Unit - I

Computer program

It is a collection of instructions which performs a specific task when executed by a computer. From the program in its human-readable form of source code, a compiler (interpreter) can derive machine code – a form consisting of instructions that the computer can directly execute.

Q1. 1a. Quality of good program

1. **Clarity and simplicity of expression**
2. **Use of proper names for identifiers / variables**
3. **Comments**
4. **Indentation**

1b. Characteristics of good programming

1. **Flexibility – Without rewriting the entire program**
2. **User friendly – Easily understood by the beginner**
3. **Portability – Run on different OS**
4. **Reliability - Ability of program to do intended functions accurately**

1c. Problem solving process:

1. **Understanding the problem**
2. **Devising a plan**
3. **Executing the plan**
4. **Evaluation**

Q2. Problem solving methodology

1. **Problem definition (What are the method / logics we are going to solve the problem?)**
2. **Problem analysis (Sub program)**
3. **Design the problem (Algorithm and flowchart)**
4. **Coding (any languages)**
5. **Testing and debugging (finding and correcting the errors – Syntax , Logical, Runtime error)**
6. **Documentation (History of the program development)**
7. **Maintenance (Uninterrupted program execution)**

UNIT I

ALGORITHMIC PROBLEM SOLVING 9

1. Algorithms (Q3)
2. Building blocks of algorithms (statements, state, control flow, functions) (Q4)
3. Notation (pseudo code, flow chart, programming language) (Q5)
4. Algorithmic problem solving (Q6)
5. Simple strategies for developing algorithms (iteration, recursion) (Q7)

Illustrative problems:

1. Find minimum in a list
2. Insert a card in a list of sorted cards
3. Guess an integer number in a range
4. Towers of Hanoi

Q3. Algorithm

Founder – Abdullah Muhammad bin Musa al-Khwarizmi (Ninth century) - Persian scientist, astronomer and mathematician.

Definition

An algorithm is a finite set of instructions for accomplish a particular task.

It is a step by step procedure of solving a problem.

An algorithm is a collection of well-defined, unambiguous and effectively computable instructions, if execute it will return the proper output.

The time and space complexity associated with each algorithm.

For a single problem, more than one algorithm is possible. The algorithm that takes less time and less memory space is considered as the best algorithm.

User needs knowledge to write algorithm.

Characteristics of algorithms

It should be precise & unambiguous

It should ultimately terminate

It should be written in sequence manner

It looks like normal English

The output should obtain only after the algorithm terminates.

Qualities of good algorithms

Time:	Less time must be taken to execute the programs
Memory:	Less memory should be used for storing & execution of program
Accuracy:	Result should be more accurate
Sequence:	Algorithm should be in sequence order to execute them easily

Method for developing an algorithm

Define the problem:	Problem to solve in clear and concise terms
List the inputs:	Information needed to solve the problem
Describe the steps:	Needed to manipulate the inputs to produce the outputs
Start at a high level:	Refining the steps until they are effectively computable operations.
Test the algorithm:	Choose data sets verify the working of the algorithm

Advantages

It is a step-by-step rep. of a solution to a given problem, which is very easy to understand.

It easy to first develop an algorithm & then convert it into a flowchart & then into a computer program.

It is easy to debug as every step is got its own logical sequence.

Disadvantages

It is time consuming & cumbersome as an algorithm is developed first which is converted into Flowchart & then into a computer program.

Q4. Building blocks of algorithms (statements, state, control flow, functions)

Algorithms can be created using four basic building blocks. They are:

1. **Statements**
2. **State**
3. **Control flow**
4. **Functions**

1. Instructions / Statements (Q4 – 1)

The computer will run the codes in order, one line at a time from the top of the program to the bottom of the program. When an instruction is executed the execution control moves to the next immediate step. The sequence is exemplified by sequence of statements placed one after the other the one above or before another gets executed first. It will start at line 1, and then executed line 2 ten line 3 so on till it reaches the last line of the program.

In a computer statements might include some of the following actions.

Input data	-	information given to the program.
Process data	-	perform operation on a given input.
Output data	-	processed result

Two kinds of statements, they are

Simple statements: Assignment operator, comment and goto statement, function call and return

Compound statements: while-loop, do-loop, for-loop, if-else statement and switch statements

2. State (Q4 – 2)

Transition from one process to another process under specified condition with in a time is called state.

