Unit

1

Flowchart and Algorithm

1.1 Definition of Flowchart :

5

2 Flowchart and algorithms are the way to start working with the problems and them converting them into the program of any language. So, let's start learning flowchart and algorithm.

A. flowcbart is a pictorial representation of a program.A Flowchart is a graphical representation of the process or system or the step-by-step solution of the problem. The Flowchart describes the flow of data and control in the systems. The flow is a set of the logic operations.

1.2 Importance of Flowchart

Flow charts are an important tool for the improvement of processes. They help to identify the different elements of a process.Flow charts shows steps in sequential order and is used in presenting the flow of processes.Flowcharts are a means to help design a program before you start to code the program.

The benefits of using flowcharts are as follows :

Communication

Flowcharts are better way of communicating the logic of a system.

Effective analysis

With the help of flowchart, problem can be analyzed in more effective way.

Proper documentation

Flowcharts serve as a good program documentation, which is needed for various purposes.

Efficient Coding

The flowcharts are useful during the systems analysis and program development phase.

Proper Debugging

The flowchart helps in debugging process.

Efficient Program Maintenance

The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part

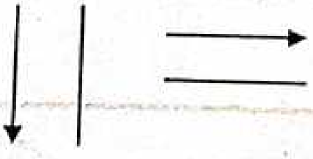
1.3 Symbols of Flowchart

Flowchart is a graphical representation of an algorithm. Programmers use it as a program-planning tool to solve a problem. It makes use of various symbols which are connected among them to indicate the flow of information and processing.

Various symbols are used to write different instructions/operations in the flowchart. All symbols are connected among themselves to indicate the flow of information and processing.

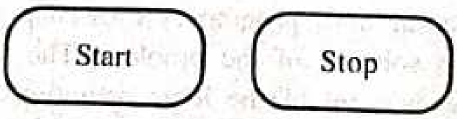
Flow Lines

Flow lines are indicated using arrow or lines. They are used to indicate flow of the problem. The flow line connects the various symbols.



Terminals

Terminals are used to indicate beginning and end of the flowchart.

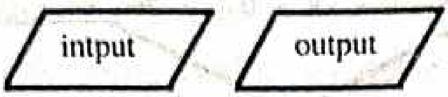


⇒ ↓| ⇉ s/e I/O P/s ◇

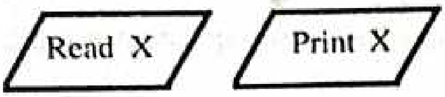
2

Input/Output

Input and output are represented as a parallelogram. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



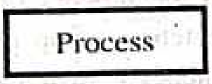
Examples: Get X from the user; display X.



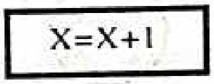
2

Processing

Processing steps are represented as rectangles.



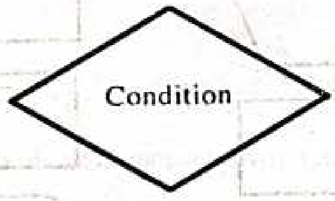
Examples: "Add 1 to X"; "replace identified part"; "save changes" or similar.



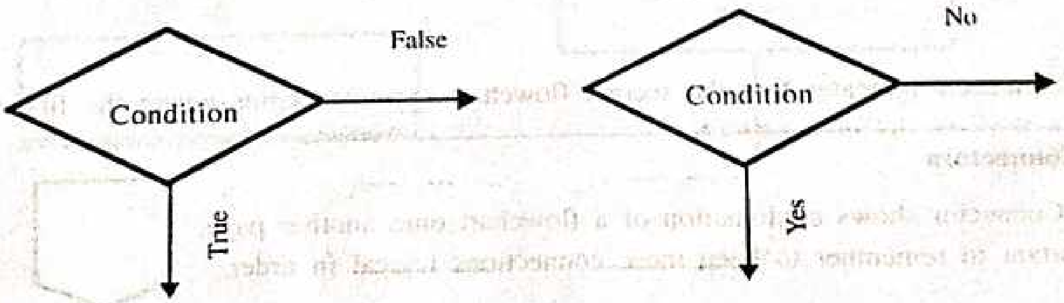
2

Decision

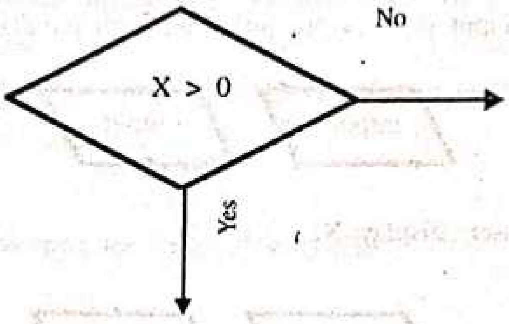
Decisions (Conditions) in flowchart are represented as a diamond (rhombus). Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



The arrows should always be labeled.



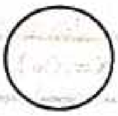
Example : Check if X is Positive or Negative ?



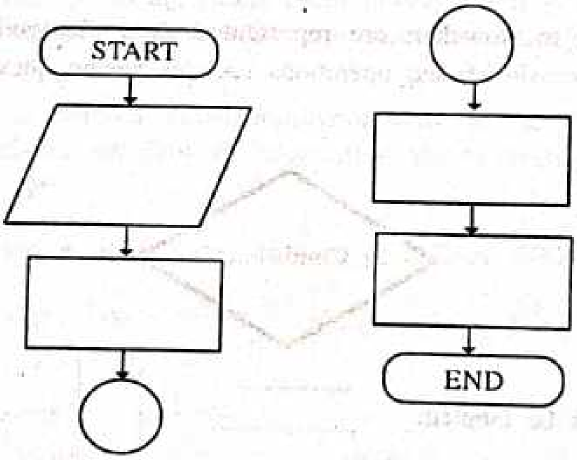
4

Connector (Inspection)

A connector (represented by a small circle) allows you to connect two flowchart segments. Connecter is used to show a jump from one point in the process flow to another. Connectors are usually labeled with capital letters (A, B, AA) to show matching jump points. They are useful for avoiding flow lines that cross other shapes and flow lines. Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.

