Logic Building & Problem Solving

Harshita Maheshwari

Agenda for Today's Session

- What is Problem?
- What is Problem Solving??
- Problem Solving steps
- Computational Problems
- Difficulties with Problem Solving
- Need of Algorithm
- Algorithm & Program
- Properties of Algorithm
- Various Patterns in Algorithms
- Flowchart and Pseudo Code
- Flowchart Symbol
- Pseudo code
- Examples
- What is logical thinking
- puzzles

What is Problem??

Problem means a issue/question proposed for solution or consideration .

Problems can be may types like –

Social

Political

Computational

Etc..

What is Problem Solving??

The process of finding solutions to difficult or complex issues.

Problem-solving is a process of solving any kind of problem.

Steps for Problem Solving

1. **Identify the problem:** Identify what the problem actually is
2. **Analysis the problem :** Breakdown the problem into subproblems.
3. **Algorithm Design:** Design step by step procedures of solving problems.
4. **Program Development :** Implement algorithm as computer program.
5. **Debugging and Testing :** Find out and remove all possible errors.
6. **Documentation :** Prepare documents that describes solution.

Examples of Computational Problems

Difficulties with Problem Solving

- Lack of training or natural ability for problem solving.
- Fear of decision making
- Inadequate use of the problem solving steps
- Personal biases inserted into the process
- Misunderstanding the problem
- Inability to explain or abstract the problem in a form that is usable by a computer.
- etc…

Solutions

- Patience and practice
- Don’t give up
- Theory knowledge
- Pen and Paper
- Dry run code
- Do as many questions as you can
- Be regular
- Go from basic to advanced
- Don’t see the solution until you haven't solved it

But first and the most important thing is -> How to build logic????

To learn how to write logic to solve a problem we have to learn the algorithms.

Algorithms


algorithms
Insertion
binary
heapsort
Analysis
heap
trees
hashing
quicksort
queues
time
DFS
linear
search
sort
sorting
solving
randomized
MST
order
statistics
tables
Selection
Probabilistic
recurrences
augmented
MinimumSpanningTrees

Algorithm

Algorithm is:

1. A step by step problem solving procedure.
2. A sequence of instruction that tell how to solve a particular problem.
3. A set of instruction for solving a problem, especially on a computer.
4. A computable set of steps to achieve a desired result.

We can say that algorithm has something to do with defining generalized processes to get **"output" from the "input".**

We can conclude that:
Algorithm a step by step problem solving process in which solution is arrived in a finite amount of time.

OR

An algorithm is a finite lost of sequential steps specifying actions that if performed result in solution of a specific problem.

Algorithm & Logical Thinking

To develop the algorithm, the programmer needs to ask:

1. **What data has to be fed into the computer??**

2. **What information do I want to get out of the computer.**

3. **Logic : Planning the processing of the program.**

Logic means the instructions that cause the input data to be turned into the desired output data.

Properties of An Algorithm

infoway TECHNOLOGIES

1. **Finiteness** -> must terminate after a finite number of steps.
2. **Definiteness**-> Every step in algorithm must be clear as to what it is supposed to do and how many times it is expected to be executed.
3. **Input** -> It must have zero or more input.
4. **Output** -> It must produce at least one output.
5. **Effectiveness**-> It must be correct and efficiently solve the problem for it is designed.

Algorithm Example

Recipe for making 2 cups of Tea:-

1. Take a medium sized vessel.
2. Measure 1 cups of water and pour into vessel.
3. Place the vessel on the burner and switch on the burner to medium flame.
4. Add 2 spoons of tea powder.
5. Once the water comes to boil add sugar
6. A minute later add 1 cups milk
7. In another minute pour the contents of the vessel into two cups after passing the tea through a strainer.
8. Serve the hot tea.

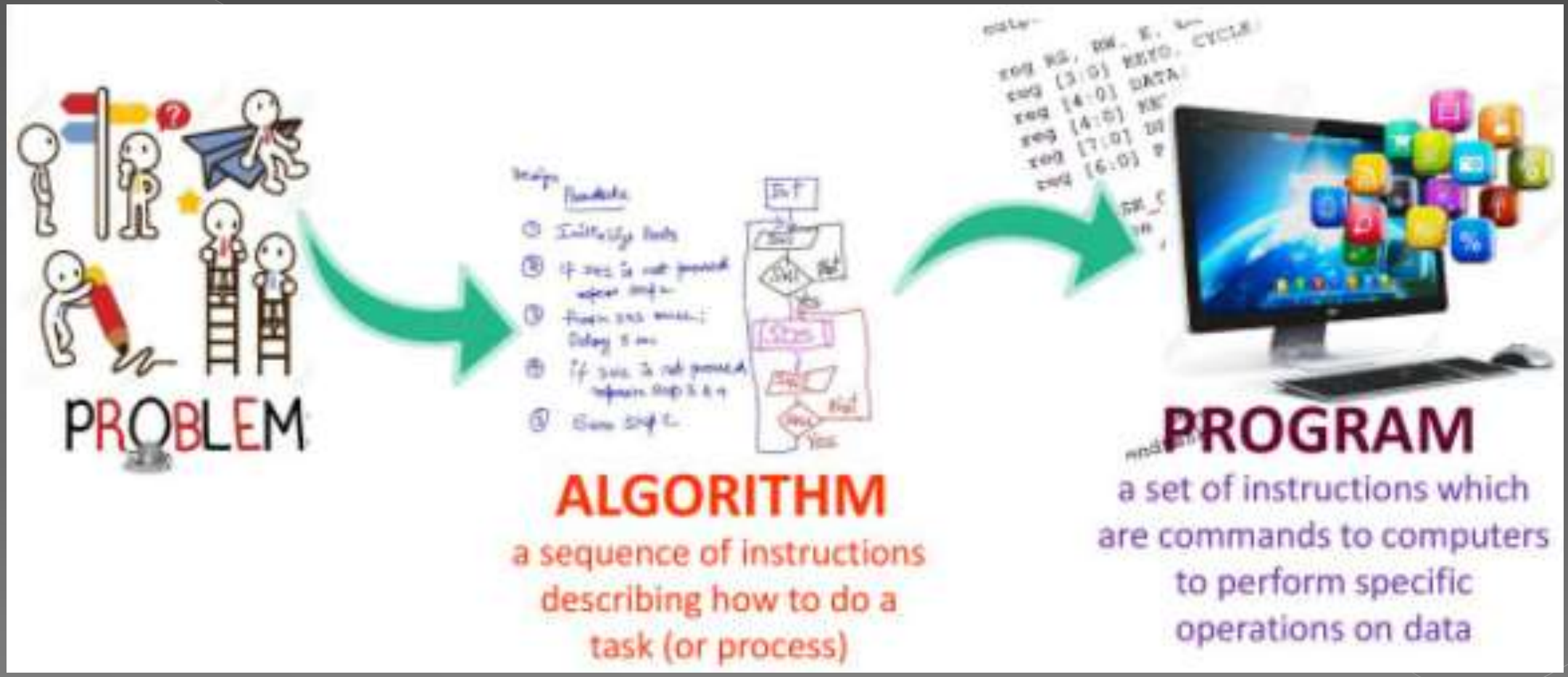
Algorithm Representation

Following notations are used for representing algorithms:

Pseudo Code: Textual representation in English statements to present the solution to a problem.