3. Control flow (Q4 – 3)

The process of executing the individual statements in a given order is called control flow.

There are three kinds of control flow structures: **Sequence (Sequence Control), Selection (Selection Control), and Repetition (Iteration).**

a. Sequence (Sequence Control)

A sequence is series of steps that are followed one after another. A sequence always occurs in the same order, without decisions or repetition. Execute a list of statements in order.

Consider an example,

Algorithm for Addition of two numbers:

Step1: Start the process

Step 2: Get two numbers as input and store it in to a and b

Step 3: Add the number a & b and store it into c

Step 4: Print c

Step 5: Stop the process

The above example is the algorithm to add the two numbers. It says the sequence of steps to be followed to add two numbers.

As, Step2 says that should get two numbers from the user and store it in some variable.

In step3, the algorithm start doing the process of adding two numbers and step4 print the result generated by step3.

b. Decision (Selection Control)

A decision is a form of questions. Usually decisions are YES/NO or TRUE/FALSE type questions (like "Is the traffic light red?"). However, some decisions can have more than two options (like "What colour is the traffic light?")

Choose at most one action from several alternative conditions.

A selection statement causes the program control to be transferred to a specific part of the program based upon the condition.

Algorithm to find biggest among 2 nos.:

Step1: Start the process

Step 2: Get two numbers as input and store it in to a and b

Step 3: If a is greater than b then

Step 4: Print a is big

Step 5: else

Step 6: Print b is big

Step 7: Stop the process

Selection structure executes some set of statements based on the condition given.

The above algorithm is an example to find the biggest numbers among two numbers.

In Step2, we are getting two numbers as input and storing it in the variables a and b.

We are checking whether a is greater than b

In step3, if yes then we are printing a is big or we are printing b is big.

c. Repetition

A repetition is something that happens over and over again, or something that you do "Until" something happens.

Repetition (loop) may be defined as a smaller program the can be executed several times in a main program. Repeat a block of statements while a condition is true.

Algorithm to calculate factorial:

Step1: Start the process

Step 2: Read the number num

Step 3: Initialize i is equal to 1 and fact is equal to 1

Step 4: Repeat step4 through 6 until I is equal to num

Step 5: fact = fact * i

Step 6: i = i+1

Step 7: Print fact

Step 8: Stop the process

The above algorithm is an example for repetition, where it repeats some processes until some condition get satisfied.

Here this algorithm helps to find the factorial of number.

In step 2 we are getting a number as input and store it in num.

Step 5 and Step 6 is going to do repeatedly base on the condition given in step 4.

Finally, Step 7 prints the result obtained from step 5 and step 6.

4. Functions (Q4 – 4)

It is a group of statements that is used to perform some task. Function is a subprogram. It easy to write function and easy to find out error in the program if we are using function.

Usually, function is used for divide a large complex algorithm into sub algorithm, due to this we can easily debug and maintain the code.

Benefits: it will minimize the development time and reduce the complexity of the program. Reduction in line of code, code reuse, Better readability, Easy to debug and test and improved maintainability.

Example: A simple calculator program can be divided into minimum four sub algorithm they are Addition, Subtraction, Multiplication and division. It is easier to analyse their behaviour and properties from a mathematical point of view.

Few examples:

Write an algorithm to fins sum of two numbers:

Step 1: Start the process

Step 2: Read first number

Step 3: Read second number

Step 4: Find the sum of first number and second number (sum = fn+sn)

Step 5: Display the value of sum

Step 6: Stop the process

Or

Step 1: Start the process

Step 2: Read number1 and number2

Step 3: Add number1, number2 and assign the result to sum (sum = number1 + number2)

Step 4: Print sum

Step 5: Stop the process

Write an algorithm to find largest of two numbers

Step 1: Start the process

Step 2: Read number1 and number2

Step 3: Compare number1 and number2. (number1 > number2) If number1 is greater, Display number1 is greatest else display number2 is greatest

Step 4: Stop the process

Write an algorithm to find the greatest among three numbers

Step 1: Start the process

Step 2: Read number1, number2 and number3

Step 3: Compare number1 and number2. If number1 is greater perform step 4 else step 5

Step 4: Compare number1 and number3. If number1 is greater, Display number1 is greatest else display number 3 is greatest

Step 5: Compare number2 and number3. If number2 is greater, Display number2 is greatest else display number3 is greatest

Step 6: Stop the process

Q5: Notations (pseudo code, flow chart, programming language)

Algorithm can be expressed in many different notations, including pseudo code, flowcharts and programming languages. Pseudo code represents the algorithm through structured human language. Flowchart represents the algorithm graphically. Programming languages are intended for expressing algorithms in a form that can be executed by a computer.