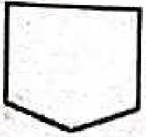


For example :



The "A" connector indicates that the second flowchart segment begins where the first segment ends.

Off-Page Connectors



Off-Page Connector shows continuation of a flowchart onto another page. It is important to remember to keep these connections logical in order.

1.4 Guidelines for Preparing Flowchart

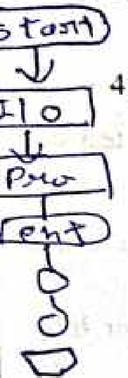
Flowcharts are usually drawn using standard symbols; however, some special symbols can also be developed when required.

The following are some guidelines in flowcharting:

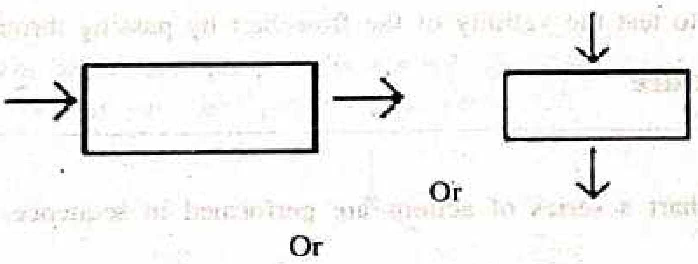
1) When drawing a proper flowchart, all necessary requirements should be listed out in logical order.

2) The flowchart should be clear, neat and easy to follow. There should not be any ambiguity in understanding the flowchart.

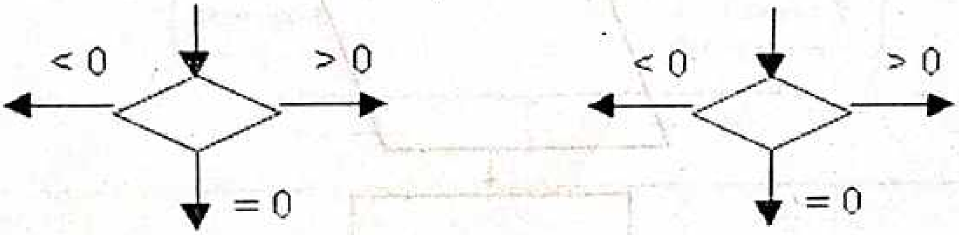
3) The usual direction of the flow of a procedure or system is from left to right or top to bottom.



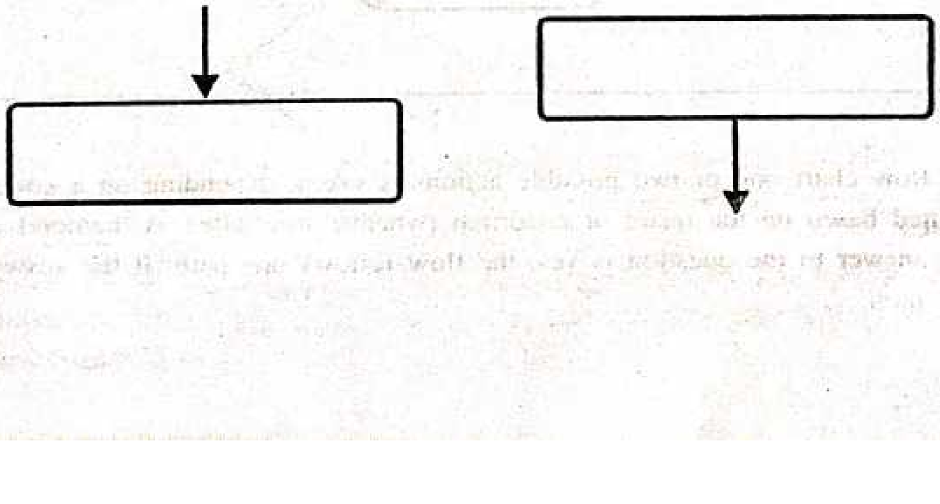
4) Only one flow line should come out from a process symbol.



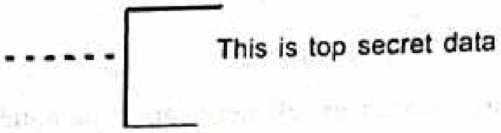
5) Only one flow line should enter a decision symbol, but two or three flow lines, one for each possible answer, should leave the decision symbol.



6) Only one flow line is used in conjunction with terminal symbol.



7) Write within standard symbols briefly. As necessary, you can use the annotation symbo describe data or computational steps more clearly.



8) If the flowchart becomes complex and lengthy, it is better to use connector symbols reduce the number of flow lines.

9) Avoid the intersection of flow lines if you want to make it more effective and better way communication.

10) Ensure that the flowchart has a logical start and finish.

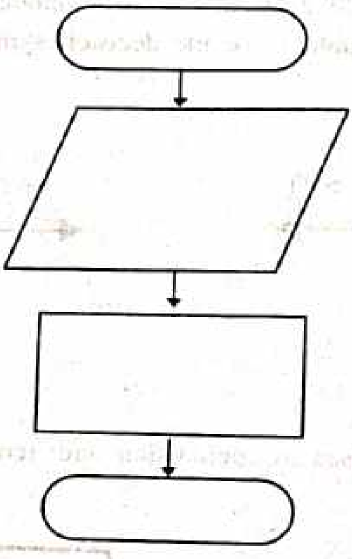
11) It is useful to test the validity of the flowchart by passing through it with a simple test da