Flow Chart : graphical representation of an orderly step by step solution to a problem.

From Algorithm to Program



Algorithm vs. Program

Algorithm	Program
Talking to humans, easy to understand	Talking to computer(complier)
Plain English	Can be regarded as a "formal expression" of an algorithm
Step by step problem solving procedure	A program is set of instructions which are commands to computer to perform specific operations on data.
A sequence of instruction that tell how to solve a particular problem.	Instructions within a program are organized in such way, when the program is run on a computer; it result in the solution of a problem.
Written in pseudo code and flow chart	Written in any programming language

Various Patterns in Algorithm

1. Sequential :- Executes the statements in the order in which they appear in the algorithm

2. Selectional(conditional):- Recursive Controls the flow of statements execution based on some condition

3. Iterational (loop):- Used when a part of the algorithm is to be executed several times

Pseudo Code

An outline of a program , written in a form that can be easily be converted into real programming statements.

It resembles the actual program that will be implemented later. However, it cannot be compiled nor executed.

Pseudocode normally codes the following actions:

- Initialization of variables
- Assignment of values to the variables
- Arithmetic Operations
- Relational Operations

Pseudo Code Structure

```
BEGIN
        statement1;
        statement2;
        .
        .
        .
END
```

Flowchart

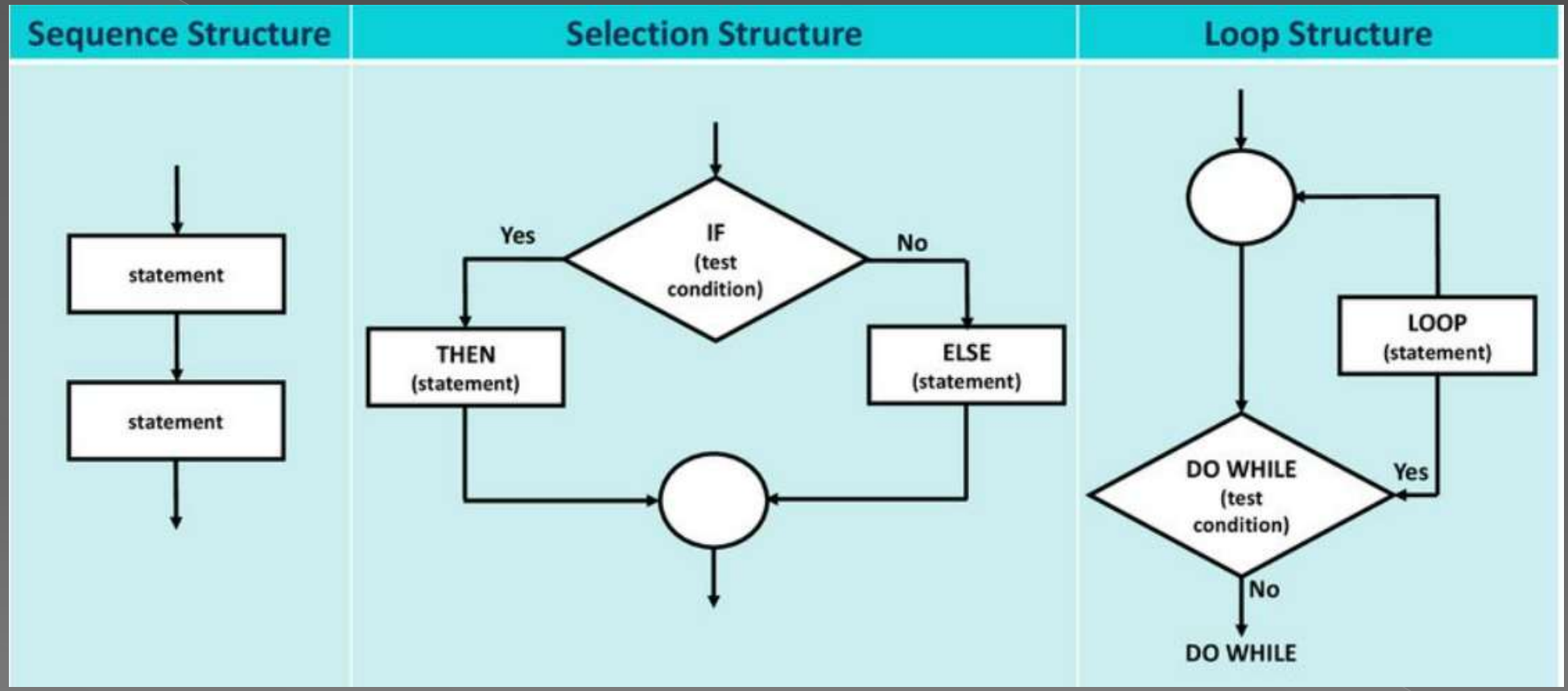
A flowchart is a type of diagram that represents a workflow or process.

A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

Flowchart Logic Structure



SEQUENCE

When we write programs, we assume that the computer executes the program starting at the beginning and working its way to the end.