1. Pseudo code (Q5 - 1)

1. Definition

“Pseudo” means imitation or false. “Code” means the set of statements or instructions written in a programming language.

Pseudo code is a language representation of algorithm in the form of annotations and informative text written in plain English.

It’s typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines.

Not need knowledge of program language to understand or write a pseudo code.

Pseudo code can’t be compiled and executed.

There is no standard syntax for pseudo code.

Pseudocode is also called as “Program Design Language [PDL]”.

2. Rules for writing pseudo code

- Write one statement per line
- Capitalize initial keyword (READ, WRITE, IF, WHILE, UNTIL).
- Indent to show hierarchy (loops, states and iterations)
- Keep statements language independent (never use any programming language & syntax)
- End multiline structures (improve readability the initial start and end of the several lines must be specified properly)

3. Guidelines for preparing a pseudo code

Common keywords used in pseudocode

a. //: This keyword used to represent a comment.
b. BEGIN, END: Begin is the first statement and end is the last statement.
c. INPUT, GET, READ: The keyword is used to inputting data.
d. COMPUTE, CALCULATE: used for calculation of the result of the given expression.
e. ADD, SUBTRACT, INITIALIZE used for addition, subtraction and initialization.
f. OUTPUT, PRINT, DISPLAY: It is used to display the output of the program.

g. IF, ELSE, ENDIF: used to make decision.
h. WHILE, ENDWHILE: used for iterative statements.
i. FOR, ENDFOR: Another iterative incremented/decremented tested automatically.

Addition of two numbers:

START **SEND 'Enter first number' TO DISPLAY** **RECEIVE num1 FROM (INTEGER) KEYBOARD** **SEND 'Enter second number' TO DISPLAY** **RECEIVE num2 FROM (INTEGER) KEYBOARD** **SET total TO num1 + num2** **SEND total TO DISPLAY** **END**	BEGIN GET a, b ADD c = a + b PRINT c END

4. Advantages

- Pseudo is independent of any language; it can be used by most programmers.
- It is easy to translate pseudo code into a programming language.
- It can be written easily and it can be easily modified as compared to flowchart.
- Converting a pseudo code to programming language is very easy as compared with converting a flowchart to programming language.
- There are no accepted standards for writing pseudo codes.

5. Disadvantages

- Pseudocode does not provide visual representation of the program's logic
- There is no standardized style or format for pseudo code
- For a beginner, it is more difficult to follow the logic or write pseudocode as compared to flowchart.
- It cannot be compiled nor executed.

2. Flow Chart (Q5 - 2)

1. Definition

Flowchart is a diagrammatic representation of the logic for solving a task.
It uses simple geometric shapes to represent processes and arrows to show relationship and process/data flow.
The beginning or end of a program is represented by an oval.
A process is represented by a rectangle.
A decision is represented by a diamond.
An I/O process is represented by a parallelogram.
Program preparation can be simplified using the flowcharts.
Flowcharts are easier to understand at a glance than the narrative description of algorithm.
Flowcharts assist in reviewing and debugging of a program.

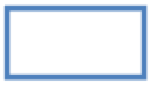
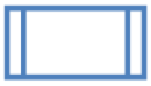
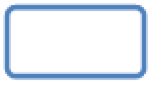
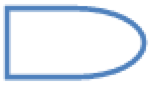
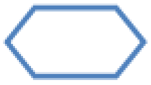
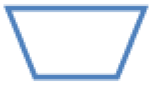
2. Rules for drawing flowchart

a. Standard symbols should only be used

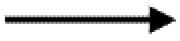
b. Flow of arrow heads allowed only from top to bottom, left to right or right to left, bottom to top
c. Flow lines should not cross each other
d. Only one flow line should come to / from process symbols
e. Only one flow line should enter into decision symbol, but there may be maximum of three output flow lines can come out.
f. Keep the flow chart should be as simple as possible
g. If new page needed, then use connectors for better representation

