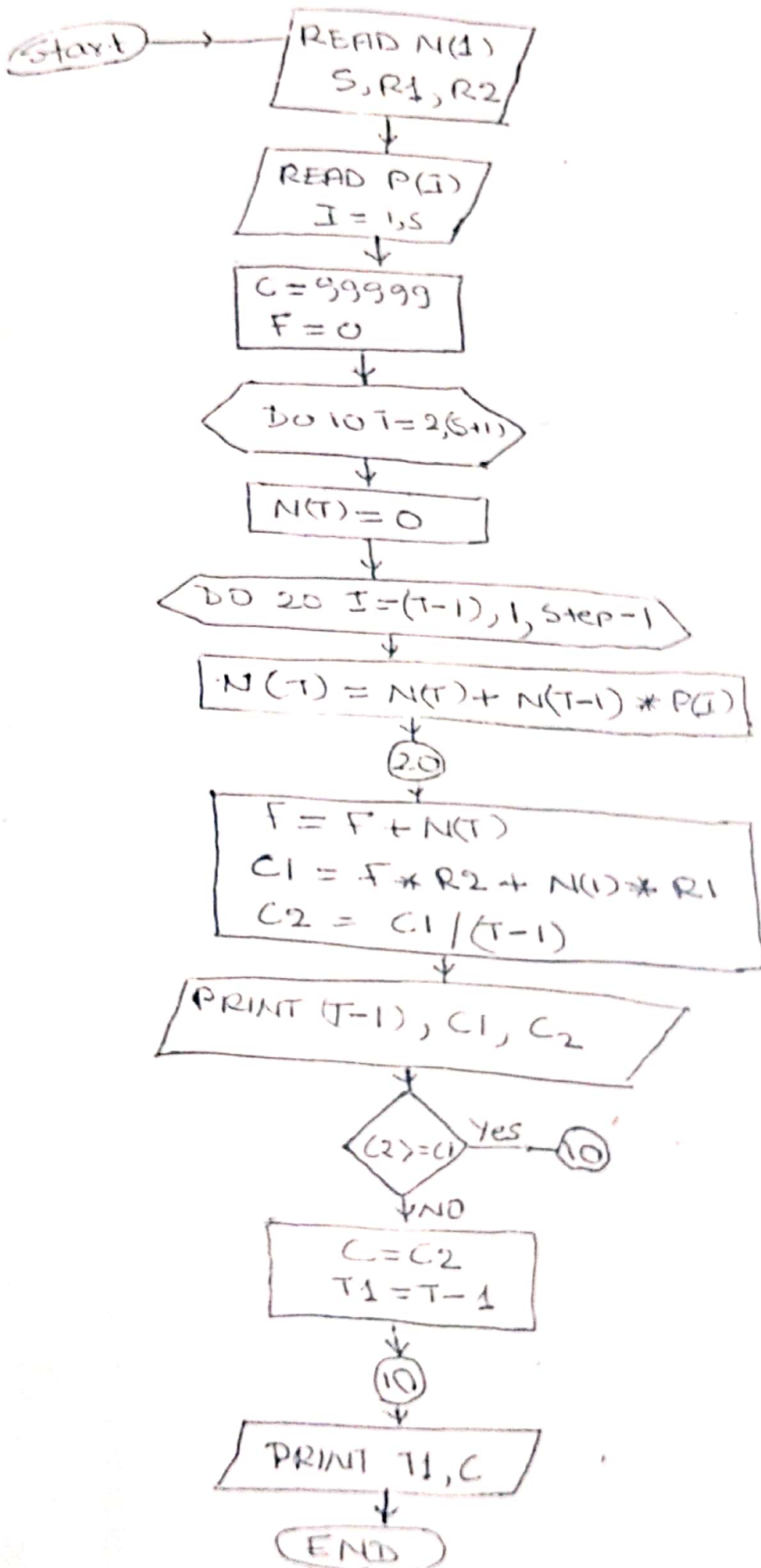
$$C_t = N_1 \cdot C + \text{cost of same replacement of } N \text{ items.}$$

