MCA First Year- KCA 102
Unit-1.1 Problem Solving & Programming

Introduction

- You can't solve a problem you don't understand. There is a difference between the problem and the problem you think you are solving. It's easy to start reading the first few lines in a problem and assume the rest of it because it's similar to something you've seen in the past.
- Example: Let's take a simple problem to find Even Numbers If there are no even numbers, return the empty Here are some questions that run through my mind:

How can a computer tell what is an even number?

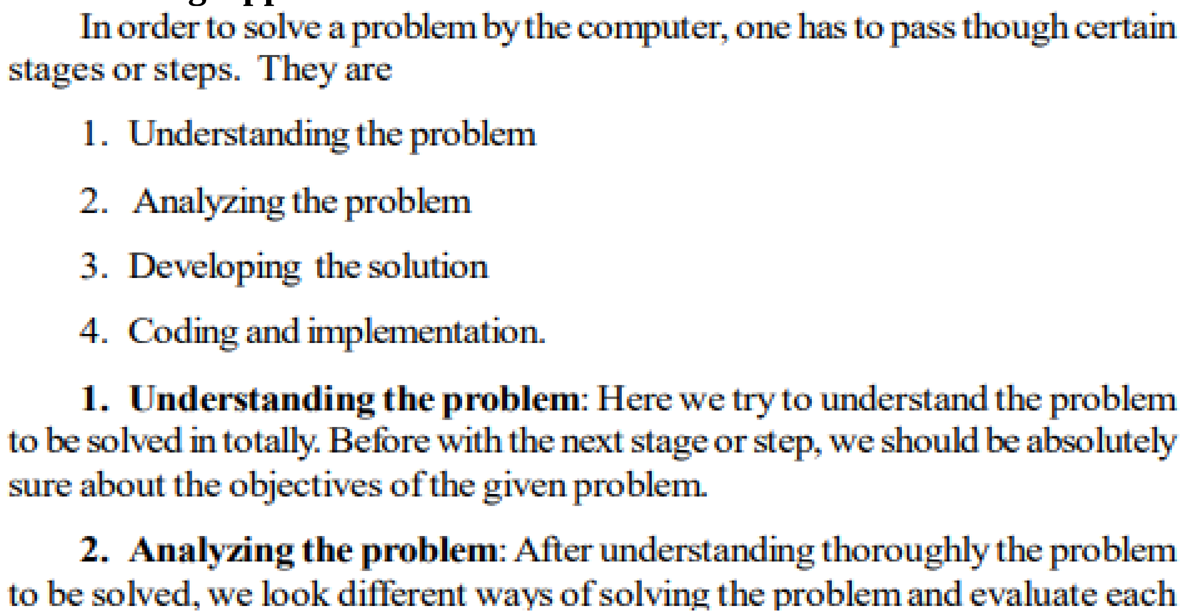
- ➢ Divide that number by 2
- ➢ see if its remainder is 0.

Take out a piece of paper and work through the problem manually. Think of at least three sets of sample data you can use.

Thumb Rules are:

1. **"Read the problem at least three times (or however many makes you feel comfortable)"**
2. **Work through the problem manually with at least three sets of sample data**

Problem Solving Approach

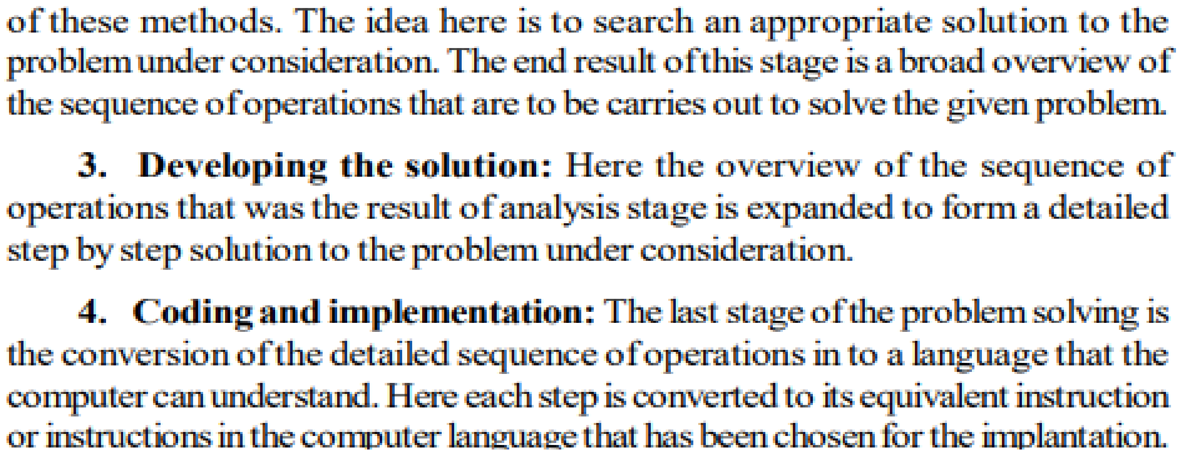


In order to solve a problem by the computer, one has to pass though certain stages or steps. They are

1. Understanding the problem
2. Analyzing the problem
3. Developing the solution
4. Coding and implementation.

1. Understanding the problem: Here we try to understand the problem to be solved in totally. Before with the next stage or step, we should be absolutely sure about the objectives of the given problem.

2. Analyzing the problem: After understanding thoroughly the problem to be solved, we look different ways of solving the problem and evaluate each



of these methods. The idea here is to search an appropriate solution to the problem under consideration. The end result of this stage is a broad overview of the sequence of operations that are to be carries out to solve the given problem.

3. Developing the solution: Here the overview of the sequence of operations that was the result of analysis stage is expanded to form a detailed step by step solution to the problem under consideration.

4. Coding and implementation: The last stage of the problem solving is the conversion of the detailed sequence of operations in to a language that the computer can understand. Here each step is converted to its equivalent instruction or instructions in the computer language that has been chosen for the implantation.

Problem Solving strategies/Techniques

There are two very commonly-used strategies in solving problems are the algorithmic and heuristic.

Algorithmic

- A finite series of steps which if faithfully performed will always result in a task or process being completed -- in this case, the problem will be solved.
- In problem solving, this approach is sometimes called a "brute-force" solution because all possible rearrangments are tried.

Heuristic

- A rule of thumb -- selective searches involving looking at only those portions of the problem space that are most likely to produce a solution.

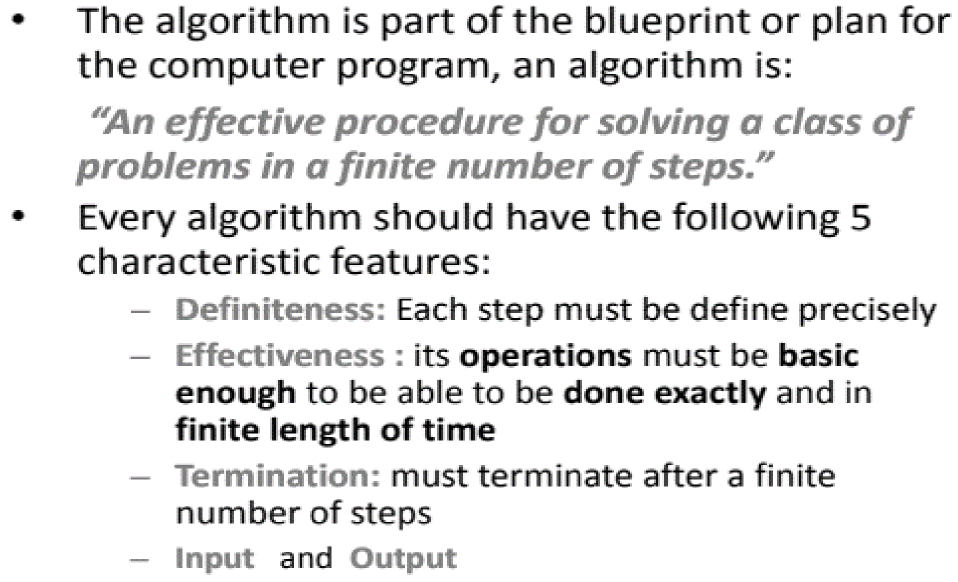
Problem Statement:

Problem Statement help diagnose the situation so that your focus is on the problem, helpful tools at this stage include Algorithms and flowcharts for identifying the expected steps of a process. Therefore, to solve any problem,

Collect and analyze information and data
Talk with people familiar with the problem
If at all possible, view the problem first hand
Confirm all findings