We call this SEQUENCE.

```
Statement1;
Statement2;
Statement3;
Statement4;
Statement5;
Statement6;
Statement7;
Statement8;
```

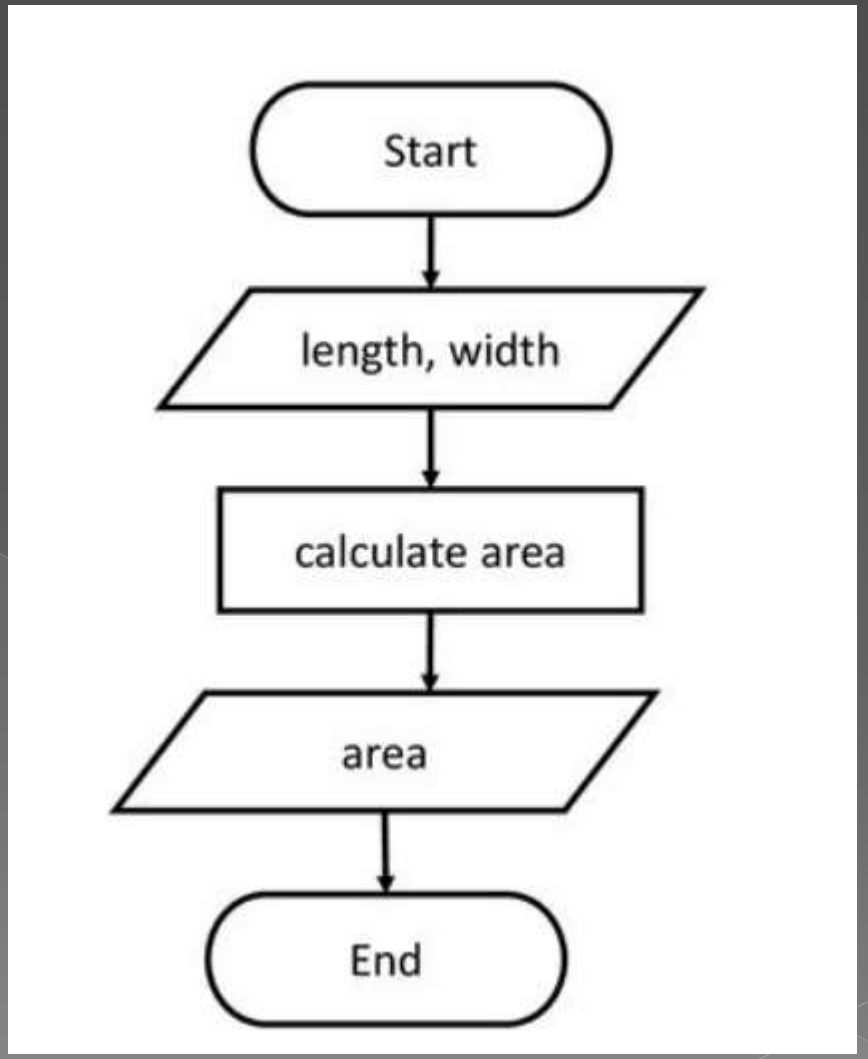
```
Organise everything together;
Plug in kettle;
Put teabag in cup;
Put water into kettle;
Wait for kettle to boil;
Add water to cup;
Remove teabag with spoon/fork;
Add milk and/or sugar;
Serve;
```

Sequential Logic Problem

Design an algorithm to find the area of a rectangle

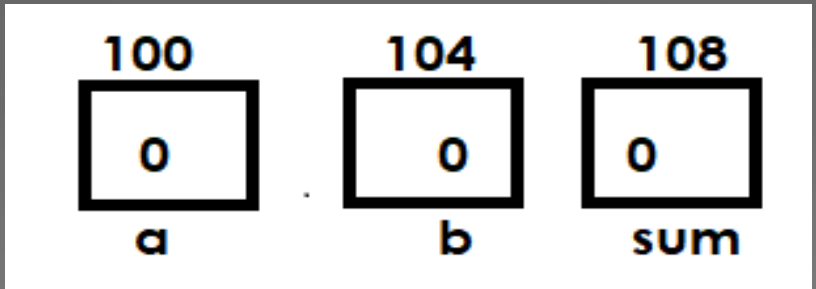
The formulas: area = length * width

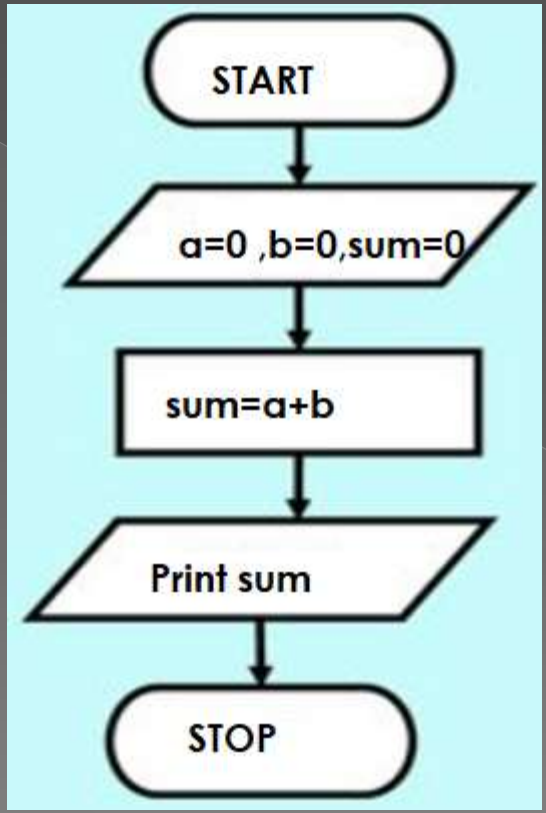
Input	Process	Output
Input variable: length width	Processing item: area Formula: area = length x width Step / Solution algorithm: get input calculate area display output	Output: area



Algorithm to find sum of two numbers.

```
START/BEGIN
1. a<-0, b<-0, sum<-0
2. Read a and b
3. sum <- a+b
4. Display sum
STOP/END
```





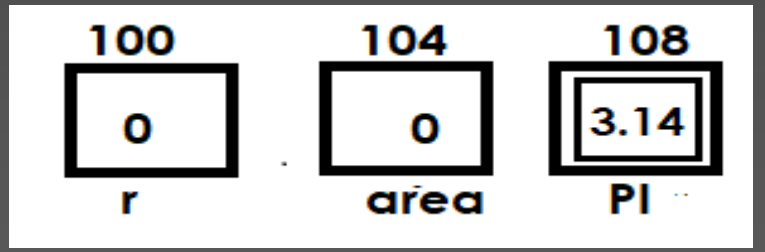
Testing the Logic:-

Test	a	b	sum
1	0	0	0
2	10	20	0
3	10	20	30
4	10	20	30

output

Algorithm to find Area of Circle -

```
START
1. r<-0, area<-0, PI<-3.14
2. Read r
3. area <- PI*r*r
4. Print "Area of Circle:",area
STOP
```



Testing the Logic:-

Test	R	Area	PI
1	0	0	3.14
2	2	0	3.14
3	2	12.56	3.14
4	2	12.56	3.14

Output- Area of Circle: 12.56

SELECTION

If we want to make a choice.