$$N_1 = \text{total items replaced till the end of } t \text{ months.}$$





# FLOW CHART FOR GROUP REPLACEMENT POLICY



```

10 REM "SAMPLE REPLACEMENT"
20 REM "COST OF NEW ITEM=C1"
30 REM "COST OF MAINTENANCE=C(I)"
40 REM "SCRAPE VALUE=S(I)"
50 DIM C(10),S(10)
52 INPUT "C1=";C1
54 LPRINT "C1=PS ";C1
50 LET C=99999.C
70 LET TR=0
90 INPUT "K";K
92 LPRINT "NO OF YEARS=";K
100 FOR I = 1 TO K
110 INPUT C(I):INPUT S(I)
112 LPRINT "C(I)=";C(I)
114 LPRINT "S(I)=";S(I)
120 NEXT I
130 LPRINT "PERIOD ", "TOTAL C COST", "DEP COST", "TOTAL COST", "AVERAGE COST/Y"
135 FOR I = 1 TO K
140 TR = TR + C(I)
155 DC = C1 - S(I)
160 TC = TR + DC
170 AC = TC / I
180 LPRINT I, TR, DC, TC, AC
190 IF AC >= C THEN 220
200 LET C = AC
210 I = I
220 NEXT I
230 LPRINT
240 LPRINT "OPTIMAL REPLACEMENT PERIOD=";I
250 LPRINT
260 LPRINT "COST OF REPLACEMENT /YEAR=PS ";C
270 GOTO 50
280 END

```

3.4.1 COMPUTER CENTRE VIDISHA 09/23/87 24516 #4916

COMPILED=18 9

```
10 REM REPLACEMENT PROBLEM
20 DIM N(20),P(20)
30 LPRINT "TYPE IN NUMBER OF ORIGINAL ITEMS"
40 INPUT N(1)
50 READ S,R1,R2
60 FOR I=1 TO S
70 READ P(I)
80 NEXT I
90 LPRINT "PERIOD(MONTHS)","TOTAL COST","COST/MONTH"
100 C=99999.0
110 F=0
120 FOR T=2 TO (S+1)
130 N(T)=0
140 FOR I=T-1 TO 1 STEP -1
150 N(T)=N(T)+N(T-I)*P(I)
160 NEXT I
170 F=F+N(T)
180 C1=F*R2+N(1)*R1
190 C2=C1/(T-1)
200 LPRINT T-1,C1,C2
210 IF C2>= C THEN 240
220 C=C2
230 T1=T-1
240 NEXT T
250 LPRINT
260 LPRINT "OPTIMAL REPLACEMENT PERIOD =";T1;"MONTHS"
270 LPRINT
280 LPRINT "COST OF REPLACEMENT / MONTH = RS";C
290 DATA 11,5,1.23,0.1,0.3,0.5,0.7,1.0,1.5,2.0,1.5,1.1,0.3,0.5
300 END
```



S.A.T.I COMPUTER CENTRE VIDISHA 06/23/87

COMPANY-ID 5

TYPE IN NUMBER OF ORIGINAL ITEMS	TOTAL COST	COST/MONTH
PERIOD(MONTH)		
1	512.5	512.5
2	550.125	275.0625
3	613.3762	204.4587
4	703.2625	175.8156
5	835.3152	166.763
6	1032.354	172.0606
7	1322.825	186.1178
8	1526.275	190.7843
9	1721.065	191.2295
10	1923.345	190.3345
11	2075.658	188.6962

OPTIMAL REPLACEMENT PERIOD = 5 MONTHS

COST OF REPLACEMENT / MONTH = RS 190.703



5.

Write a computer program for the SOLUTION OF A QUADRATIC EQUATION.

Evaluation of one real root of the equation.

$$Ax^2 + Bx + C = 0$$

using the NEWTON-RAPHSON iteration method.