To Understand and Analyze the Problem following techniques used

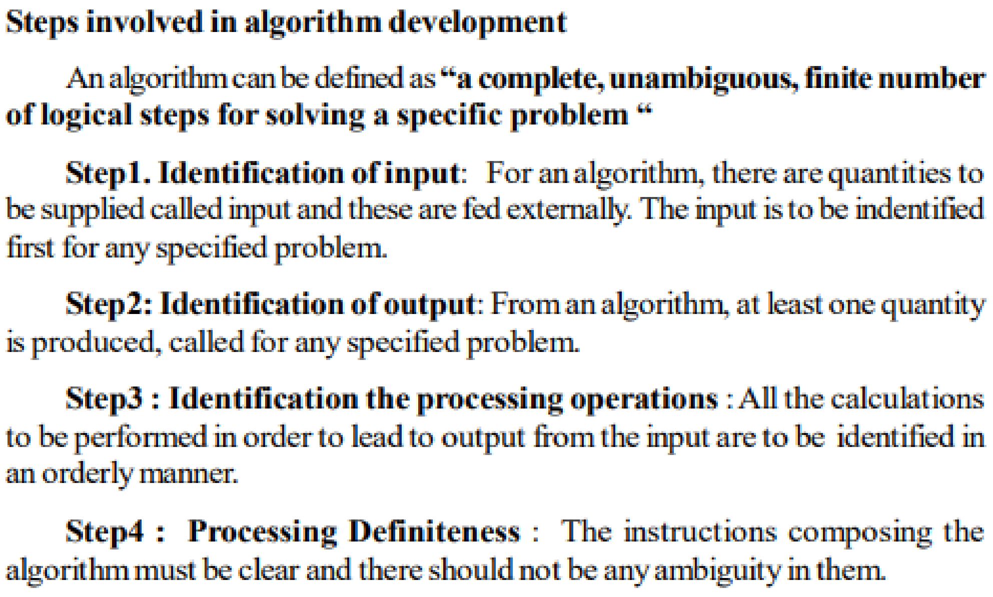
1. ***Algorithm***



- The algorithm is part of the blueprint or plan for the computer program, an algorithm is:

  "An effective procedure for solving a class of problems in a finite number of steps."

- Every algorithm should have the following 5 characteristic features:
  - Definiteness: Each step must be define precisely
  - Effectiveness : its **operations** must be **basic enough** to be able to be **done exactly** and in **finite length of time**
  - Termination: must terminate after a finite number of steps
  - Input and Output



Steps involved in algorithm development

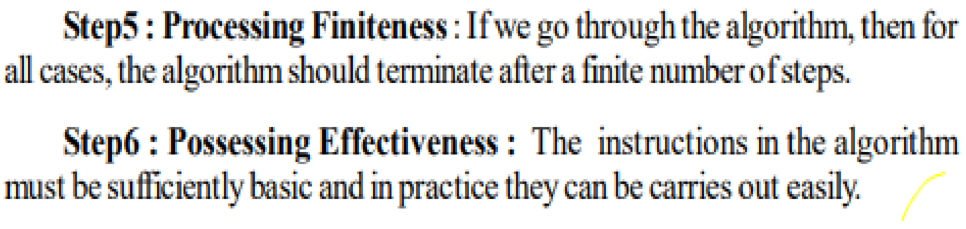
An algorithm can be defined as **"a complete, unambiguous, finite number of logical steps for solving a specific problem "**

Step1. Identification of input: For an algorithm, there are quantities to be supplied called input and these are fed externally. The input is to be indentified first for any specified problem.

Step2: Identification of output: From an algorithm, at least one quantity is produced, called for any specified problem.

Step3 : Identification the processing operations : All the calculations to be performed in order to lead to output from the input are to be identified in an orderly manner.

Step4 : Processing Definiteness : The instructions composing the algorithm must be clear and there should not be any ambiguity in them.



Step5 : Processing Finiteness : If we go through the algorithm, then for all cases, the algorithm should terminate after a finite number of steps.

Step6 : Possessing Effectiveness : The instructions in the algorithm must be sufficiently basic and in practice they can be carries out easily.

Characteristics of an algorithm

An algorithm must possess the following properties

1. **Finiteness:** An algorithm must terminate in a finite number of steps
2. **Definiteness:** Each step of the algorithm must be precisely and unambiguously stated
3. **Effectiveness:** Each step must be effective, in the sense that it should be primitive easily convert able into program statement) can be performed exactly in a finite amount of time.
4. **Generality:** The algorithm must be complete in itself so that it can be used to solve problems of a specific type for any input data.
5. **Input/output:** Each algorithm must take zero, one or more quantities as input data produce one or more output values.

An algorithm can be written in English like sentences or in any standard representation sometimes, algorithm written in English like languages are called Pseudo Code

Example 1. Suppose we want to find the average of three numbers, the algorithm is as follows

```
Step 1 Read the numbers a, b, c
Step 2 Compute the sum of a, b and c
Step 3 Divide the sum by 3
Step 4 Store the result in variable d
Step 5 Print the value of d
Step 6 End of the program
```

Example 2: Write an algorithm to calculate the simple interest using the formula. Simple interest = P*N* R/100. Where P is principle Amount, N is the number of years and R is the rate of interest.

```
Step 1: Read the three input quantities' P, N and R.
Step 2: Calculate simple interest as Simple interest = P* N* R/100
Step 3: Print simple interest.
Step 4: Stop.
```

Example 3. Area of Triangle: Write an algorithm to find the area of the triangle.

```
Let b, c be the sides of the triangle ABC and The included angle between the given sides.
 Step 1: Input the given elements of the triangle namely sides b, c and angle between the sides A.
Step 2: Area = (1/2) *b*C* sin A
Step 3: Output the Area Step
4: Stop.
```

Example 4. Write an algorithm to find the largest of three numbers X, Y, Z.

```
Step 1: Read the numbers X, Y, Z.
 Step 2: if (X > Y) Big = X else BIG = Y
Step 3: if (BIG < Z)
Step 4: Big = Z
Step 5: Print the largest number i.e. Big
Step 6: Stop.
```

Techniques used

Pseudocodes A way of expressing algorithms that uses a mixture of English phrases and indention to make the steps in the solution explicit. There are no grammar rules in pseudocode is not case sensitive. A mixture of English and formatting to make the steps in an algorithm explicit

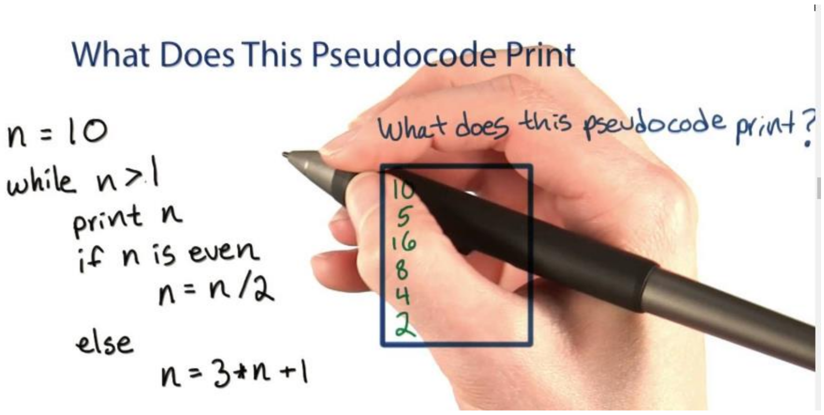
Algorithm vs. Pseudocode

An unambiguous specification of how to solve a problem

An informal high-level description of the operating principle of a computer program or other algorithm

Helps to simplify and understand the problem

A method of developing an algorithm


What Does This Pseudocode Print

```
n = 10
while n > 1
    print n
    if n is even
        n = n/2
    else
        n = 3*n + 1
```

What does this pseudocode print?

```
10
5
16
8
4
2
```

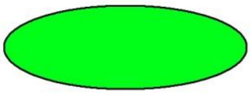
What is a flowchart?

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams.

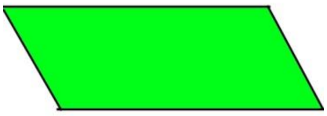
➔ PICTORIAL REPRESENTATION OF PROGRAM

Basic Symbols used in Flowchart Designs

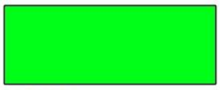
1. **Terminal:** The oval symbol indicates Start, Stop and Halt in a program's logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart.



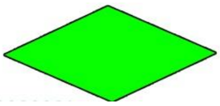
2. **Input/Output:** A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



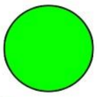
3. **Processing:** A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



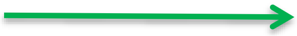
4. **Decision** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



5. **Connectors:** 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.



6. **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.



General Rules to draw flowchart

1. All boxes of the flowchart are connected with Arrows. (Not lines)
2. Flowchart symbols have an entry point on the top of the symbol with no other entry points. The exit point for all flowchart symbols is on the bottom except for the Decision symbol.
3. The Decision symbol has two exit points; these can be on the sides or the bottom and one side.

Advantages of Flowchart:

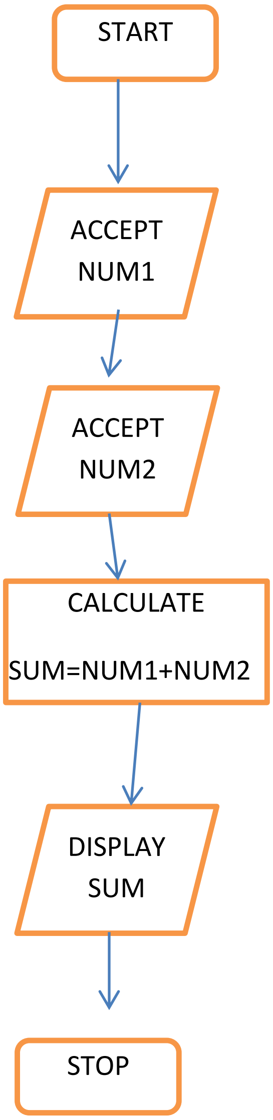
- Flowcharts are better way of communicating the logic of system.
- Flowcharts act as a guide for blueprint during program designed.
- Flowcharts helps in debugging process.
- With the help of flowcharts programs can be easily analyzed.
- It provides better documentation.
- Flowcharts serve as a good proper documentation.