1.5 Flowchart Structure

1. Sequence

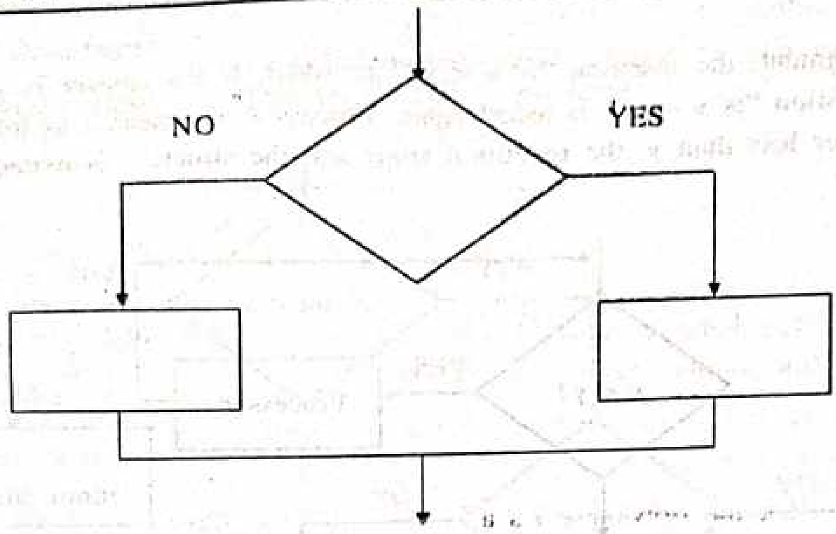
In this type of flow chart a series of actions are performed in sequence. There is no jump or loo

For **Example :**



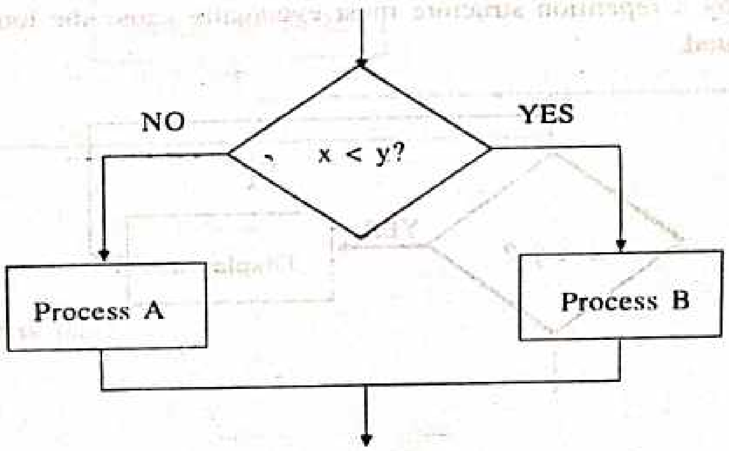
2. Selection

In this type of flow chart one of two possible actions is taken, depending on a condition result. Nex flow is determined based on the result of condition (whether true/false). A diamond indicates a yes/nc question. If the answer to the question is yes, the flow follows one path. If the answer is no, the flow follows another path



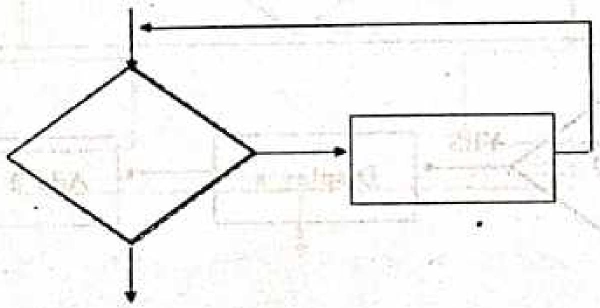
For **example :**

In the flowchart segment below, the question "is x < y?" is asked. If the answer is no, then process A is. performed; if the answer is yes, then process B is performed.



3. Repetition

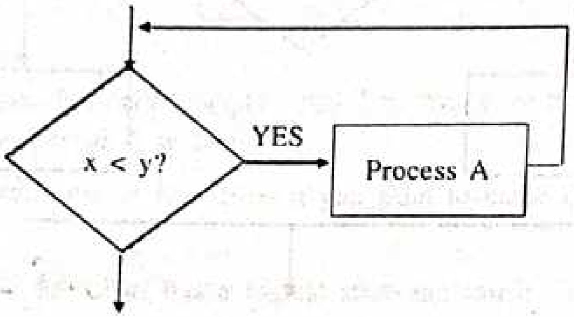
A repetition structure represents part of the program that repeats. This type of structure is commonly known as a loop or iteration.



Notice the use of the diamond symbol. A loop tests a condition, and if the condition exists, it performs an action. Then it tests the condition again. If the condition still exists, the action is repeated. This continues until the condition no longer exists.

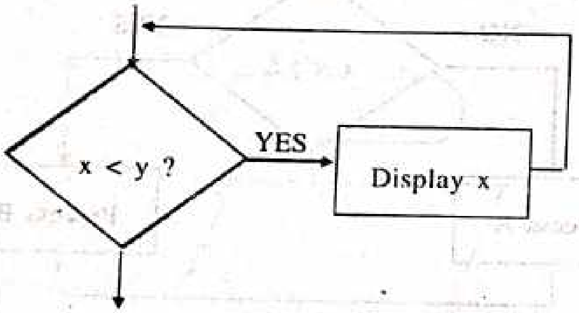
For example :

In the flowchart segment, the question "is x < y?" is asked. If the answer is yes, then Process A is performed. The question "is x < y?" is asked again. Process A is repeated as long as x is less than y. When x is no longer less than y, the repetition stops and the structure is exited.