Problem Analysis:

Assume that

$$Y = f(x) = Ax^2 + Bx + C$$

Then the Newton-Raphson method provides the following recurrence relationship

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$f'(x) = \frac{dy}{dx}$$

If  $x_k$  is some guess answer, then  $x_{k+1}$  is called the improved estimate of the answer. The recurrence equation should be repeatedly evaluated till a desired value of  $x_{k+1}$  is reached.

Such recurrence relationships always suggest the use of self replacement technique in a loop to evaluate successively the value of  $x_{k+1}$ .

The function  $F(x)$  and  $F^{-1}(x)$  can be defined through BASIC functions as follows:

$$F(x) = FMY(X) = 2x^2 + 3x + 1$$

$$F^{-1}(x) = INV(X) = 2x + 1$$

Since the replacement technique is used to evaluate  $x_{k+1} = F(x_k)$  and  $x_k$  is used in the program to represent  $x_{k-1}$  and  $x_{k-2}$  respectively.

Let the program be  $x^2 - 20 = 0$

Here  $A=1$

$B=9$

$C=20$

Flow chart of fig. 2.2 illustrates the use of replacement model. The program listing with output results is enclosed.



The function  $f(x)$  and  $f^{-1}(x)$  can be defined through BASIC functions as follows:

$$f(x) = \text{FNY}(X) = Ax^2 + Bx + C$$

$$f^{-1}(x) = \text{FNY}(X) = 2Ax + B$$

Since the replacement technique is used to evaluate  $x_{k+1}$  and  $x_k$  are used in the program to represent  $x_k$  and  $x_{k+1}$  respectively.

Let the problem is  $x^2 - 9x + 20 = 0$

Here  $A=1$

$B=-9$

$C=20$

Flow chart of fig. 4.2 illustrate the use of replacement model. The program listing with output result is enclosed.

```

10 T.1. "NEWTON RAPHSON METHOD"
11 REM **THIS PROGRAM CALCULATES THE ROOTS OF QUAD. EQU.**
12 REM ** THE EQU. OF TYPE A*X^2 + B*X + C = 0 **
13 REM ** GENERAL PROGRAM **
14 REM ** DEFINITION OF FUNTION **
15 REM "DO YOU WANT TO CONT."
16 LPRINT "1-CONT."
17 LPRINT "2-EXIT"
18 INPUT "N";N
19 ON N GOTO 30,190
20 DEF FNY(X)=A*X*X+B*X+C
21 DEF FND(X)=2*A*X+B
22 LPRINT "VALUES OF CONSTANTS A,B,C"
23 LPRINT "A=";A
24 INPUT "A=";A
25 LPRINT "B=";B
26 INPUT "B=";B
27 LPRINT "C=";C
28 INPUT "C=";C
29 LPRINT "INITIAL ESTIMATE OF X"
30 INPUT X
31 LPRINT "INITIAL GUESS=";X
32 FOR I=1 TO 100
33 LET XD=X
34 Y=FNY(XD)
35 D=FND(XD)
36 X=XD-Y/D
37 LPRINT I,X,Y,D
38 IF ABS(X-XD)<.000001 THEN 170
39 NEXT I
40 LPRINT "ROOTS OF THE EQU.=";X
41 GOTO 10
42 ENL
    
```

INITIAL ESTIMATE OF X

INITIAL GUESS=	0		
1	2.222222	20	-9
2	3.306233	4.938272	-4.555555
3	3.798406	1.175079	-2.387533
4	3.971037	.2422333	-1.403137
5	3.999203	2.930232E-02	-1.357926
6	4	7.93457E-04	-1.301535
7	4	0	-1

ROOTS OF THE EQU. = 4  
INITIAL ESTIMATE OF X

INITIAL GUESS=	1		
1	2.714286	12	-7
2	3.537143	2.938774	-3.571428
3	3.883749	.6770935	-1.925714
4	3.989877	.1236286	-1.222502
5	3.999901	.0102272	-1.020247
6	3.999998	9.727478E-05	-1.000198
7	4	1.907349E-06	-1.000003
8	4	0	-1