Disadvantages of Flowchart:

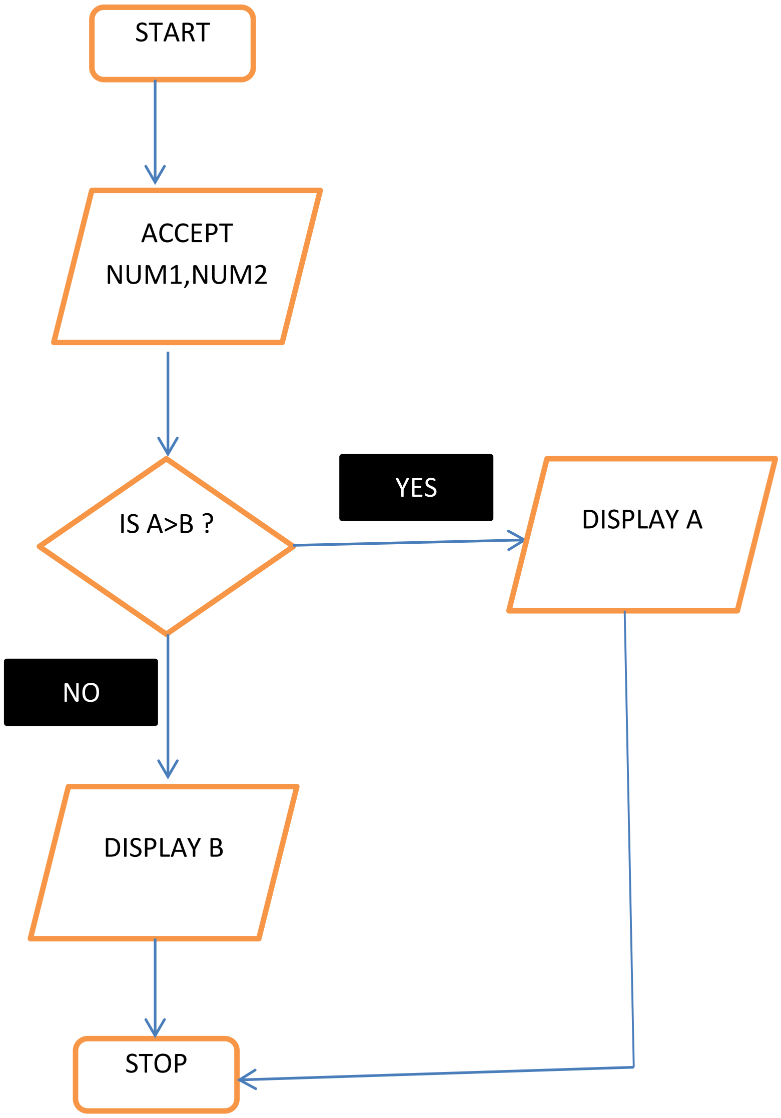
- It is difficult to draw flowchart for large and complex programs.
- In this there is no standard to determine the amount of detail.
- Difficult to reproduce the flowcharts.
- It is very difficult to modify the Flowchart.
- Takes more time to draw.

Imagine developing a detailed flowchart for a program containing 50000 lines or statements of instructions

Example 1 : ENTER TWO NUMBER FROM THE USER DISPLAY THE SUM OF THE NUMBERS

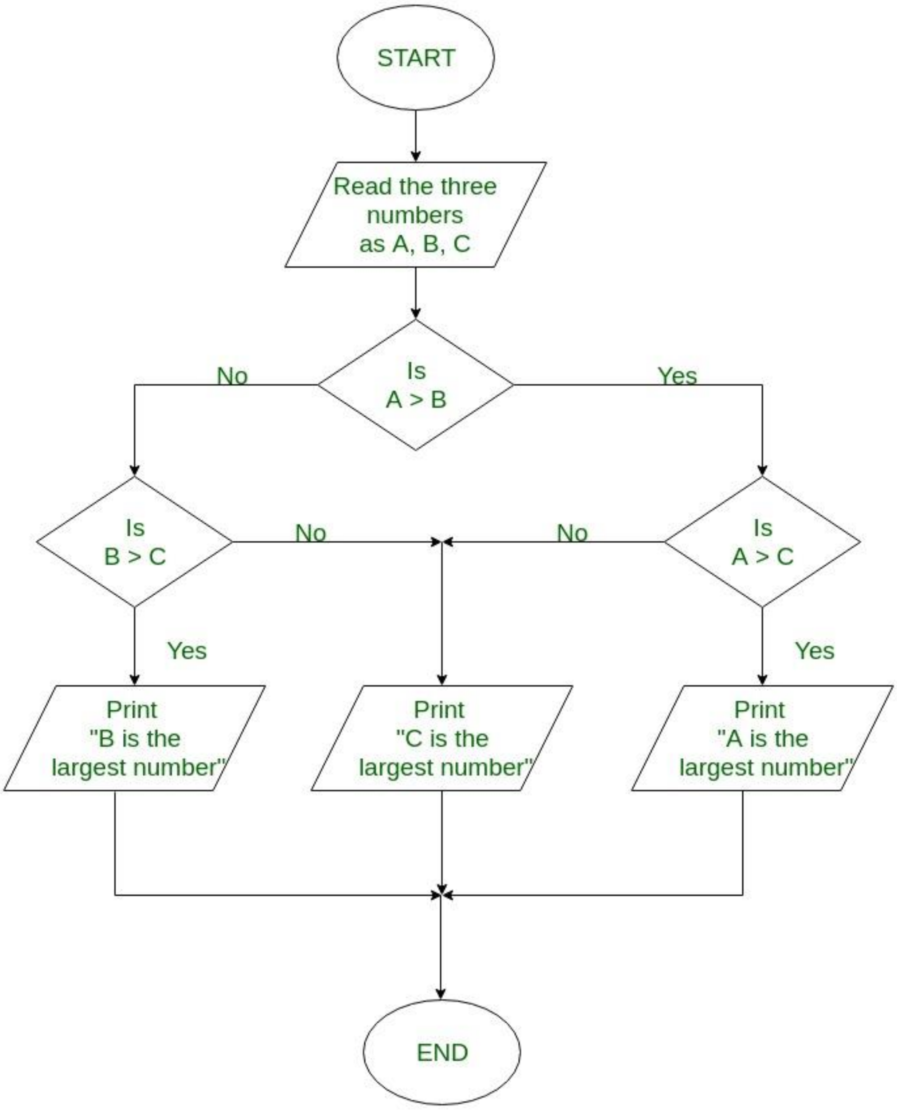


EXAMPLE 2: ENTER TWO NUMBER FROM THE USER , FIND OUT THE LARGEST VALUE

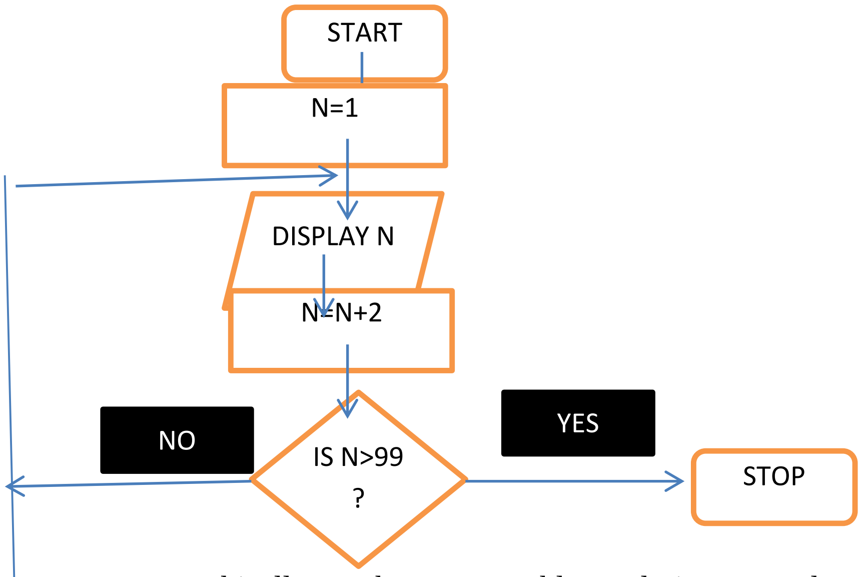


EXAMPLE 3: ENTER THREE NUMBER FROM THE USER , FIND OUT THE LARGEST VALUE

NUM1, NUM2,NUM3
CASE 1 : 5, 7, 3
5> 7 NO , 7> 3 YES 7
NUM1 > NUM2 NO
NUM2 > NUM3 YES
PRINT NUM2
CASE2: 5,7,8
5>7 NO, 7 > 8 NO -> 8
NUM1> NUM2 -> NO
NUM2> NUM3-> NO
PRINT NUM3
CASE 3: 8,7,3
8>7 YES , 8> 3 YES PRINT 8
NUM1> NUM2 YES , NUM1>NUM3 YES PRINT NUM1