For ex – do we want to add sugar or not to the tea?

We call this SELECTION.

So , we could state this as –

```
IF (sugar is required)
    THEN add sugar;
    ELSE don't add sugar;
ENDIF;
```

Decision Making/Selection Problems

Algorithm to print message only if number is 100.

```
IF (<CONDITION>)
    THEN <Statements>;
    ELSE <Statements>;
ENDIF;
```

```
BEGIN
    num <- 0
    Read num
    IF (num = 100) THEN
        Print "Congratulations"
    ENDIF
END
```

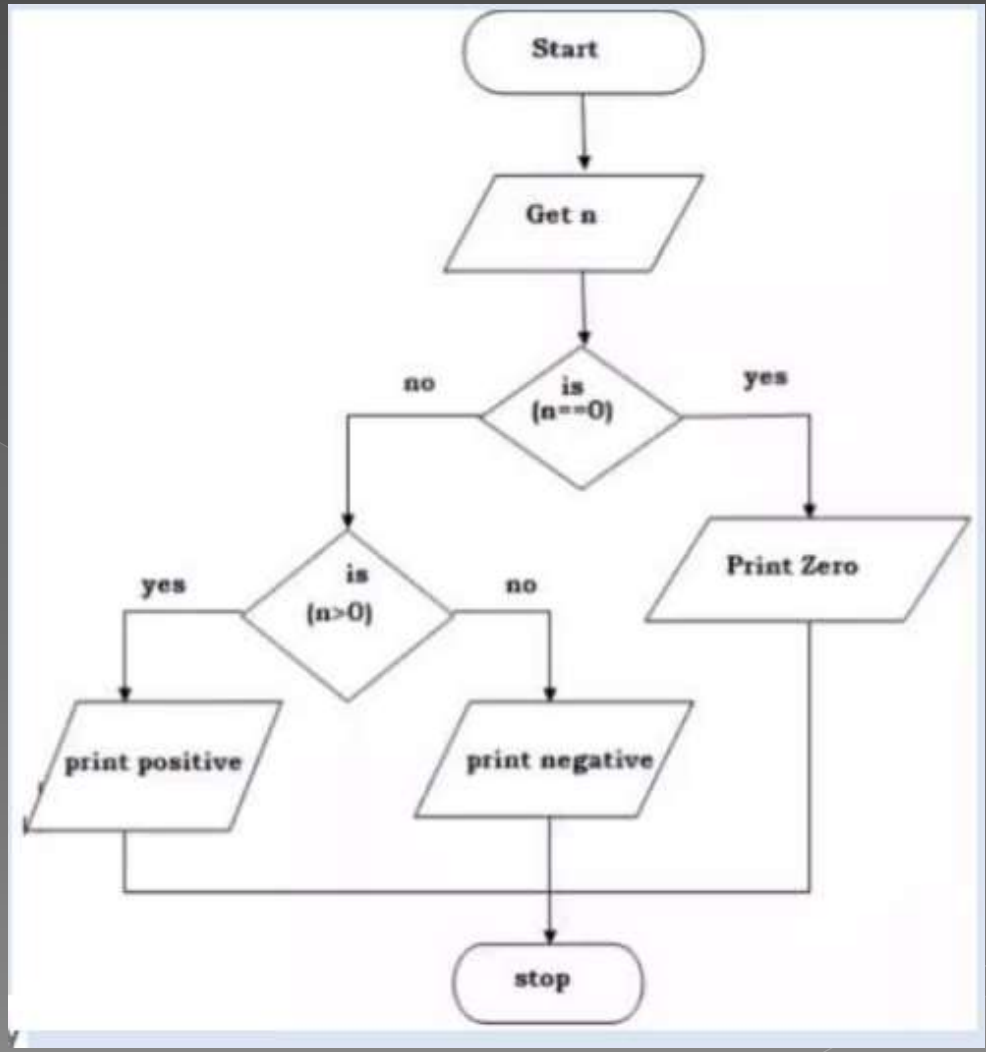
	Num (1st test)	2nd test
1	0	0
2	35	100
3		Msg printed

Algorithm to find given number is even or odd

```
BEGIN
    num <- 0
    Read num
    IF (num mod 2 = 0)
    THEN
       {
          Print "Number is Even"
       }
     ELSE
       {
          Print "Number is odd"
       }
    ENDIF
END
```