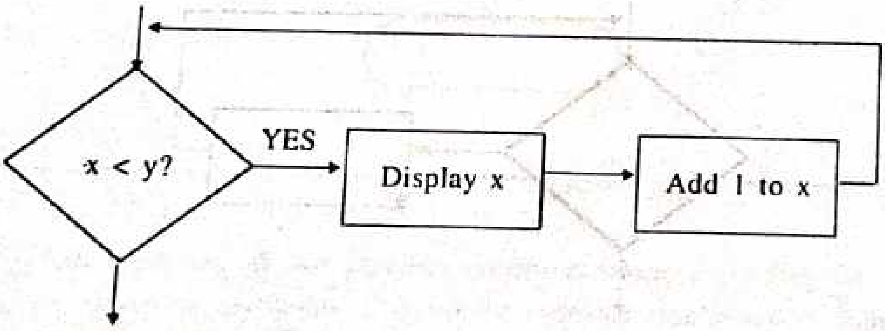


Controlling a Repetition Structure

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

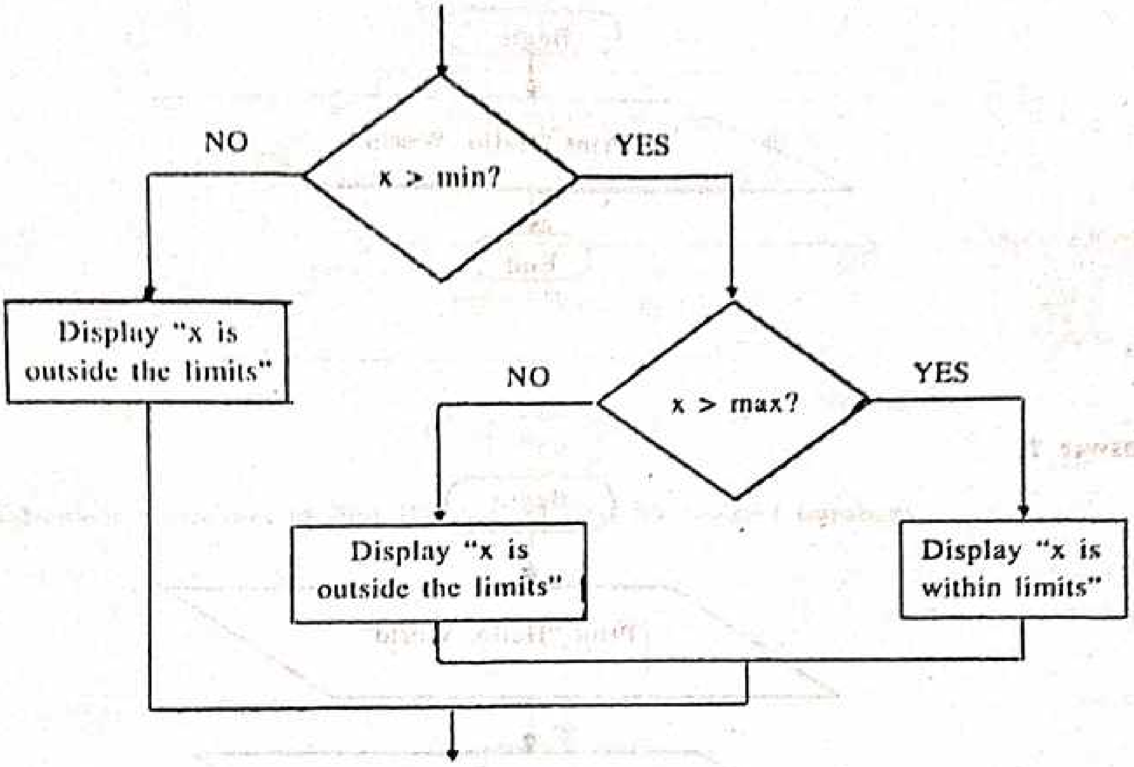


In this flowchart segment, x is never changed. Once the loop starts, it will never end. So our question is: How cans this flowchart is modified so it is no longer an infinite loop? Yes, by adding an action within the repetition that changes the value of x.



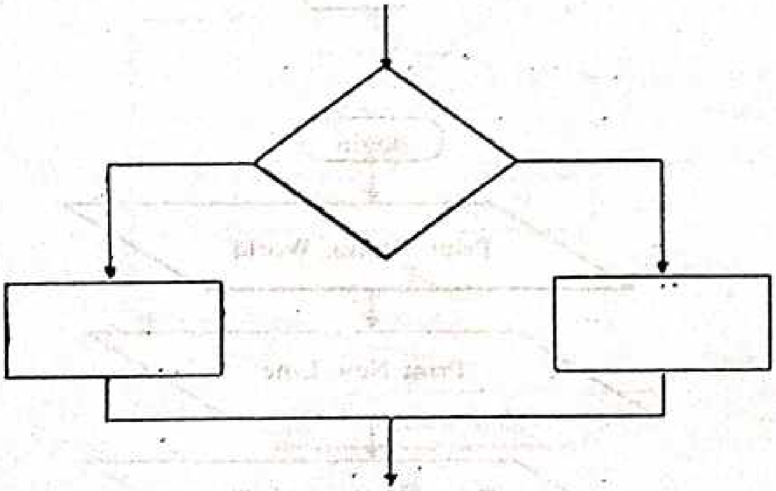
1.5.1 Combining Structures

This flowchart segment shows two decision structures combined.


NO
x > min?
YES
Display "x is outside the limits"
NO
x > max?
YES
Display "x is outside the limits"
Display "x is within limits"

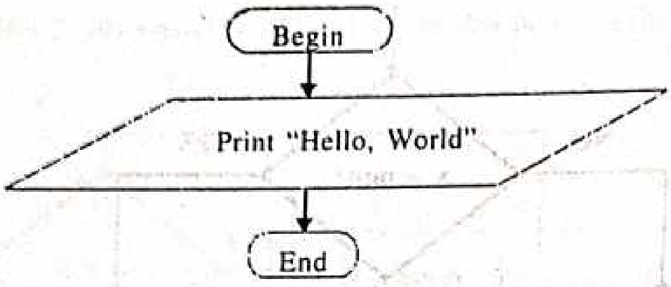
Test your skills

What type of structure is this?