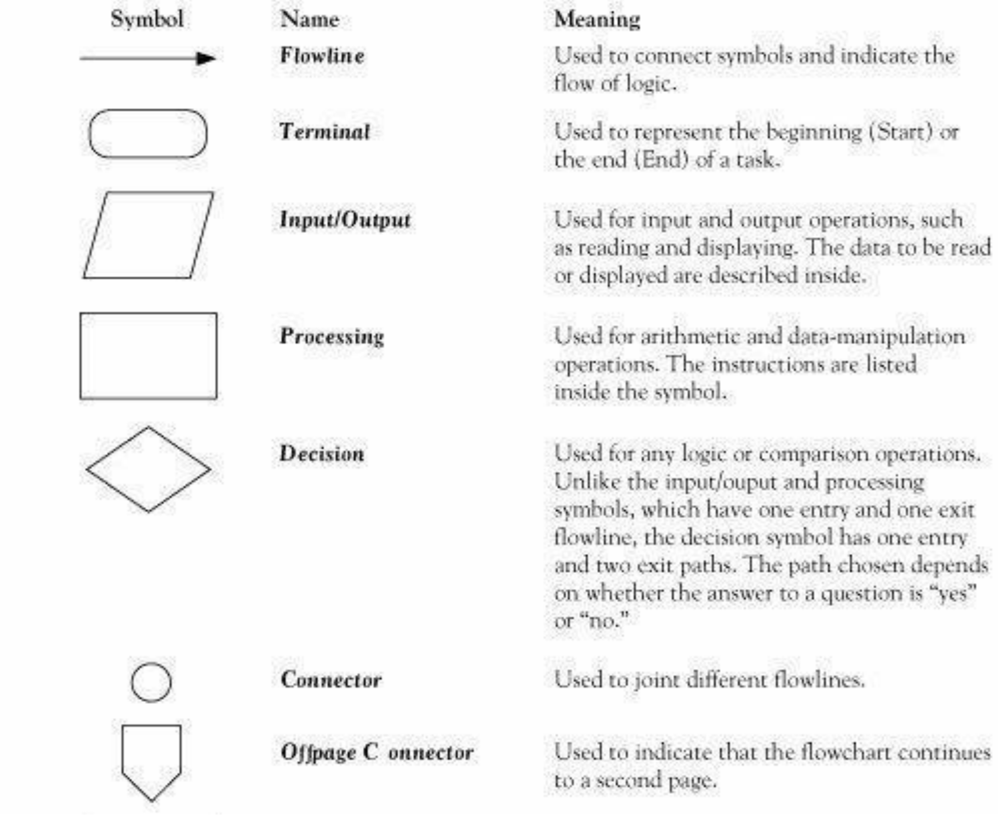

START
Read the three numbers as A, B, C
Is A > B
No
Yes
Is B > C
No
No
Is A > C
Yes
Yes
Print "B is the largest number"
Print "C is the largest number"
Print "A is the largest number"
END

EXAMPLE 4: PRINT ALL ODD NUMBERS FROM 1 TO 100



Thus a flowchart represents graphically step by step a problem solution or an algorithm using specific symbols. Each symbol has name.

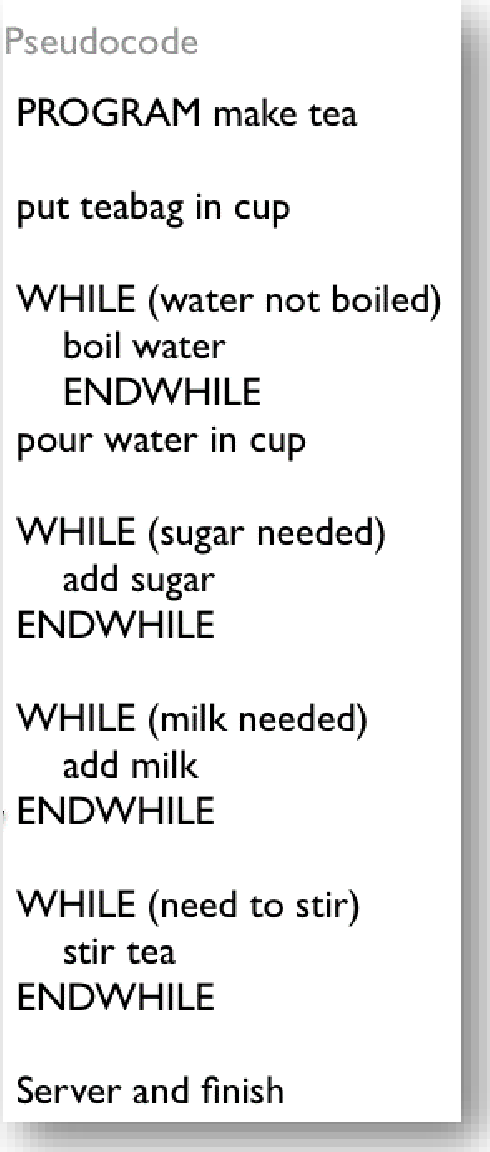
Symbols used to Draw Flowchart



Symbol	Name	Meaning
→	***Flowline***	Used to connect symbols and indicate the flow of logic.
▭	***Terminal***	Used to represent the beginning (Start) or the end (End) of a task.
▱	***Input/Output***	Used for input and output operations, such as reading and displaying. The data to be read or displayed are described inside.
□	***Processing***	Used for arithmetic and data-manipulation operations. The instructions are listed inside the symbol.
◇	***Decision***	Used for any logic or comparison operations. Unlike the input/ouput and processing symbols, which have one entry and one exit flowline, the decision symbol has one entry and two exit paths. The path chosen depends on whether the answer to a question is "yes" or "no."
○	***Connector***	Used to joint different flowlines.
⬠	***Offpage C onnector***	Used to indicate that the flowchart continues to a second page.

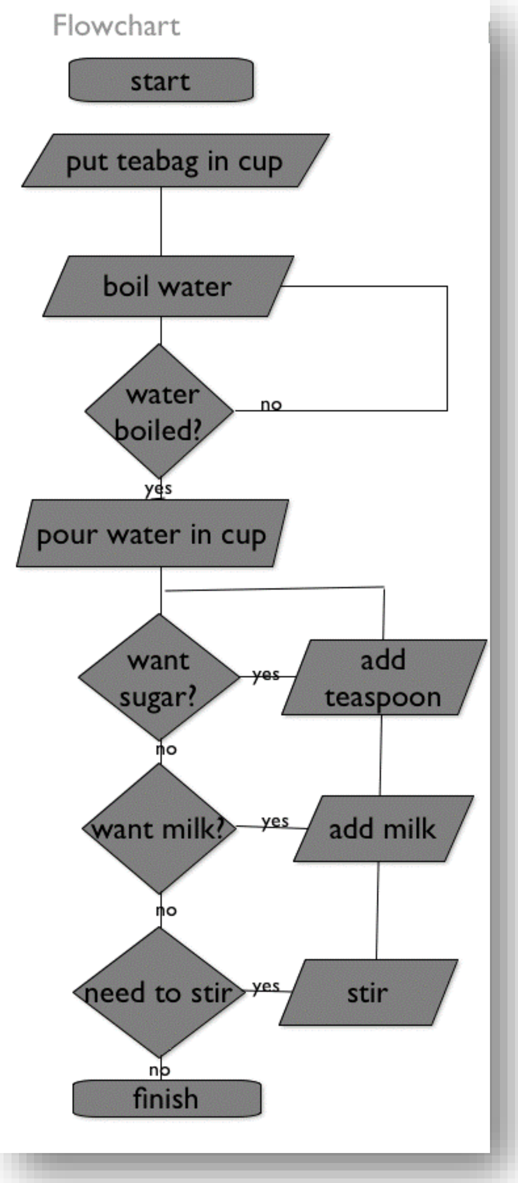
Examples

Write a pseudocode and draw a flow chart to make a cup of tea

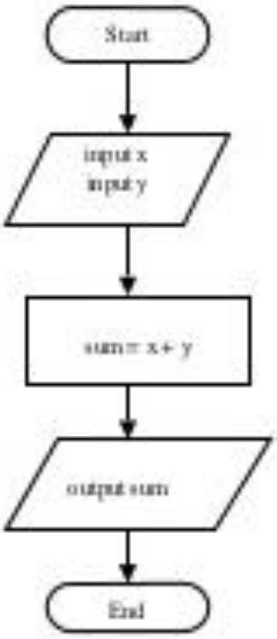


Pseudocode

```
PROGRAM make tea

put teabag in cup

WHILE (water not boiled)
    boil water
    ENDWHILE
pour water in cup

WHILE (sugar needed)
    add sugar
ENDWHILE

WHILE (milk needed)
    add milk
ENDWHILE

WHILE (need to stir)
    stir tea
ENDWHILE

Server and finish
```