Algorithm - number is positive , Negative or zero

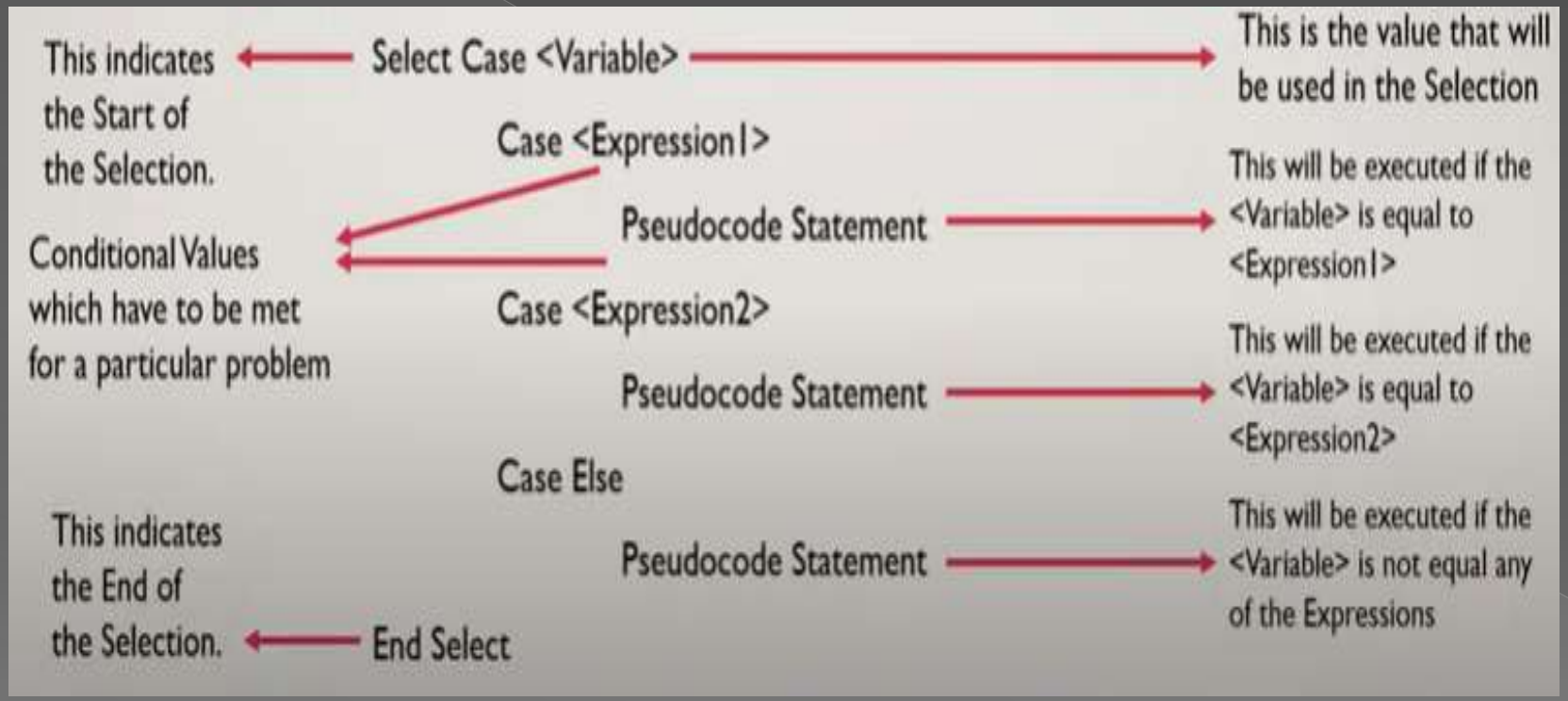
```
 START
1.Declare num
2. Read num
3. IF (num ==0 ) THEN
     Print "Zero"
   ELSE IF(num>0)
     Print "positive"
   ELSE
    Print "Negative"
   ENDIF
 STOP
```

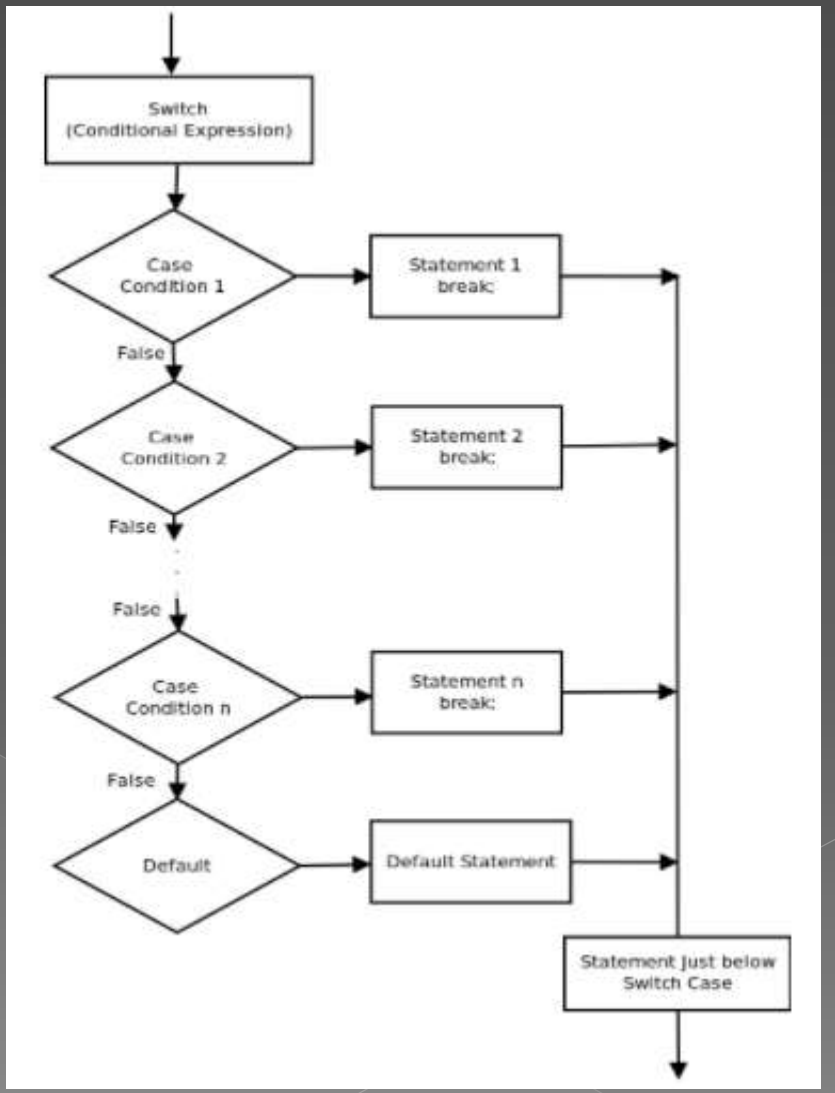


Algorithm - smallest of three numbers

```
START
1. a,b,c,small
2. Read a,b,c
3. small <- a
4. IF (b <small)THEN
     small<-b
   ENDIF
5. IF (c < small) THEN
     small<-c
   ENDIF
6. Write "Smallest is",small
STOP
```

Select case statement





Write Algorithm ,that will input a category and output the range of code numbers based on the table below. Any other category will result to an error.

Category	Code Number
A	1-300
B	301-700
C	701-1000

```
 START
1.Declare category
2.Read category
3.Select category
    Case A
        Output "The range of code number is 1-300"
    Case B
        Output "The range of code number is 301-700"
    Case C
        Output "The range of code number is 701-1000"
    Case Else
        Output "Invalid Category"
 End Select
 STOP
```

ITERATION

What if we need to tell the computer to keep doing something until some condition occurs?

For Ex - we wish to indicate that you need to keep filling the kettle with water until it is full.