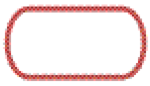
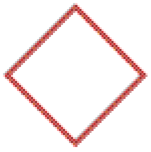
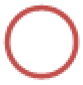
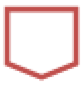
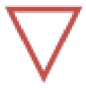
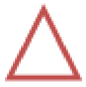
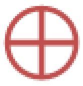
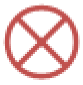
3. Symbol used in flowchart

a. Process / Operation Symbols

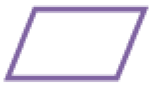
Symbol	Name	Description
	Process	Show a Process or action step.
	Predefined Process (Subroutine)	A Predefined Process symbol is a marker for another process step or series of process. This shape commonly depicts sub-processes or subroutines.
	Alternate Process	This flowchart symbol is used when the process flow step is an alternate to the normal process step.
	Delay	The Delay flowchart symbol depicts any waiting period that is part of a process.
	Preparation	Any process step that is a Preparation process flow step, such as a set-up operation.
	Manual Operation	Manual Operations flowchart shapes show which process steps are not automated.

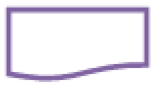
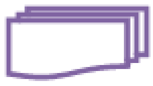
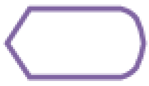
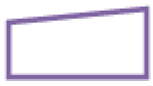
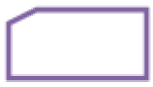
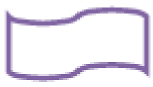
b. Branching and Control of Flow Symbols

Symbol	Name	Description
	Flow Line (Arrow, Connector)	Flow line connectors show the direction that the process flows.

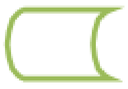
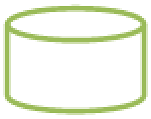
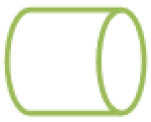
Symbol	Name	Description
	Terminator (Terminal Point, Oval)	Terminators show the start and stop points in a process.
	Decision	Indicates a question or branch in the process flow.
	Connector (Inspection)	It is used as a Connector to show a jump from one point in the process flow to another.
	Off-Page Connector	Off-Page Connector shows continuation of a process flowchart onto another page.
	Merge (Storage)	This symbol shows the merging of multiple processes or information into one.
	Extract (Measurement)	This symbol shows when a process splits into parallel paths.
	Or	The logical Or symbol shows when a process diverges - usually for more than 2 branches.
	Summing Junction	The logical Summing Junction flowchart shape is shows when multiple branches converge into a single process.

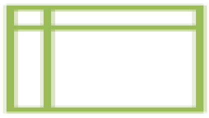
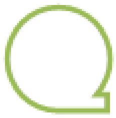
c. Input and Output Symbols

Symbol	Name	Description
	Data (I/O)	The Data flowchart shape indicates inputs to and outputs from a process.

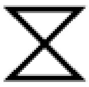
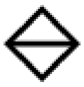
Symbol	Name	Description
	Document	Document flowchart symbol is for a process step that produces a document.
	Multi-Document	Multi-Document flowchart symbol is for a process step that produces a multiple docuents
	Display	Indicates a process step where information is displayed.
	Manual Input	Manual Input flowchart get the input manually into a system.
	Card	This is the companion to the punched tape flowchart shapes.
	Punched Tape	Punched Tape symbol - used for input into old computers and CNC machines.

d. File and Information Storage Symbols

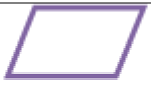
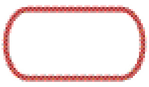
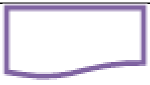
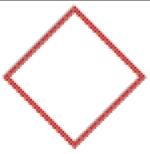
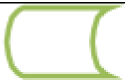
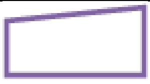
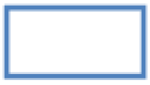
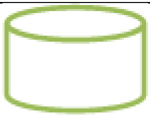
Symbol	Name	Description
	Stored Data	A general Data Storage flowchart shape used for any process step that stores data.
	Magnetic Disk (Database)	The most universally recognizable symbol for a data storage location, this flowchart shape depicts a database.
	Direct Access Storage	Direct Access Storage is a fancy way of saying Hard Drive.