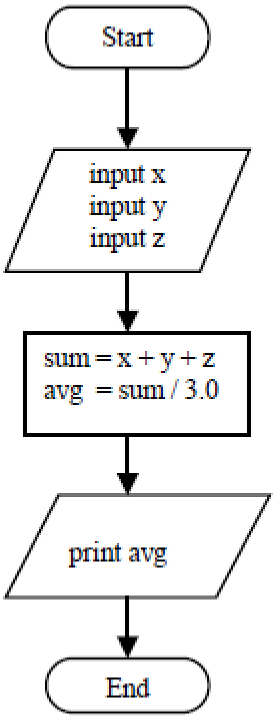


Sum of 2 Numbers - sequence



```
Begin
    input x, y
    sum = x + y
    print sum
End
```

Average of 3 Numbers - sequence

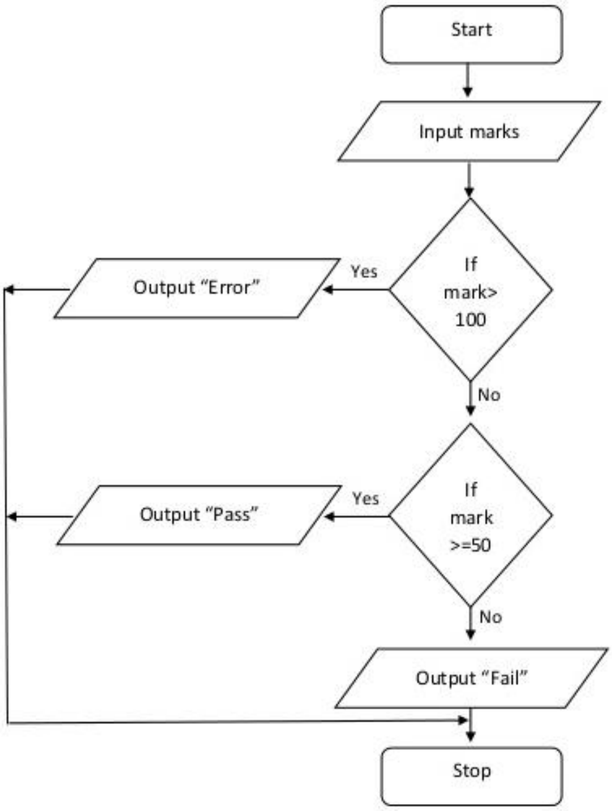


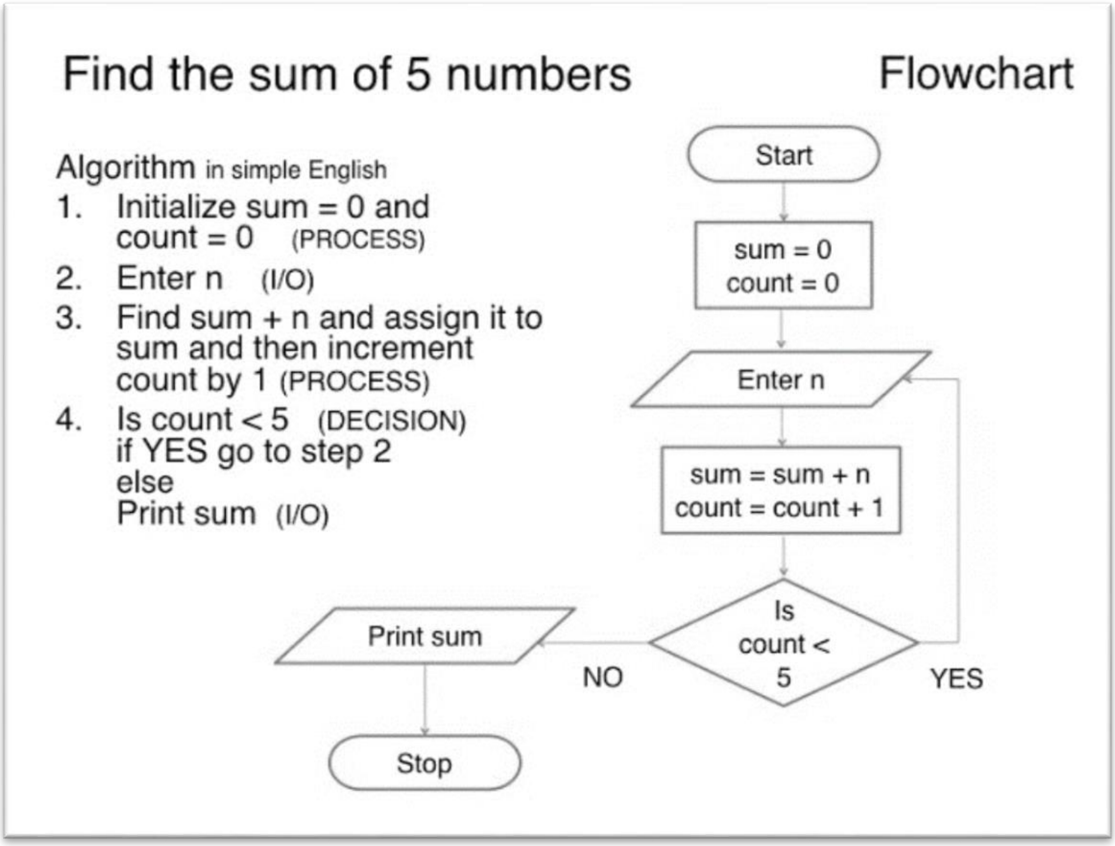
```
Begin
    input x
    input y
    input z
    sum = x + y + z
    avg = sum / 3.0
    print avg
End
```

Pseudocode and Flowchart

To read a marks from student and display "pass" if the marks greater than or equal to 50. "fail" otherwise.

```
Begin
        Read marks
        If marks > 100 Then
                Display "Error"
        Else:
        If marks >= 50 Then
                Display "Pass"
        Else:
                Display "Fail"
        End if
        End if
End
```