ROOTS OF THE EQU. = 4  
INITIAL ESTIMATE OF X

INITIAL GUESS=	1		
1	2.714286	12	-7
2	3.537143	2.938774	-3.571428
3	3.883749	.6770935	-1.925714
4	3.989877	.1236286	-1.222502
5	3.999901	.0102272	-1.020247
6	3.999998	9.727478E-05	-1.000198
7	4	1.907349E-06	-1.000003
8	4	0	-1

ROOTS OF THE EQU. = 4  
INITIAL ESTIMATE OF X

INITIAL GUESS=	2		
1	3.2	6	-5
2	3.753846	1.439999	-2.6
3	3.959348	.3067474	-1.492309
4	3.993476	4.223150E-02	-1.081204
5	3.999998	1.525679E-03	-1.003047
6	4.000001	3.814697E-06	-1.000005
7	4.000001	0	-.9999971

ROOTS OF THE EQU. = 4.000001  
INITIAL ESTIMATE OF X

INITIAL GUESS=	1		
1	2.714286	12	-7
2	3.537143	2.938774	-3.571428
3	3.883749	.6770935	-1.925714
4	3.989877	.1236286	-1.222502
5	3.999901	.0102272	-1.020247
6	3.999998	9.727478E-05	-1.000198
7	4	1.907349E-06	-1.000003
8	4	0	-1

ROOTS OF THE EQU. = 4  
INITIAL ESTIMATE OF X

Handwritten text at the top of the page, possibly a title or introductory paragraph.

Handwritten text in the middle section, possibly a date or a specific reference.

Handwritten text in the lower middle section, possibly a list or detailed notes.

Handwritten text on the left side, possibly a signature or a small note.

Handwritten text below the signature area.

Vertical handwritten text, possibly a list of items or a column of data.

Handwritten text at the bottom of the page, possibly a footer or a concluding note.





Write a program to solve the given first order ordinary differential equation by Runge-Kutta method.

$$y' + y = 1, \quad y(0) = 0$$

sol

sol

Interval of  $x$  is from 0 to 1 with step size  $h = 0.25$

Solution

$$y' + y = 1$$

$$y' = 1 - y$$

Since  $y = 0$  at  $x = 0$ , the value of  $y$  is computed till

$$x = 1$$

$y = 0$  is the initial value of  $y$

$$y = 0$$

Now check the value of  $y$  at  $x = 1$

Write a program to solve the linear first order ordinary differential equation by predictor - corrector method.

$$Y^1(X) = \frac{dy}{dx} = X^2 Y$$

$$X=0$$

$$Y=1.0$$

interval = 0.1 each and calculate value up to  $x = 3.5$

Solution:

$$X_1 = 0 \quad Y_1 = 1.0 ,$$

$$H = 0.1$$

Since  $K = 35$  ( Since value is to be computed till  $X = 3.5$  )

IO = 40 maximum number of iteration

$$E_p = 0.00001$$

Flow chart is as shown in fig. (4.4 )

S.A.T. : COMPUTER CENTRE VIDISHA

05/16/87

DIFF. DIFF.

```
10 REM "FIRST ORDER DIFF. EQN."
20 REM "PREDICTOR AND CORRECTOR METHOD"
30 REM "NUMERICAL ANALYSIS"
40 DIM X(40), Y(40)
50 READ X1, Y1, H, K, IO, E
60 DATA 0, 1, 0.1, 35, 40, 0.00001
70 X(1) = X1
80 Y(1) = Y1
90 FOR N = 1 TO (K-1)
100 F1 = (X(N)^2) * Y(N)
110 YP = Y(N) + H * F1
120 Y(N+1) = X(N) + H
130 I = 1
140 F2 = (X(N+1)^2) * YP
150 YC = Y(N) + (H/2) * (F1 + F2)
160 IF (ABS(YC - YP) > E) THEN 240
170 Y(N+1) = YC
180 NEXT N
190 FOR N = 1 TO K
200 PRINT
210 PRINT N, X(N), Y(N)
220 NEXT N
230 GOTO 300
240 IF (I >= 10) THEN 280
250 YP = YC
260 I = I + 1
270 GOTO 140
280 PRINT
290 PRINT "FALLS TO CONVERGE"
300 END
```



5.