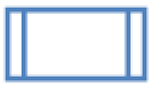
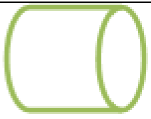
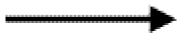
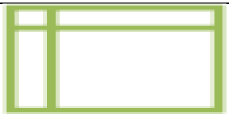
Symbol	Name	Description
	Internal Storage	Used in programming flowcharts to mean information stored in memory, as opposed to on a file.
	Sequential Access Storage (Magnetic Tape)	Although it looks like a 'Q', the symbol is supposed to look like a reel of tape.

e. Data Processing Symbols

Symbol	Name	Description
	Collate	The Collate flowchart shape indicates a process step that requires organizing data, information, or materials according into a standard format.
	Sort	Indicates the sorting of data, information, materials into some pre-defined order.

The most symbols in the flowchart are:

	Data (I/O)		Terminator (Terminal Point, Oval)
	Document		Decision
	Stored Data		Manual Input
	Process		Magnetic Disk (Database)

	Predefined Process (Subroutine)		Direct Access Storage
	Flow Line (Arrow, Connector)		Internal Storage

4. Types of flowcharts (4 types they are)

i. **Document flowcharts** have the purpose of showing existing controls over a document flow through the components of the system. This chart is read from left to right and explain the flow of documents through various business units.

ii. **Data flowcharts** have the purpose of showing the controls governing data flows in the system. To show how data is transmitted through the system rather than the control flow.

iii. **System flowcharts** show the controls located at the physical or resources level. Show the flow of data through the major components of the system such as data entry, programs storage media, processors and communication networks.

iv. Program flowcharts show the controls placed internally to the program within a system. Program flow cart is a gift to programmers as it makes programming task very easy and systematic.

5. Levels of flowchart

a) Macro flowchart

A flowchart may not need more amount of details required for the process. A view of the process is enough for their process. It shows only lesser details.

b) Micro flowchart

A flowchart that requires detailed description of process steps. It provides the detailed view for specific portion of process by documenting every action and decision. It is commonly used to perform a particular task.

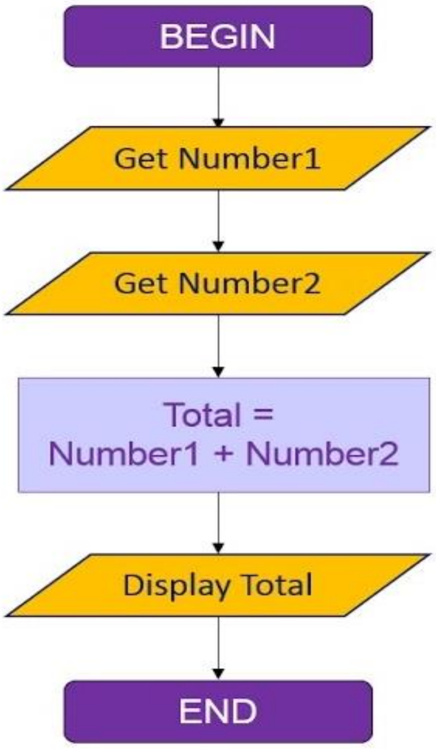
c) Mini flowchart

It is used for a flowchart that falls between detail view of macro level and lesser view of micro level.

6. Basic design structure in flowchart

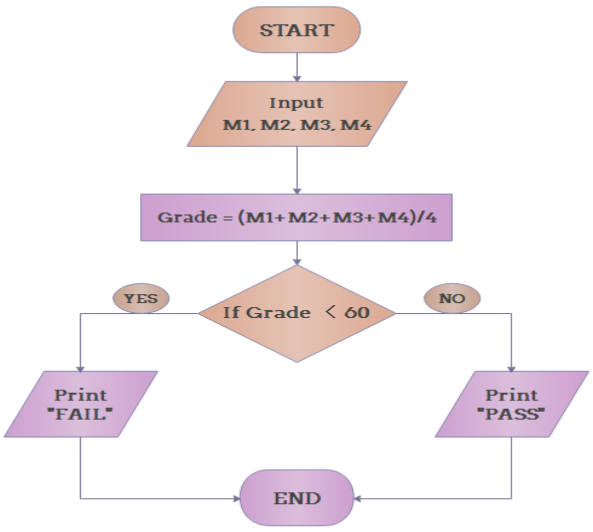
i. Sequence structure

A series of actions are performed in sequence
The pay-calculating example was a sequence flowchart.