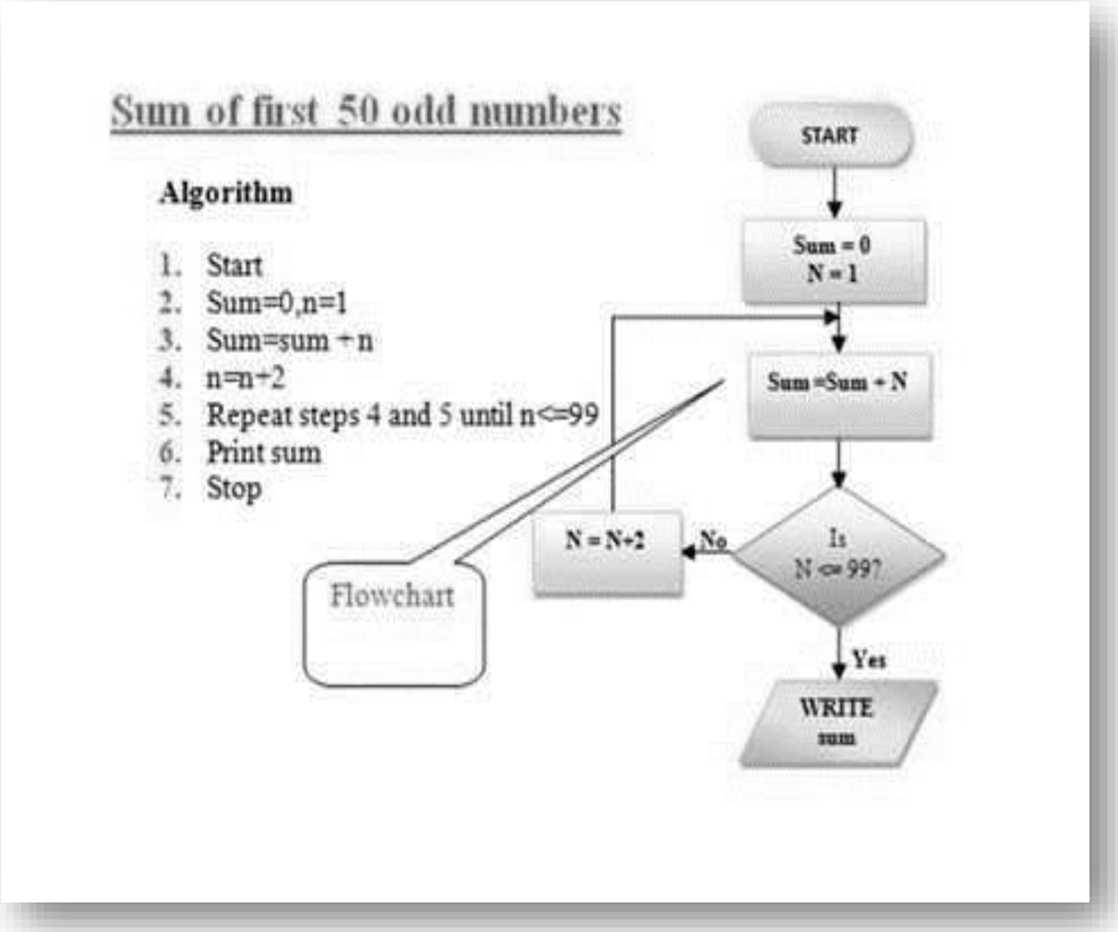


Find the sum of 5 numbers

Flowchart

Algorithm in simple English

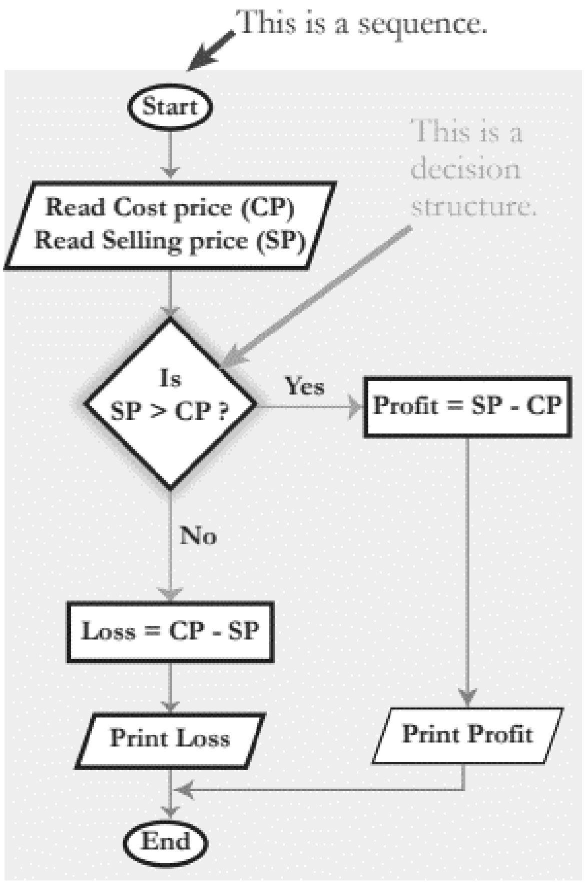
1. Initialize sum = 0 and count = 0 (PROCESS)
2. Enter n (I/O)
3. Find sum + n and assign it to sum and then increment count by 1 (PROCESS)
4. Is count < 5 (DECISION)
   if YES go to step 2
   else
   Print sum (I/O)



Sum of first 50 odd numbers

Algorithm

1. Start
2. Sum=0,n=1
3. Sum=sum + n
4. n=n+2
5. Repeat steps 4 and 5 until n<=99
6. Print sum
7. Stop



Finding profit or loss when CP = 325 and SP = 458

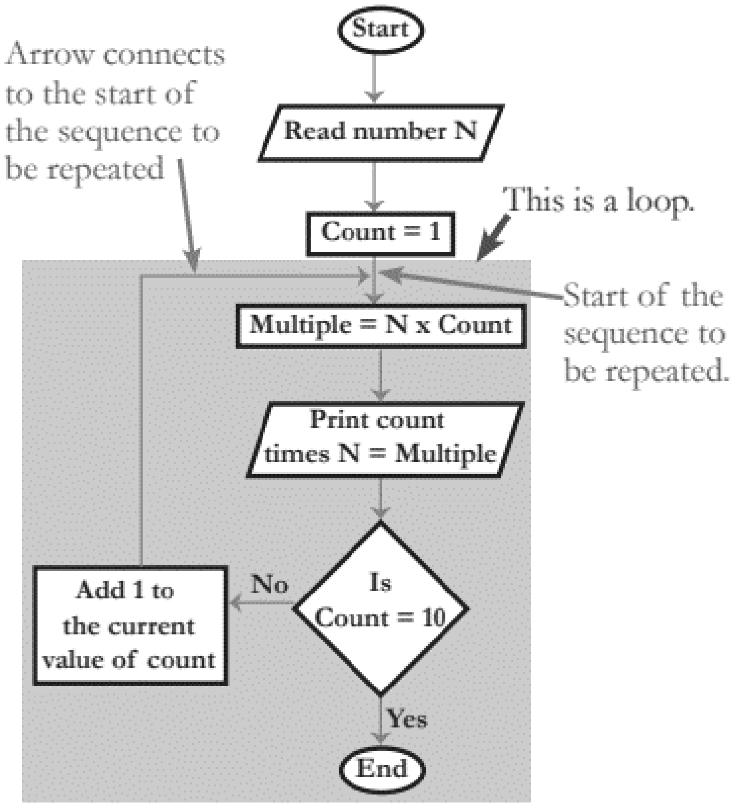
Start

Read CP=325
Read SP= 458

Condition:
Is 458 > 325? — Profit= 458-325

Profit= Rs. 133

End


Start
Arrow connects to the start of the sequence to be repeated
Read number N
This is a loop.
Count = 1
Multiple = N x Count
Start of the sequence to be repeated.
Print count times N = Multiple
Add 1 to the current value of count
No
Is Count = 10
Yes
End

Multiplication table of 12

Start

N = 12

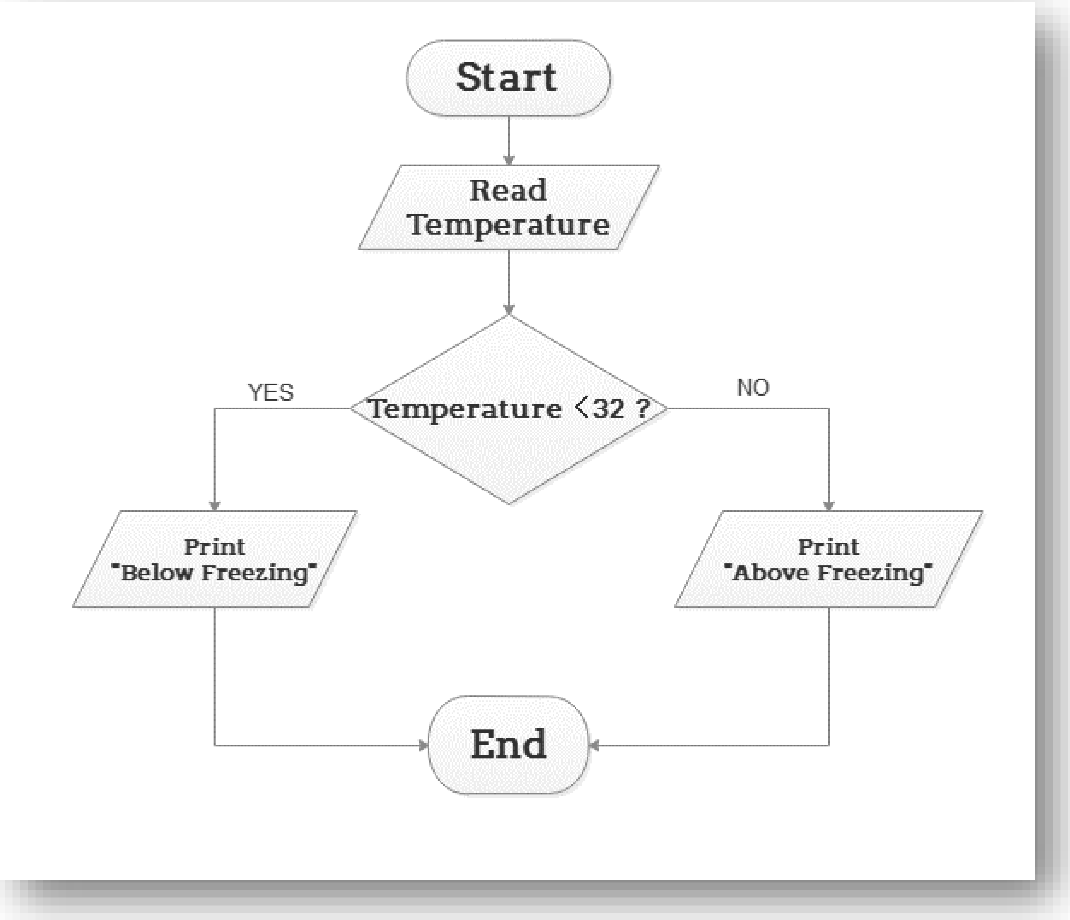
Count =1

12 * 1 = 12
Count = 1+1 = 2
12 * 2 = 24
Count = 2+1 = 3
......
Count = 9+1 = 10
12 * 10 = 120