Motion of a body is described by differential equation as follow

$$\frac{dv}{dt} = (2000 - 2v) / (200 - t)$$

The body is at rest (i.e velocity  $V=0$ ) at time  $t=0$ .  
Write a program to compute the velocity of the body at one interval of 5 second in the first into minutes by Predictor corrector method.  
Providing check against endless cycle and print out intermediate computed value.

$V_1$  = Initial value of velocity ( =0 given )

$T_1$  = Initial Value of time ( = 0 given )

$H$  = Interval of time ( =5 second given )

$EO$  = Term for test of convergence

= 0.00001 (let)

$IO$  = Maximum number of iteration = 30 (days)

$K$  = No. of ordinates = 24 given

Since computation is to be done for 2 minute at interval of 5 seconds starting from 0 time.

Flow chart is as shown in fig( 4.5 ) . Program listing with output result is attached.

Motion of a body is described by differential equation as follow

$$\frac{dv}{dt} = (2000 - 2v) / (200 - t)$$

The body is at rest (i.e velocity  $V=0$ ) at time  $t=0$ .

Write a program to compute the velocity of the body at one interval of 5 second in the first into minutes by Predictor corrector method.

Providing check against endless cycle and print out intermediate computed value.

$V_1$  = Initial value of velocity ( =0 given )

$T_1$  = Initial Value of time ( = 0 given )

$H$  = Interval of time ( =5 second given )

$EO$  = Term for test of convergence  
= 0.00001 (let)

$IO$  = Maximum number of iteration = 30 (days)

$K$  = No. of ordinates = 24 given

Since computation is to be done for 2 minute at interval of 5 seconds starting from 0 time.

Flow chart is as shown in fig( 4.5 ) . Program listing with output result is attached.

```
10 REM "MOTION OF BODY"  
20 REM "PREDICTOR AND CORRECTOR METHOD"  
30 DIM V(40), T(40)  
40 READ V1, T1, ED, T0, K  
50 DATA 0, 5, .00001, 30, 24  
60 V(1) = V1  
70 T(1) = T1  
80 FOR N = 1 TO (K-1)  
90 F1 = (2000 - 2 * V(N)) / (200 - T(N) - 1))  
100 VP = V(N) + H * F1  
110 T(N+1) = T(N) + H  
120 I = 1  
130 F2 = (2000 - 2 * VP) / (200 - T(N+1))  
140 VC = V(N) + (H/2) * (F1 + F2)  
150 IF (ABS(VC - VP) > ED) THEN 190  
160 V(N+1) = VC  
170 PRINT F1, VP, T(N+1), VC, F2  
180 GOTO 230  
190 IF (I > 10) THEN 240  
200 VP = VC  
210 I = I + 1  
220 GOTO 130  
230 NEXT N  
240 PRINT "DOES NOT CONVERGE"  
250 GOTO 290  
260 FOR H = 1 TO K  
270 PRINT N, T(N), V(N)  
280 NEXT N  
290 END
```





8. Write a computer program for finding the standard deviation and coefficient of variation.

Algorithm

If  $x_1, x_2, \dots, x_i, \dots, x_n$  are the  $n$  values of a variate  $x$ , and  $f_1, f_2, \dots, f_i, \dots, f_n$  be the respective frequencies of  $x$  then mean is given by

$$A.M = \frac{f_1 x_1 + f_2 x_2 + \dots + f_n x_n}{N}$$

Where  $N =$  total frequency

and

$$S.D = \sqrt{\frac{1}{N} \sum f x^2 - \left(\frac{\sum f x}{N}\right)^2}$$

$$V = (S.D)^2$$

Notations :

$$X(I) = x_i$$

$$F(I) = f_i$$

$$Y2 = f_i x_i^2$$

$$S2 = \sum f_i x_i^2$$

$$A2 = (A1)^2$$

$$N = \sum f_i$$

$$X2 = x_i^2$$

$$S1 = \sum f x$$

$$A1 = \text{Mean}$$

$$C = \text{Coeff. Variation}$$

8. Write a computer program for finding the standard deviation and coefficient of variation.

Algorithm

If  $x_1, x_2, \dots, x_n$  are the  $n$  values of a variate  $x$ , and  $f_1, f_2, \dots, f_n$  be the respective frequencies of  $x$  then mean is given by