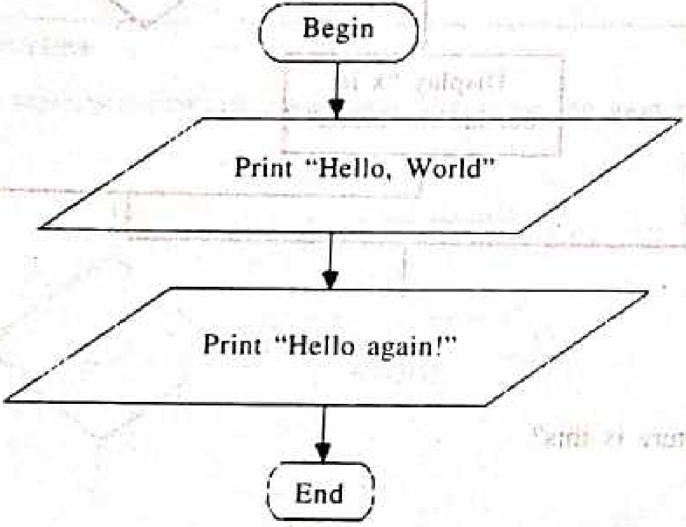


1.5.2 Flowchart Examples

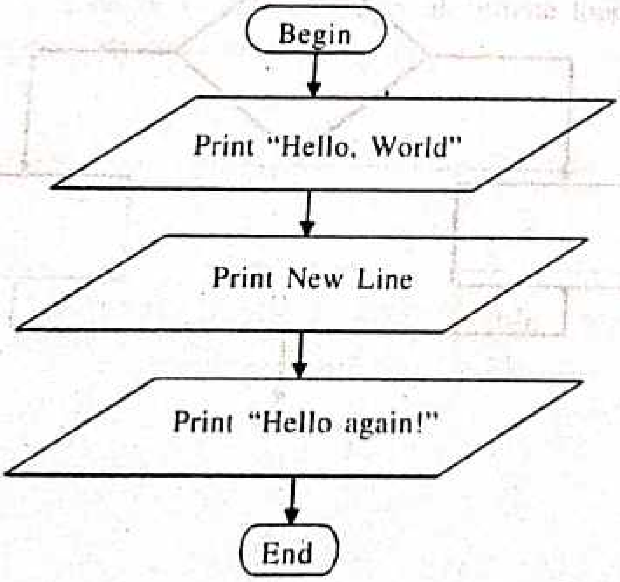
Example 1 :



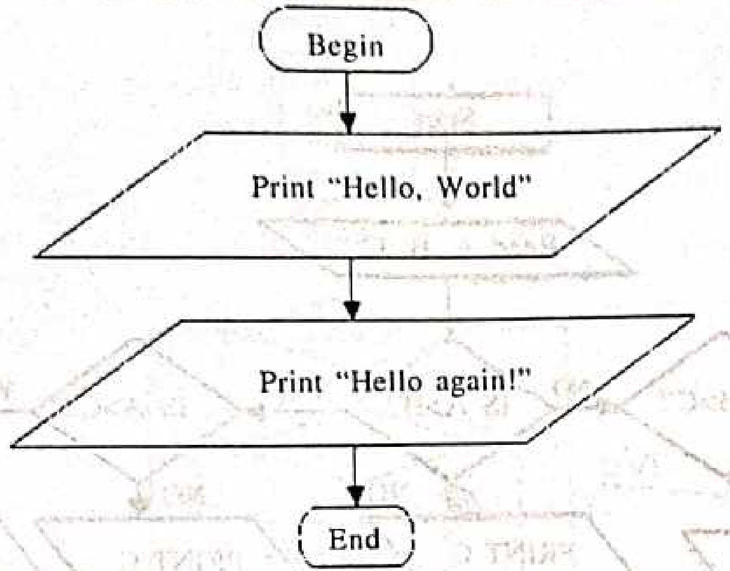
Guess Answer ?



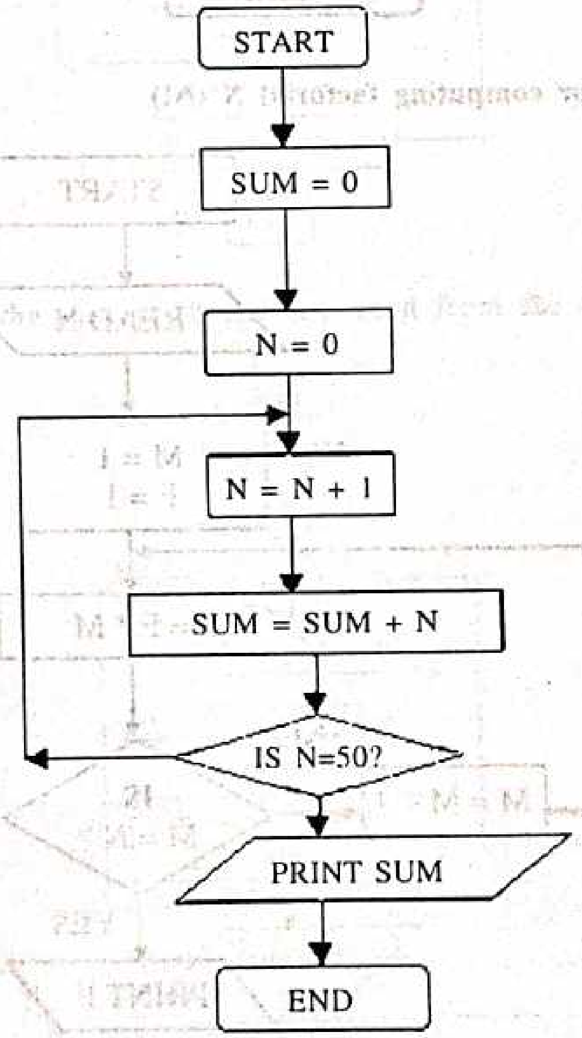
Example 2 :



Example 3 :