We need a loop , or ITERATION.

```
WHILE (Kettle is not full)
    DO keep filling kettle;
ENDWHILE;
```

Iterative Logic Problems

Loop or Repetition

Two Types of Problems –

1. Pre-Tested
2. Post-Tested

Pre -Tested

Write Algorithm to display numbers from 1 to N.

suppose n=5

```
WHILE (<CONDITION>)
    DO <Statements>;
ENDWHILE;
```

```
START
1. i<-1,n
2. Read n
3. WHILE (i <= n) DO
      Print i
      i <- i+1
      ENDWHILE
STOP
```

	i	Output
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6 (condition false)	Come out from loop

Write Algorithm to sum of natural numbers from 1 to N.

suppose n=5

```
START
1. i<-1,n ,sum <- 0
2. Read n
3. WHILE (i <= n) DO
     sum <- sum+i
     i <- i+1
     ENDWHILE
4. Print "Sum",sum
STOP
```

	i	sum
1	1	0
2	2	1
3	3	3
4	4	6
5	5	10
6	6 (condition false)	15
	So 15 will get printed on screen	

Write Algorithms that will input 100 numbers.

```
 START
1.Declare count,num
2. For count <- 1 To 100
     Get num
   Next
 STOP
```

Post - Tested

Write Algorithms that will print 1 to 100 numbers.

```
 START
1.num <- 1
2. REPEAT
    Print num
    num <- num+1
    Until (num<=100)
 STOP
```

Example

Write an algorithm to find the average marks scored by a student. Also check whether the student has passed or failed. For a student to pass, the average marks secured should be 65.

Input->mark1,mark2,mark3

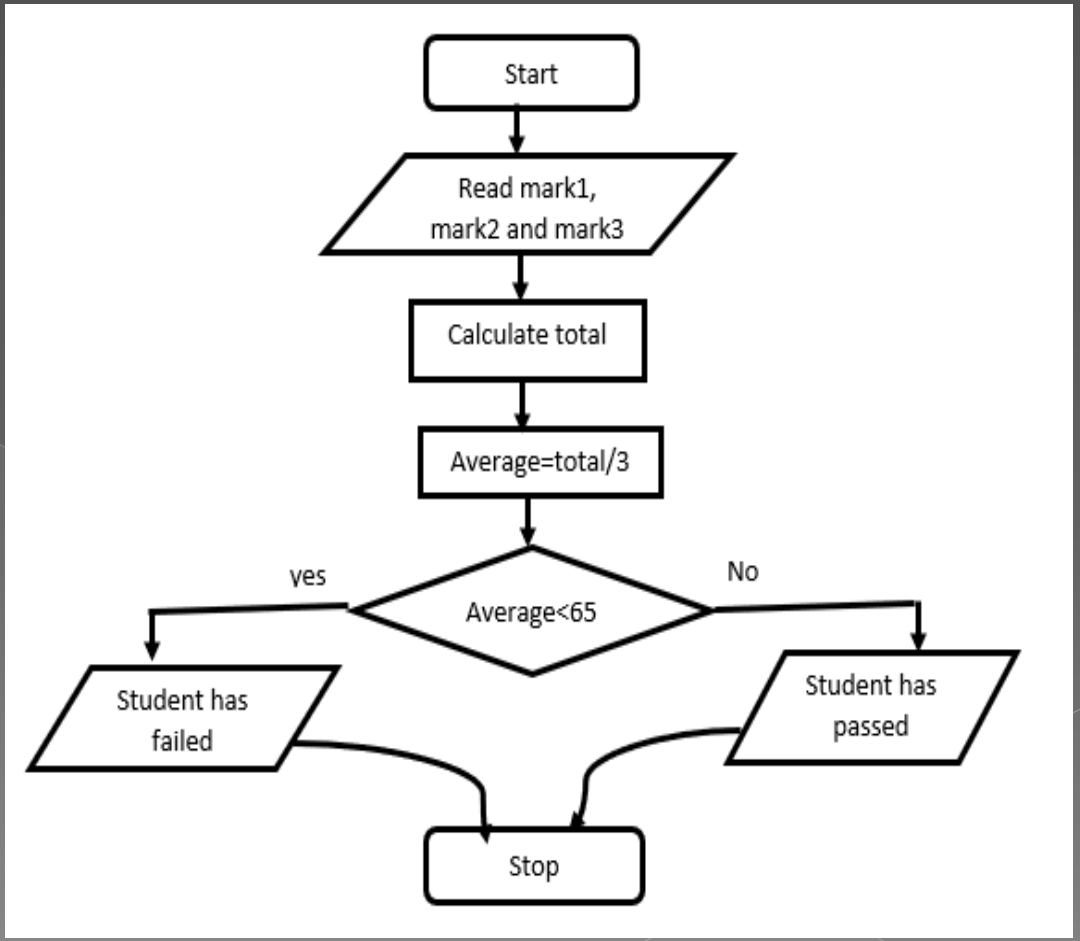
Process-> total & average

Check average is below 65->failed

Check average is above 65->passed

Output-> Print "student failed" or "student passed"

```
START
1. declare mark1,mark2,mark3,total,average
2. Read mark1,mark2,mark3
3. total = mark1+mark2+mark3
4.average=total/3
5. IF (average < 65) THEN
    Write "Student has failed"
   ELSE
    Write "Student has passed"
ENDIF
STOP
```



Example

Draw a flowchart for a program that calculates and prints the sum of the even integers from 2 to 30.