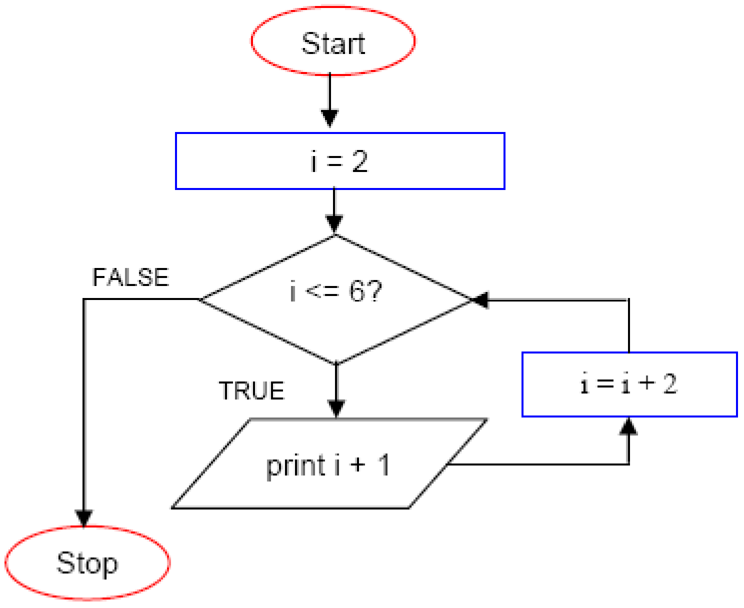
ii. Selection control structure

The flowchart segment below shows how a decision structure is expressed in python as an if/else statement



iii. Loop control structure

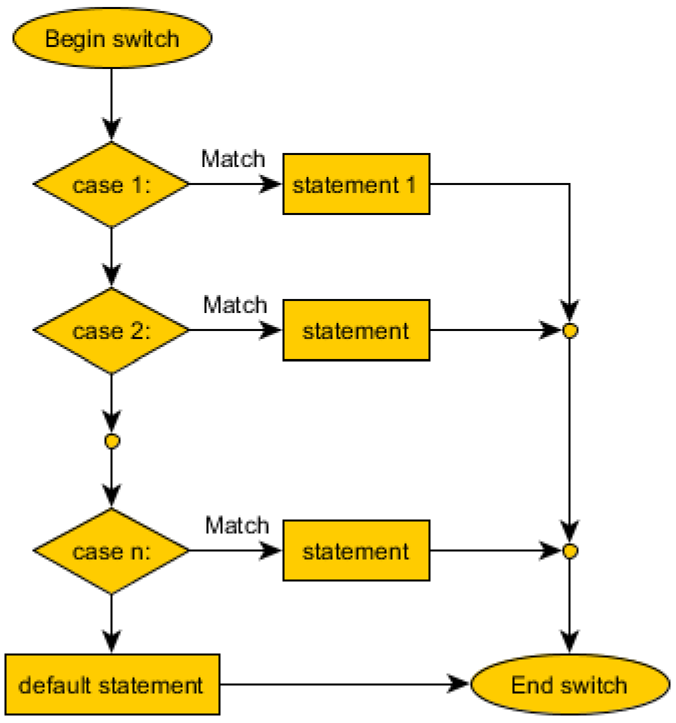
The flowchart segment below shows a repetition structure expressed in Python as a while loop.



The action performed by a repetition structure must eventually cause the loop to terminate. Otherwise, an infinite loop is created.

iv. Switch Case:

The switch tries to match an expression to a number of possible values, called cases. If no case matches, the mechanism executes the default statement.



7. Advantages

Crystal Clear Communication – better way for communicating the logic of a system.

Standard symbols: improves flowchart's readability.

Proper documentation: to know what the program actually does and how to use the program

Efficient coding: it is the blueprint during the systems analysis and program development phase

Effective analysis, Easy for debugging, efficient program maintenance.

8. Disadvantages

Complex Logic: Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.

Alterations and Modifications: If alterations are required the flowchart may require re-drawing completely.

Reproduction : As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.