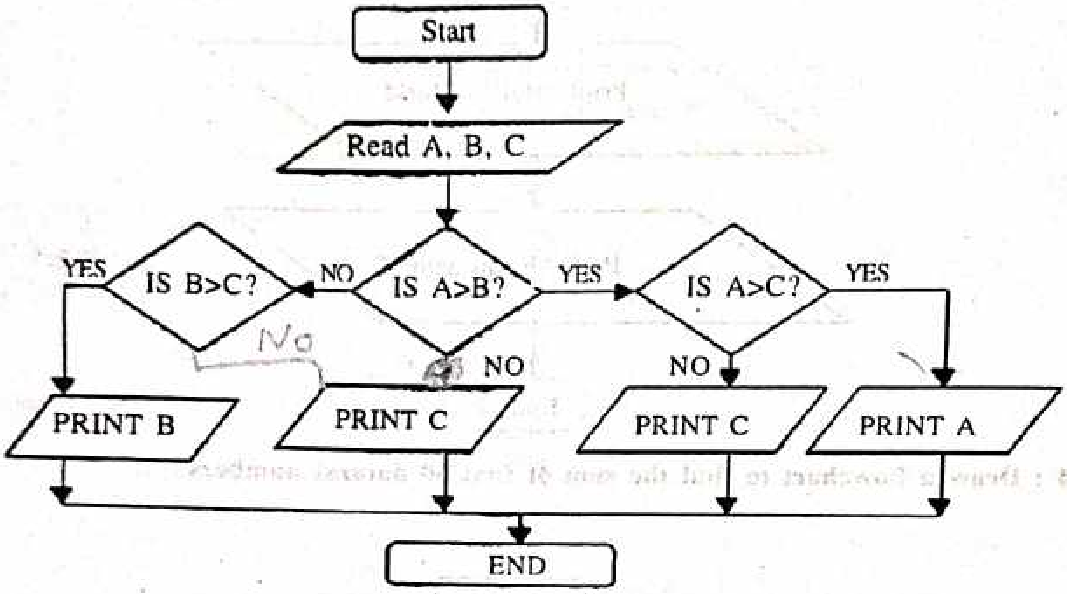


Example 4 : Draw a flowchart to find the sum of first 50 natural numbers



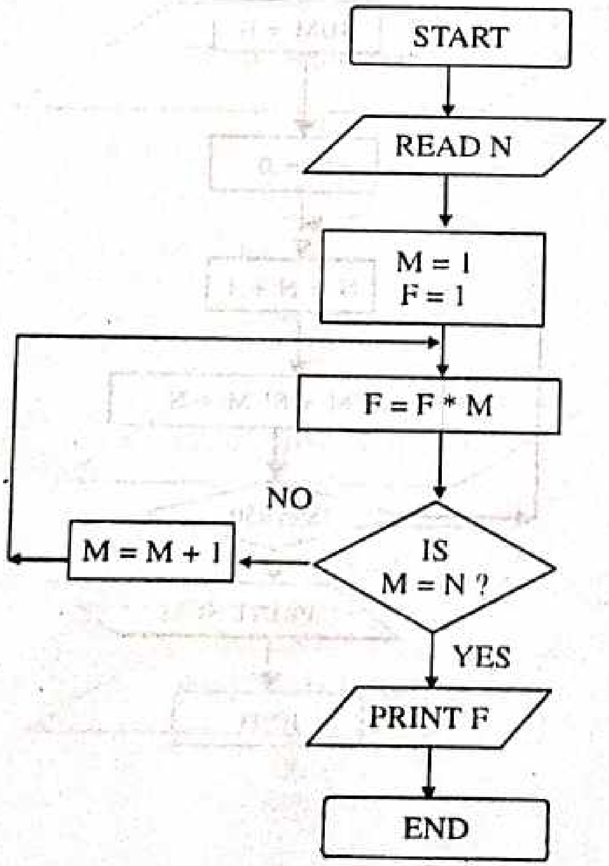
Example 5 : Flowchart for finding out the largest of three numbers

4

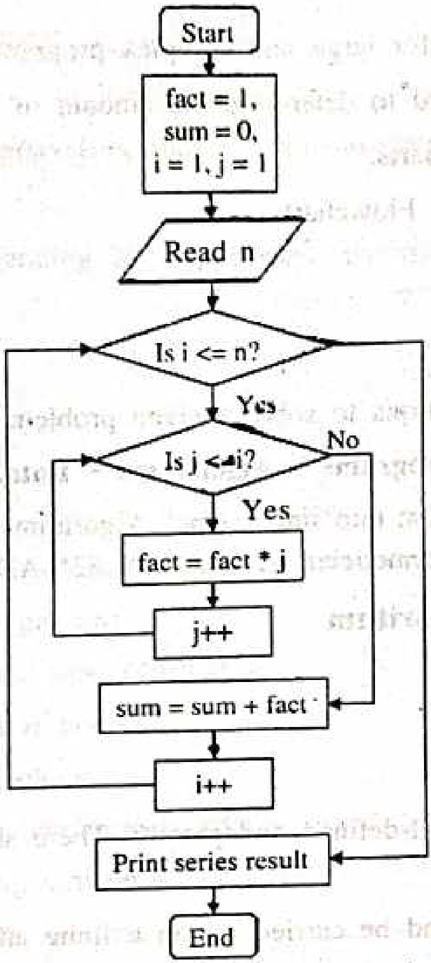


mple 6 : Draw a flowchart for computing factorial N (N!)

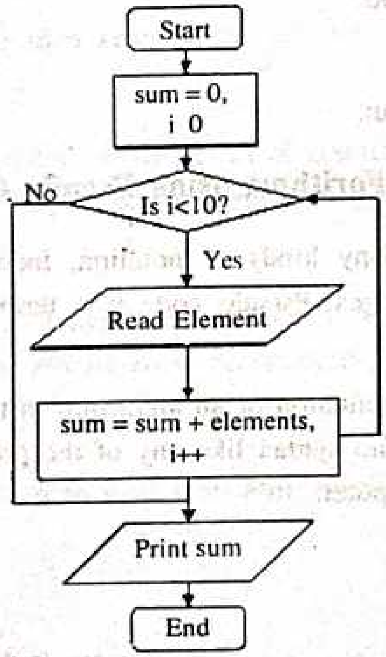
4



Example 7 : Flowchart for computing the series 1! + 2! + 3! + ...+ n!



Example 8 : Flow chart to do the sum of 10 elements read from the user



1.6 Limitations of using Flowchart

It is difficult to draw flowchart for large and complex programs.

In flowcharts there is no standard to determine the amount of detail.

Difficult to reproduce the flowcharts.

It is very difficult to modify the Flowchart.

1.7 Algorithm

1.7.1 Definition

An algorithm is a set of instructions to solve a given problem.

Programs = Algorithms + Data.

An algorithm is division of problem into small steps. Algorithm was first developed by Al-Khwarizmi, the Persian astronomer and mathematician in Arabic in 825 AD.