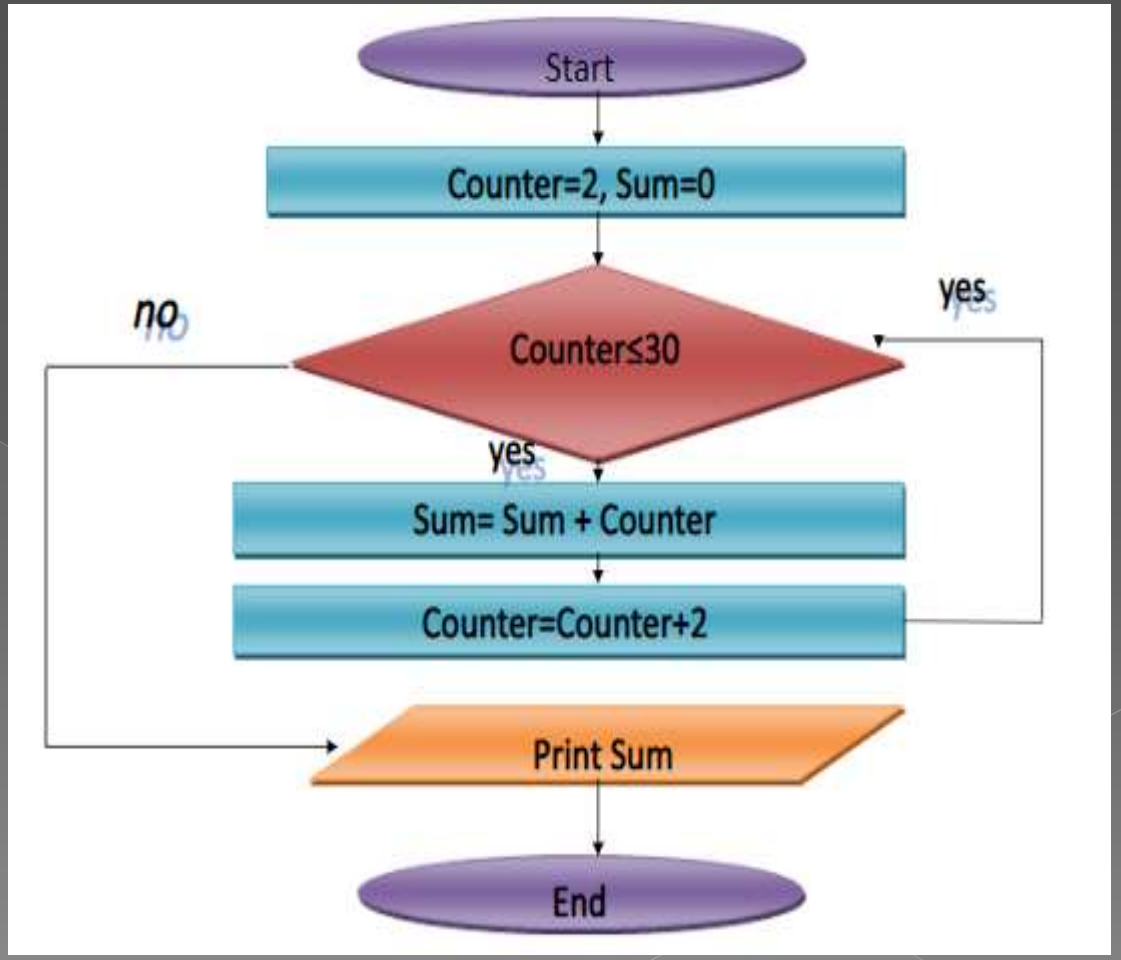
Input-> No input.

Processing->

Sum =2+4+6+8 ……+28+30

Output->sum

```
START
1. counter <- 2 ,sum <- 0
2.WHILE (counter <= 30)DO
     sum<- sum+counter
     counter<- counter+2
ENDWHILE
3. Print sum
STOP
```

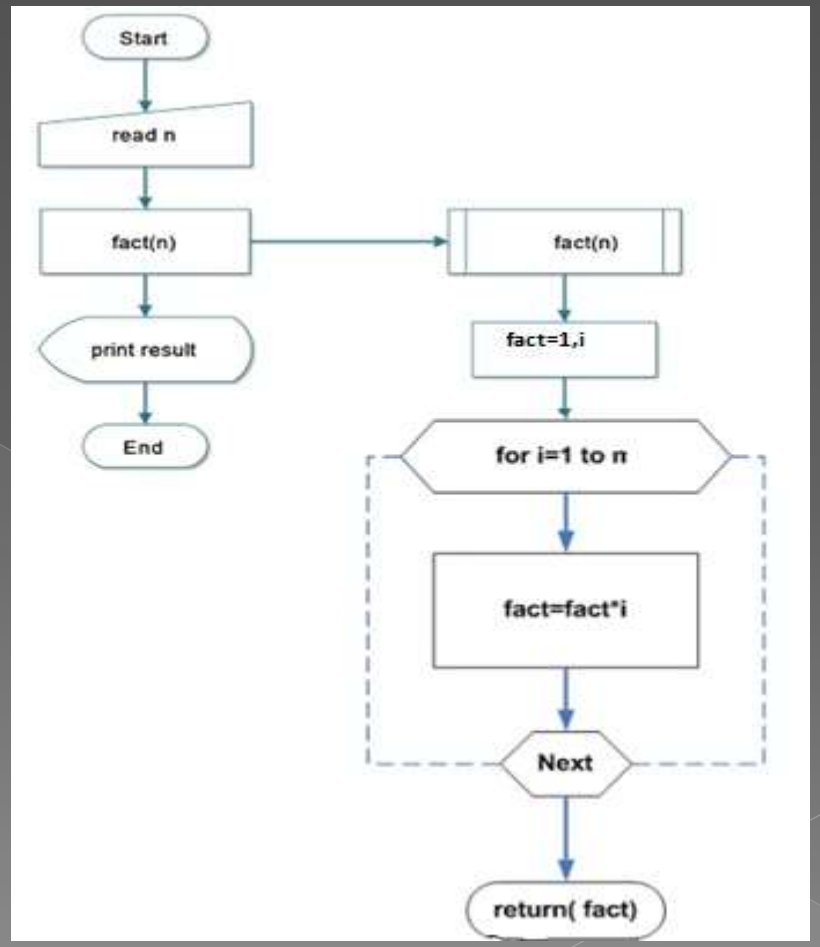


Example

Write an algorithm and design flowchart to find factorial of a number using function.

```
START
1. declare n ,result
2. Read n
3. result=fact(n)
4. print result
STOP

FUNCTION fact(n)
    1. fact<-1 ,i
    2. for i <- 1 to n
            fact <- fact*i
       Next
    3. return fact
END FUNCTION
```