$$A.M = \frac{f_1 x_1 + f_2 x_2 + \dots + f_n x_n}{N}$$

Where  $N =$  total frequency

and

$$S.D = \sqrt{\frac{1}{N} \sum f x^2 - \left(\frac{\sum f x}{N}\right)^2}$$

$$V = (S.D)^2$$

Notations ;

$$X(I) = x_i$$

$$F(I) = f_i$$

$$Y2 = f_i x_i^2$$

$$S2 = \sum f_i x_i^2$$

$$A2 = (A1)^2$$

$$N = \sum f_i$$

$$X2 = x_i^2$$

$$S1 = \sum f x$$

$$A1 = \text{Mean}$$

$$C = \text{Coeff. Variation}$$

### Problem Testing

Find the Mean standard deviation, variation ant coffeif  
coefficient of variation for the following :

X : 5    15    25        35        45        55

F : 15    17    19        27        19        12

Flow chart for the problem is as shown in fig. ( 4.7 ).

### Problem Testing

Find the Mean standard deviation, variation ant coffefi  
coefficient of variation for the following :

X : 5    15    25    35    45    55

F : 15    17    19    27    19    12

Flow chart for the problem is as shown in fig. ( 4.7 ).



S.A.T.I COMPUTER CENTRE VIDYHA 06/02/87 VAR1 V4

```
10 KEY "STANDARD DEVIATION, MEAN, COEFF. OF VARIATION"
20 DIM X(20)
30 DIM F(20)
40 INPUT "K":K
50 FOR I=1 TO K
60 INPUT X(I):INPUT F(I)
70 LPRINT "X(I)";X(I),"F(I)";F(I)
80 NEXT I
90 N=0
100 FOR I=1 TO K
110 LET N=N+F(I)
120 NEXT I
130 S1=0
140 S2=0
150 FOR I=1 TO K
160 X1=F(I)*X(I)
170 X2=X(I)*X(I)
180 Y2=F(I)*X2
190 S1=S1+X1
200 S2=S2+Y2
210 LPRINT " " / "X1";X1/"X2";X2/"Y2";Y2
220 NEXT I
230 LPRINT "-----"
240 LPRINT "N";N,"S1";S1," " / "S2";S2
250 A1=S1/N
260 A2=A1*A1
270 S3=S2/N
280 V=(S3-A2)
290 S(I)=SQR(V)
300 C=S(I)/A1
310 LPRINT "-----"
320 LPRINT "S.D=";S(I),"VAR.=";V,"MEAN=";A1,"COEFF. VAR.=";C
330 END
```

X(1)	5	F(1)	15
X(1)	15	F(1)	17
X(1)	25	F(1)	19
X(1)	35	F(1)	27
X(1)	45	F(1)	19
X(1)	55	F(1)	12

X1	75	X2	25	Y2	375
X1	255	X2	225	Y2	3425
X1	475	X2	625	Y2	11675
X1	945	X2	1225	Y2	33075
X1	255	X2	2025	Y2	33475
X1	650	X2	3025	Y2	36300

N 109

S1 3265

S2 123425

S.D 15.4315      VAR. 237.0760      MEAN 29.95413      COEFF. VAR. .5163404

R E F E R E N C E S

1. Data Processing  
By- Martin M. Lipschutz &  
Seymour Lipschutz.
2. Computer Programming in COBOL  
By- V. Rajaraman & H.V. Sahasra Buddha.
3. Computer Programming in Fortran  
By- V. Rajaraman.
4. Computer Programming in Fortran  
By- K.D. Sharma
5. Programming in COBOL  
By- M.K.Ray & D.Gosh
6. Programming in Basic  
By- Martin Lipschutz & Seymour Lipschutz.