Count =10

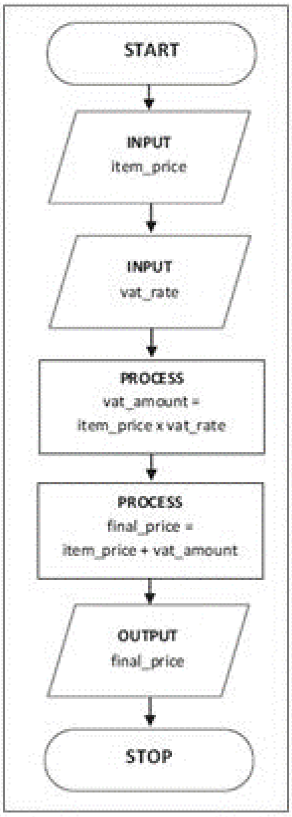
End

Check The Temperature freezing point



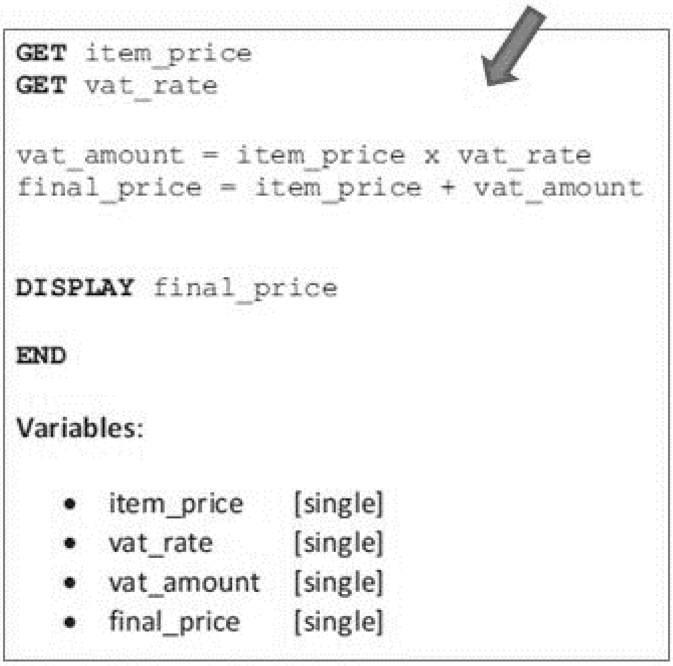
Writing Algorithms

Problem: Calculate the Sales Tax (VAT) of an item and then using it to work out the final price



1. Flowchart

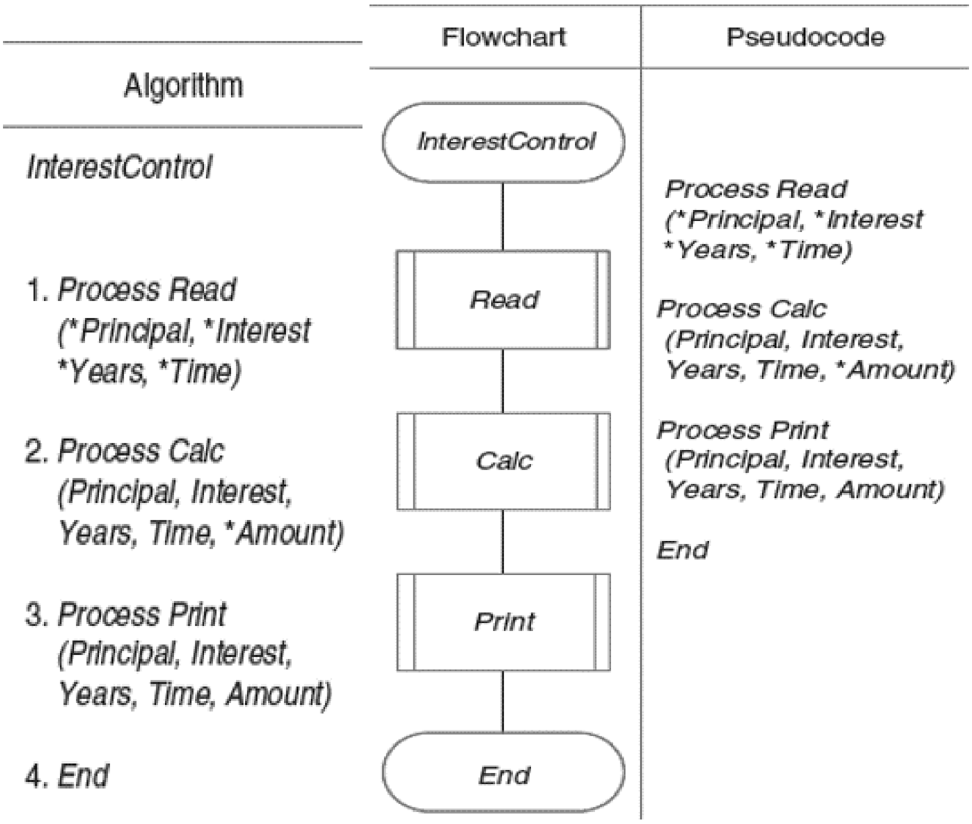
2. Pseudocode



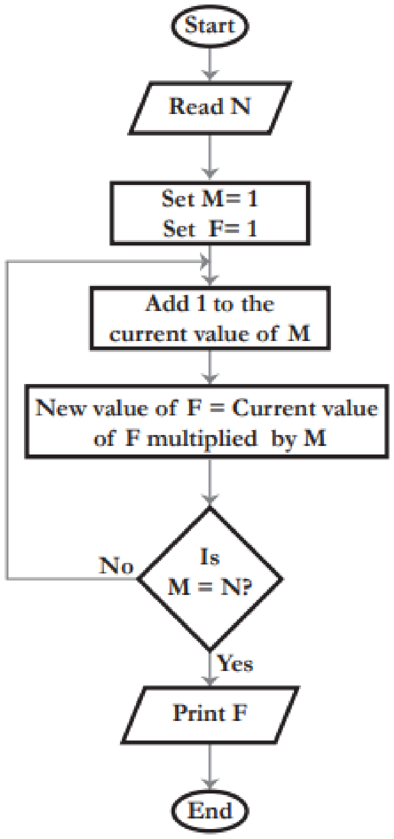
```
GET item_price
GET vat_rate

vat_amount = item_price x vat_rate
final_price = item_price + vat_amount


DISPLAY final_price

END
```

Variables:

item_price	[single]
vat_rate	[single]
vat_amount	[single]
final_price	[single]



Algorithm	Flowchart	Pseudocode
InterestControl	*InterestControl*	
1. *Process Read (*Principal, *Interest *Years, *Time)*	*Read*	*Process Read (*Principal, *Interest *Years, *Time)*
2. *Process Calc (Principal, Interest, Years, Time, *Amount)*	*Calc*	*Process Calc (Principal, Interest, Years, Time, *Amount)*
3. *Process Print (Principal, Interest, Years, Time, Amount)*	*Print*	*Process Print (Principal, Interest, Years, Time, Amount)*
4. *End*	*End*	*End*


Start
Read N
Set M= 1
Set F= 1
Add 1 to the current value of M
New value of F = Current value of F multiplied by M
Is M = N?
No
Yes
Print F
End

Finding factorial of 10

Start

N = 10

M = 1

F = 1

F = 1 * 1 = 1; M < 10; M = 1 +1 = 2

F = 1 * 2 = 2; M < 10; M = 2 +1 = 3

F = 2 * 3 = 6; M<10; M = 3 + 1 = 4

F = 6 * 4 = 24; M<10; M = 4 +1 = 5

....

M<10; M = 9 + 1 = 10

F = 362880 * 10 = 3628800; M = 10

Factorial of 10 = 3628800

End

Playing Snakes and Ladder

Playing snakes and ladders between two players

Start

Throw the dice: the number indicated by dice is 3.

Move the coin: through 3 blocks on the board.

Landed on snake head?: No.

Landed on the bottom of the ladder?: Yes (See the snakes and ladders board given).

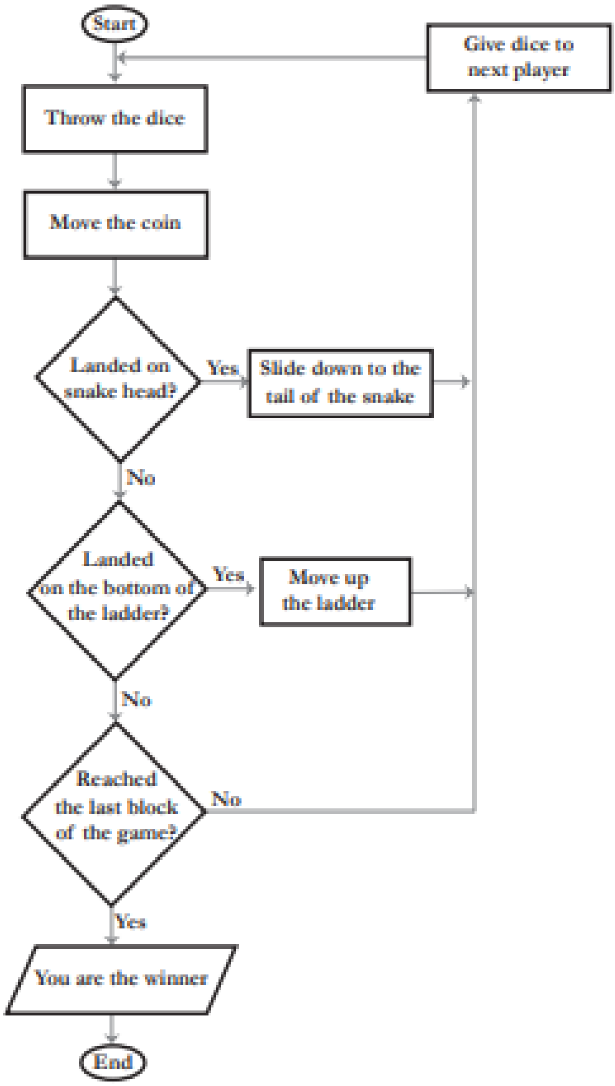
Move up the ladder: reached block 14 on the board.

Reached the last block of the game?: No.