1.7.2 Characteristics of an Algorithm

Input

It accept zero or more inputs

Definiteness

Each instruction must be clear, well-defined and precise. There should not be any ambiguity

Effectiveness

Each instruction must be simple and be carried out in a finite amount of time.

Finiteness

Algorithm should have finite sequence of instructions. That is, it should end after a fixed time. It should not enter into an infinite loop

Output

It should produce at least one output.

1.7.3 Developing and writing Algorithms using Pseudo Code :

Algorithms can be expressed in many kinds of notation, including natural languages, pseudo-code, flowcharts, and programming languages. Pseudo code is a term which is often used in programming and algorithm.

2

Pseudo code : It's simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.

Notations used in Pseudo code

Name of algorithm

This specifies problem to be solved.

Input

It specifies any given data to process.

Step no

It is an identification tag of an instruction and it is an unsigned position number.

Explanatory comments

This is for the ease of understanding the algorithms; comments can be given within the square brackets.

Termination

To indicate end of algorithm

How to write a Pseudo-code?

- Start by writing down the purpose of the process.
- Arrange the sequence of tasks and write the pseudocode accordingly.
- Write only one statement per line.
- Use white space and indentation effectively.
- Capitalize key commands if necessary.
- Write using simple terminology
- Keep your pseudocode in the proper order
- Use standard programming structures
- Organize your pseudocode sections.

1.7.4 Different Patterns of Algorithms IMP

Sequential

In this pattern different steps occur in a sequence

Conditional

In this different steps are executed based on the result of condition (whether true/false). In programming languages, a conditional pattern is implemented using decision making statements.

Iterative

In this a task (one or more steps) is repeated more than once. In programming languages, an iterative pattern is implemented using loops. An iterative construct is also known as "repetitive" construct

1.7.5 Examples of Algorithm

3 **Example 1 (Sequential) : An algorithm to find sum and average of 3 numbers**

```
Algorithm : CalculateSum
Step 1: Start
Step 2: Read A, B, C [Input 3 Numbers]
Step 3: Sum ß A + B + C [Find Sum]
Step 4: Avg ß Sum/3 [Find Average]
Step 5: Print Sum, Avg [Output the Result]
Step 6: Stop
```

Example 2 (Decision making) : An Algorithm to find whether an entered integer number is positive or negative

```
Algorithm : CheckMagnitude
Step 1: Start [Beginning of the Algorithm]
Step 2: Read a number A [Input the integer number]
Step 3: Check A > 0 [Compare A to zero]
        Print "number as positive"
        Else
        Print "number as negative"
Step 4: Stop [End of the Algorithm]
```

Example 3(Decision making) : Write an algorithm to find largest of two numbers

```
Algorithm: FindMax
Step 1: Start [Beginning of the Algorithm]
Step 2: Read a, b    [Input two numbers a, b]
Step 3: Check a > b    [Compare a and b]
        Print "a is maximum"
        Else
        Print "b is maximum" [Output the result]
Step 4: Stop
```

Example 4(Iterative): Algorithm to do the sum of 10 numbers read from the user

```
Algorithm: AddNumbers
Step 1: Start
Step 2: sum = 0, i = 0
Step 3: Repeat steps 3 to 5 until i<10
Step 4: Read number
Step 5: sum = sum + number [Add the number to sum]
Step 6:i = i+1 [Increment to read next number]
Step 7: Print sum
Step 8: Stop
```

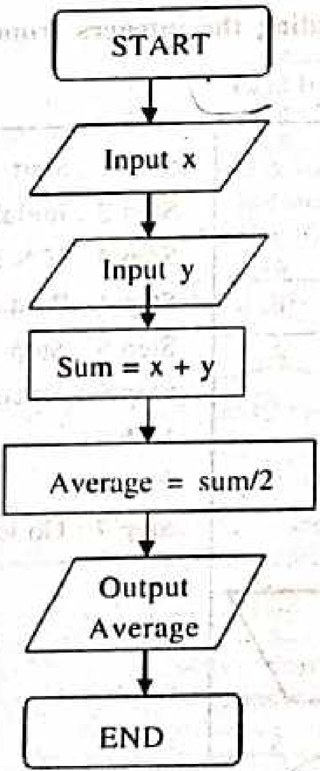
Example 5(Iterative) : Algorithm to compute the series 1! + 2! + 3! + ...+ n!

```
Algorithm : AddFactorials
Step 1: Start
Step 2: fact = 1, sum = 0, i = 1, j= 1
Step 3: Read number into n
Step 4: Repeat steps 5 to 9 for i<=n [To add the factorials of numbers]
Step 5: Repeat steps 6 and 7 for j<=I [To find the factorials of numbers]
Step 6: fact = fact * j [Find the factorial of number]
Step 7: j = j+1
Step 8: sum = sum + fact [Add the factorials of numbers]
Step 9: i = i+1
Step 10: Print series result
Step 11: Stop
```