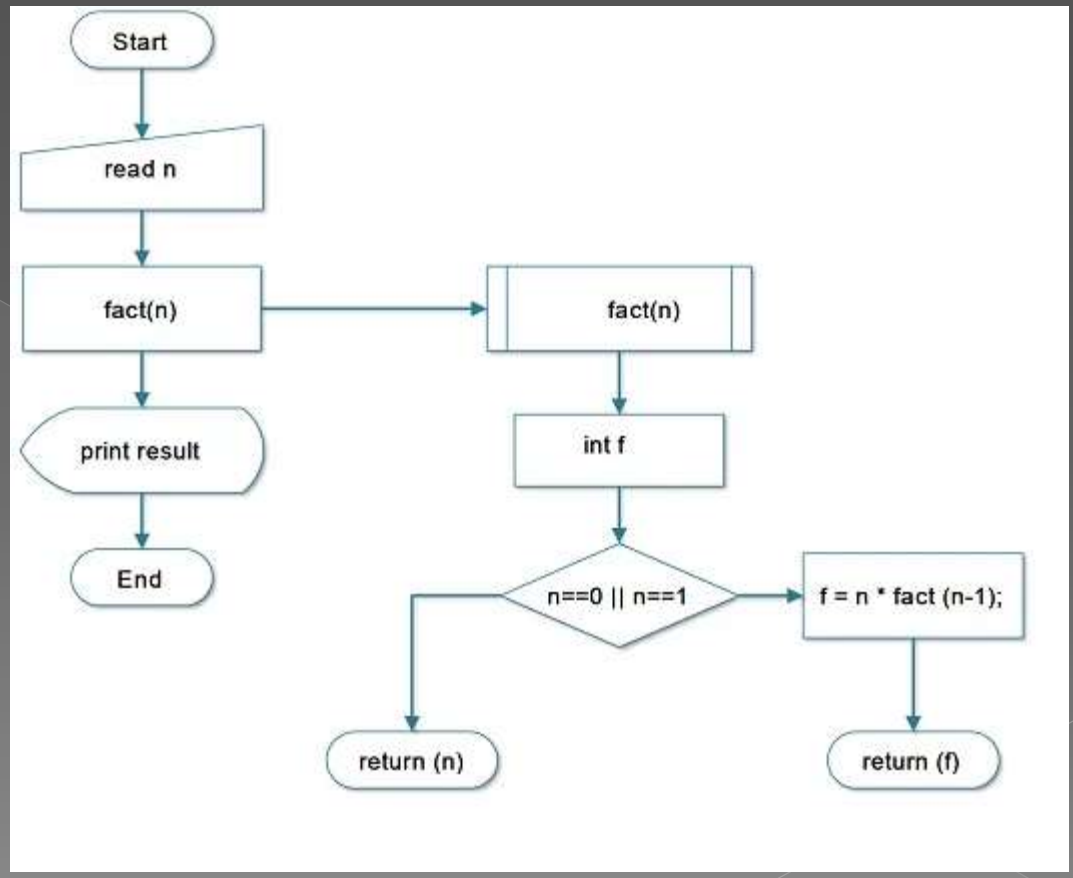


Example

Write an algorithm and design flowchart to find factorial of a number using recursion.

```
START
1. declare n ,result
2. Read n
3. result=fact(n)
4. print result
STOP

FUNCTION fact(n)
    1. f<-1
    2. IF(n==0 || n==1)  THEN
            return  n
        ELSE
            f=n*fact(n-1)
            return f
END FUNCTION
```



Essential Soft Skills Every Programmer Needs

analyze information

determine relevance or truth

be objective not emotional

devise solutions to problems

break ideas into parts

evaluate and revise ideas

LOGICAL

What is Logical Thinking

Being a good programmer isn't about know the commands and syntax of a particular programming language.

The skill in programming is being able to use logic to break down a big problem into a series of smaller steps that you know how to solve.

Logical thinking is basically the process through which one estimates the difference between what is correct and what is not.

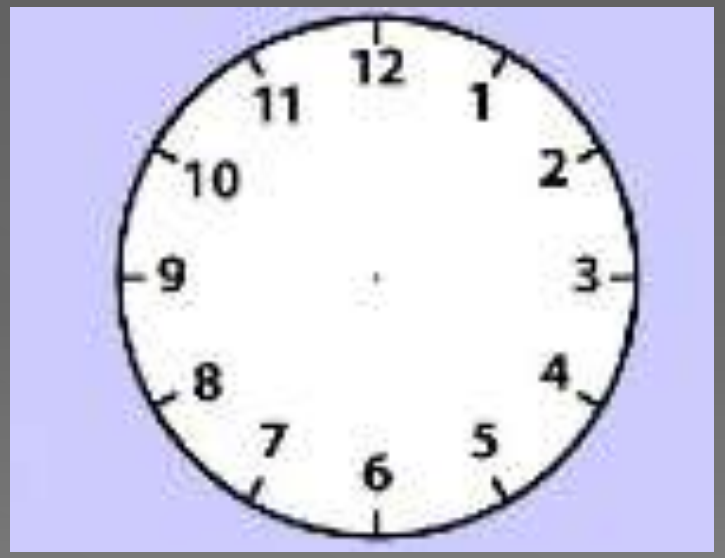
5 Ways to Develop Logical Thinking in a Programming Unit

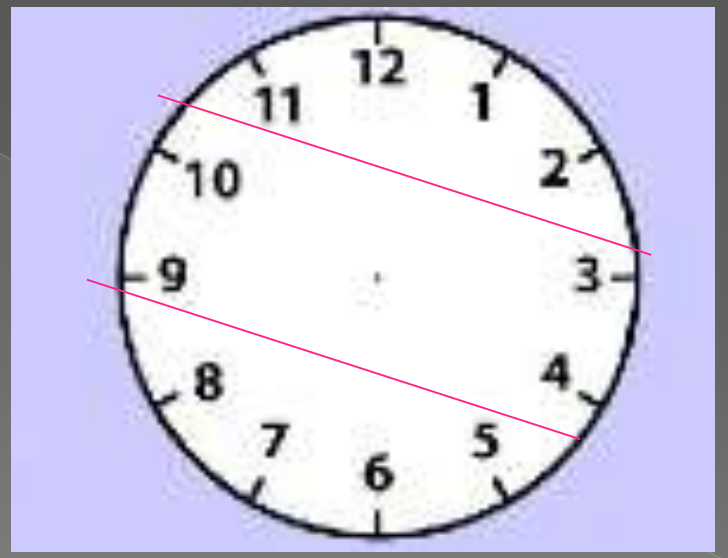
1. Analyze the Problem
2. Formulate a Plan
3. Develop Code to Solve the Problem
4. Evaluate the Solution and Revise the Code
5. Justify Decisions

Some puzzles to test your logical thinking skills

1. You're driving down the road on wild, stormy night, and you pass three people waiting at the bus stop: – an old lady who needs urgent medical treatment – your best friend who once saved your life – the partner of your dreams! • You can only fit one passenger in your car – who do you choose?

2. Can you divide a standard clock face with two straight lines so that the total of the numbers in each section is the same?





3. How can you use all of the ascending digits in order – 1 2 3 4 5 6 7 8 9

combined with only + or – operators to reach the total 100?

e. g. 1 + 23 – 4 + 5… • What stages would you go through?

Solution: 12 + 3 – 4 + 5 + 67 + 8 + 9

4. A man who lives on the tenth floor of a building takes the elevator every day to go down to the ground floor to go to work or to go shopping. When he returns in the evening, he takes the elevator to the seventh floor and walks up the stairs to the tenth floor to reach his apartment. Why does he do this? Note that if it's a rainy day, or if there are other people in the elevator, he goes to his floor directly. Also, he hates walking.

Thank you !!