Give dice to next player.

Enter the loop till the last block of the game is reached.

End



Exercise

1. Explain steps involve in drawing of a flowchart.
2. Explain uses of Flowchart.
3. Write algorithm for the problem given below
   Ramsh goes to market for buying some fruits and vegetables. He has Rs 500 with him for marketing. From a shop he purchases 2.0 kg Apple priced Rs. 50.0 per kg, 1.5 kg Mango priced Rs.35.0 per kg, 2.5 kg Potato priced Rs.10.0 per kg, and 1.0 kg Tomato priced Rs.15 per kg. He gives the currency of Rs. 500 to the shopkeeper. Find out the amount shopkeeper will return to Ramshewak. and also tell the total item purchased.
4. Find factorial of N?
5. Draw a flowchart to find the sum of first 100 natural numbers. 9) Draw a flowchart to find the largest of three numbers x, y and z.
6. Write an Algorithm and draw flowchart for determining a number is prime or not prime number or not
7. Draw a flowchart which generates first 50 items of the Fibonacci series: 1, 1, 2, 3, 5, 8,…?
8. Design an algorithm to convert a decimal number, n, to binary format?

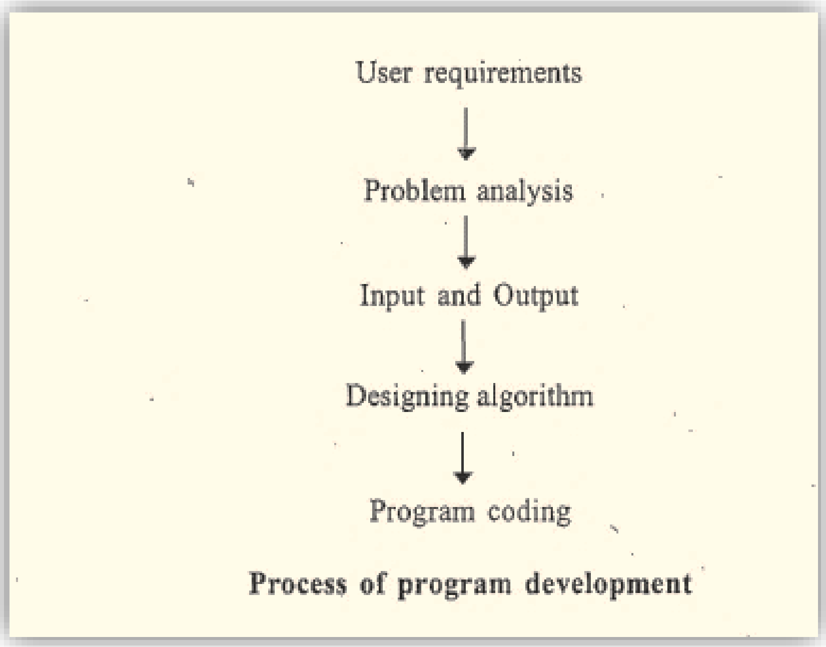
What is Programming?

Computer programming is a medium for us to communicate with computers just like we use Hindi or English to communicate with each other, programming is a way for us to deliver our instructions to the computer.

- **Programming** Planning or scheduling the performance of a task or an event

Computer programming: The process of specifying the data types and the operations for a computer to apply to data in order to solve a problem

- **Data** Information in a form a computer can use
- **Information** Any knowledge that can be communicated
- **Data type** The specification of how information is represented in the computer as data and the set of operations that can be applied to it



Process of program development

How to Do Programming?

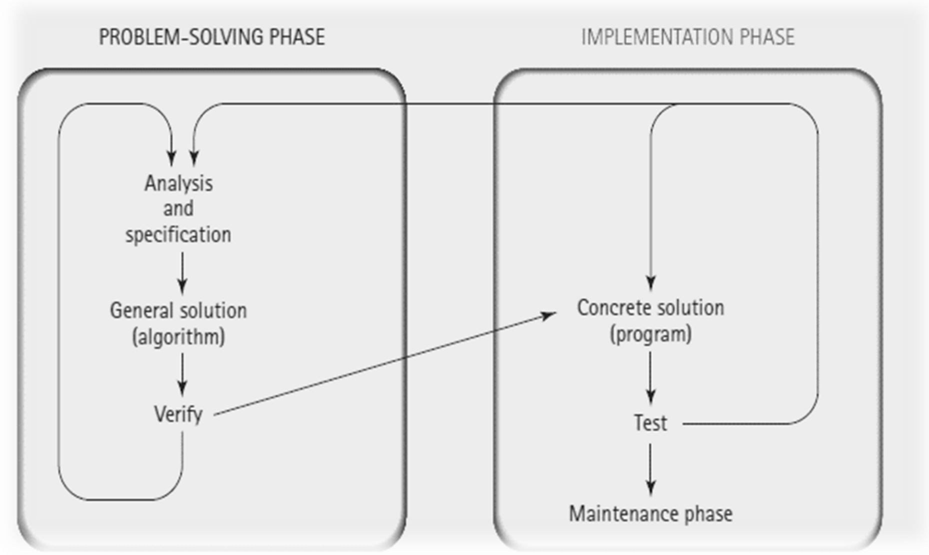
A computer is not intelligent. It cannot analyze a problem and come up with a solution.A Person who (the programmer) analyze the problem, develop the instructions for solving the problem, and then have the computer carry out the instructions.

To write a program for a computer to follow, we must go through a Problem solving phase/ process:

1. **Problem-Solving Phase**
1. Analysis and Specification. Understand (define) the problem and what the solution must do.
2. General Solution (Algorithm). Specify the required data types and the logical sequences of steps that solve the problem.
3. Verify. Follow the steps exactly to see if the solution really does solve the problem.
2. **Implementation Phase**
4. Concrete Solution (Program). Translate the algorithm (the general solution) into a programming language.
5. Test. Have the computer follow the instructions. Then manually check the results. If you find errors, analyze the program and the algorithm to determine the source of the errors, and then make corrections.

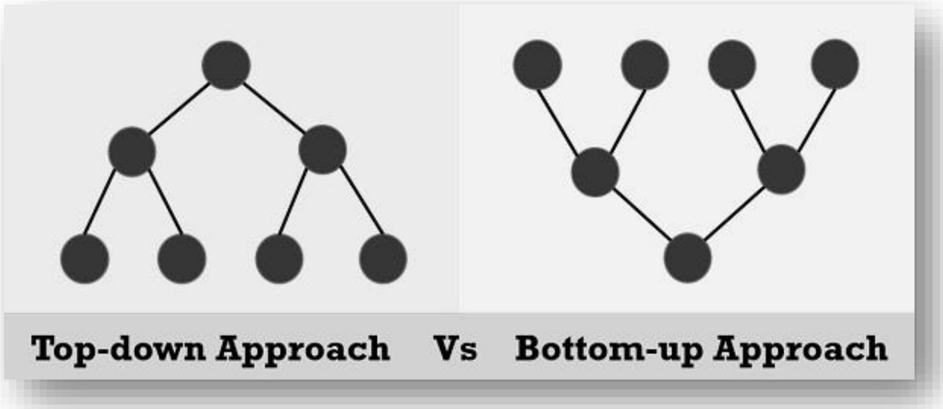
   Once a program has been written, it enters a third phase: maintenance.

   Maintenance Phase
6. Use. Use the program.
7. Maintain. Modify the program to meet changing requirements or to correct any errors that show up while using it.


PROBLEM-SOLVING PHASE
IMPLEMENTATION PHASE
Analysis and specification
General solution (algorithm)
Verify
Concrete solution (program)
Test
Maintenance phase

Design Methods: Designing is the first step for obtaining solution of a given problem. The purpose of designing is ~o represents the solution for the system. It is really difficult to design a large system because the complexity system cannot be represented easily. So various methods have been evolved for designing.

Top-Down Design:. 'Every system has several hierarchies of components. The top-level component represents the whole tern. Top-Down design method starts from top-level component to lowest level (bottom) component. this design method, the system is divided into some major components.



Difference between Top-down and Bottom-up Approach

Top-Down Approach	Bottom-Up Approach
In this approach problem is divided into smaller units and then solve it.	Starts from bottom, focus on data first (like Object Oriented approach) solving small modules and adding them up together.
May contain redundant information.	Redundancy can easily be eliminated.
A well-established communication is not required.	Communication among steps is mandatory.
The individual modules are thoroughly analyzed.	Works on the concept of data-hiding and encapsulation.
Structured programming languages **such as C uses top-down approach**.	OOP languages like C++ and Java, etc. uses bottom-up mechanism.
Relation among modules is not always required.	The modules must be related for better communication and work flow.
Primarily used in code implementation, test case generation, debugging and module documentation.	Finds use primarily in testing.

References

1. Hanly J. R. and Koffman E. B.,“Problem Solving and Program Design in C”, Pearson Education.
2. https://www.brainkart.com/article/Examples-algorithms--pseudo-code,-flow-chart,-programming-language_35900/
3. https://www.geeksforgeeks.org/an-introduction-to-flowcharts/
4. https://www.it.iitb.ac.in/~vijaya/ssrvm/dokuwiki/media/s6_l7_20jan.pdf