1.7.6 Examples Using Flowchart and Algorithm

Example 1: The flowchart and algorithm to find average of two numbers

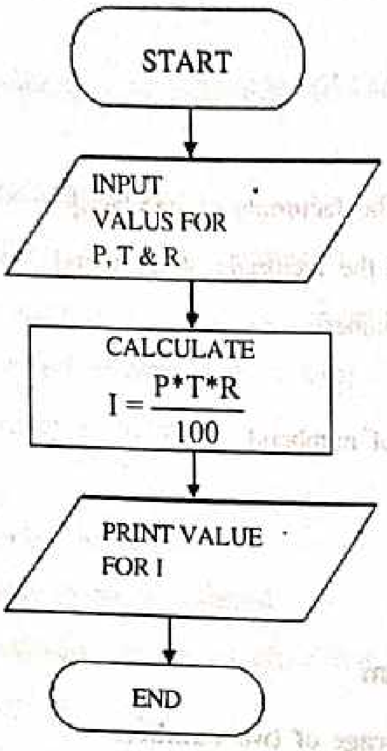
Flowchart



Algorithm : FindAverage

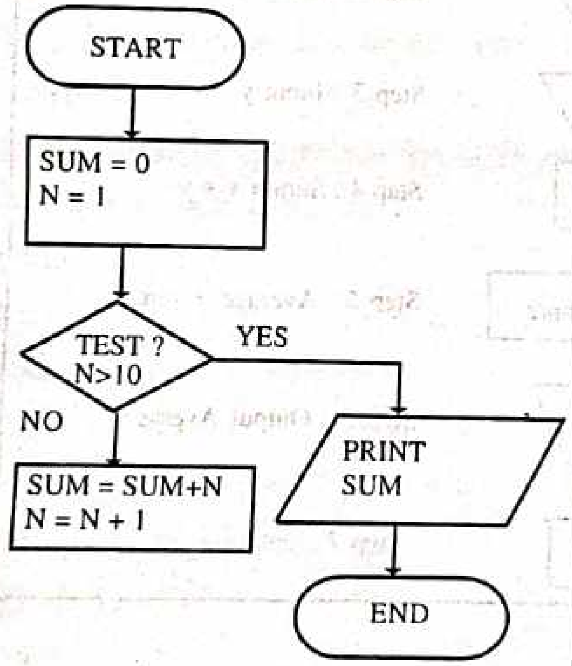
```
Step 1 : Start
Step 2 : Input x
Step 3 : Input y
Step 4 : Sum = x + y
Step 5 : Average = sum/2
Step 6 : Output Average
Step 7 : End
```

Example 2: The flowchart and algorithm to find simple interest



```
Step 1 : START
Step 2 : Read P T and R
Step 3 : Calculate I = P * R * T 100
Step 4 : Write I
Step 5 : End
```

Example 3: The flowchart and algorithm for adding the integers from 1 to 10



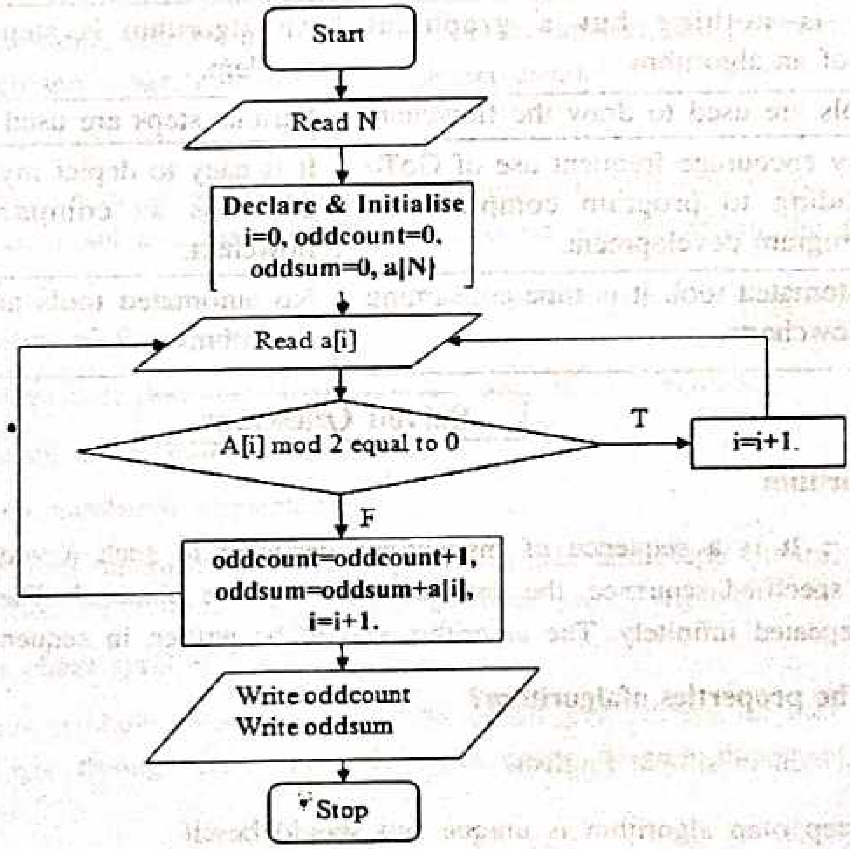
```
Step 1 : Start
Step 2 : Initials SUM=0 and N=1
Step 3 : If N is greater than 10 then
Step 4 : Print SUM
Step 5 : Stop
Step 6 : Calculate SUM=SUM+N
N=N+1
End if
Step 7 : Go to Step 3
```

Example 4 : An algorithm and flowchart to accept N numbers and count how many of them are odd and also compute sum of all these odd numbers

Algorithm :

```
Step 0: Start
Step 1: Read N.
Step 2: Declare and initialize
        Integer a[N], i=0,oddcount=0,oddsum=0.
Step 3: repeat step4 to step6 until i<N.
Step 4: read a[i]
Step 5: if (a[i] mod 2 Not Equal to 0) then
        oddcount=oddcount+1
        oddsum=oddsum+a[i]
Step 6: i=i+1
Step 7: display oddcount, oddsum
Step 8: stop
```

Flowchart



1.7.7 Advantages of Algorithm

2

Effective communication

Since algorithm is written in English like language, logic of the problem can be explained easily others

Easy and efficient coding

It acts as a blue print during a program development

Program debugging

Helps in debugging so that we can identify the logical errors in a program

Program maintenance

Maintaining the software becomes much easier

1.7.8 Disadvantage of Algorithm

2

1) Developing algorithm for large and complex problems would be time consuming and difficult understand

2) Understanding complex logic through algorithms would be difficult

1.7.9 Comparison of Flowchart and Algorithm

2

Flowchart	Algorithm
A flowchart is nothing but a graphical representation of an algorithm.	An algorithm is step by step solution of any problem.
Various symbols are used to draw the flowchart	Various steps are used to write an algorithm
Flowcharts may encourage frequent use of GoTo statements leading to program complex and unstructured program development.	It is eacy to depict my complex problem in terms of steps as compared to drawing complex flowchart.
Without an automated tool, it is time-consuming to maintain Flowcharts.	No automated tools are required to maintain the algorithms.