

## **Introduction to Problem solving**

**Problem Solving** is the sequential process of analyzing information related to a given situation and generating appropriate response options

As for the nature of problems, they range from the mathematical to the philosophical.

In computing, we deal with algorithmic problems whose solutions are expressed as algorithms.

An algorithm is a set of well-defined steps and instructions that can be translated into a form, usually known as the code or program, that is executable by a computing device.

There are 6 steps that you should follow in order to solve an algorithmic problem:

1. Understand the Problem

2. Formulate a Model

3. Develop an Algorithm

4. Write the Program

5. Test the Program

6. Evaluate the Solution

- Problem-1: You are back from a leisure stroll in the park, and discover that you have dropped your wallet that is of great sentimental value to you. How do you retrieve your lost wallet?

- The problem statement is incomplete.

Where did you actually drop the wallet?

Did you assume that it was dropped at the park?

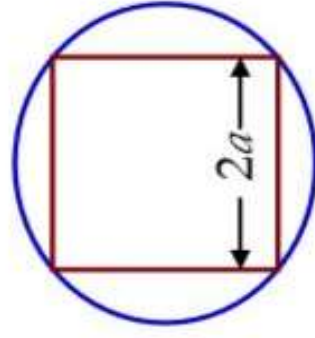
Could it have happened on the way home?

If the wallet was indeed dropped at the park, what is the park like?

These questions (and more, if you can think of) must be answered before you could reach a complete understanding of the problem.

- Assumptions should not be made without basis.
- Irrelevant information (such as the sentimental value of the wallet) should not get in the way.
- Incomplete information must be sought.
- You could only fully understand the problem after all ambiguities are removed. Only then should you set out to solve the problem.

**Problem-2:** A square, where each side has length  $2a$ , is inscribed in a circle. What is the area of the circle?



This is a simple, clearly stated problem on geometry. After you have obtained a full understanding of the problem, you need to devise a plan. You need to determine the input (the given datum – the length of each side of the square) and the output (the unknown – the area of the circle). You use suitable notation to represent the data, and you examine the relationship between the data and the unknown, which will often lead you to a solution. This may involve the calculation of some intermediate result, like the length of the diagonal of the square. You need to make use of the domain knowledge on basic geometry.

**Problem-3:** You are to ship a wolf, a sheep and a cabbage across the river. The boat can only hold the weight of two: you plus another item. You are the only one who can row the boat. The wolf must not be left with the sheep unsupervised, or the wolf will eat the sheep. Neither should the sheep be left alone with the cabbage. How do you fetch them over to the other side of the river?



# The Problem Solving Process

There are 4 phases.

**Phase 1:** Understanding the problem.

First, we need to understand the problem, and clarify any doubts about the problem statement if necessary

**Phase 2:** Devising a plan.

We need to find the connection between the data and the unknown, to make out a plan for the solution.

### **Phase 3: Carrying out the plan.**

- Carrying out your plan of the solution, check each step.
- Can you see clearly that the step is correct?
- Can you prove that it is correct?

### **Phase 4: Looking back.**

- Can you check the result?
- Can you derive the result differently?
- Can you use the result, or the method, for some other problem?

# • Properties of an Algorithm:

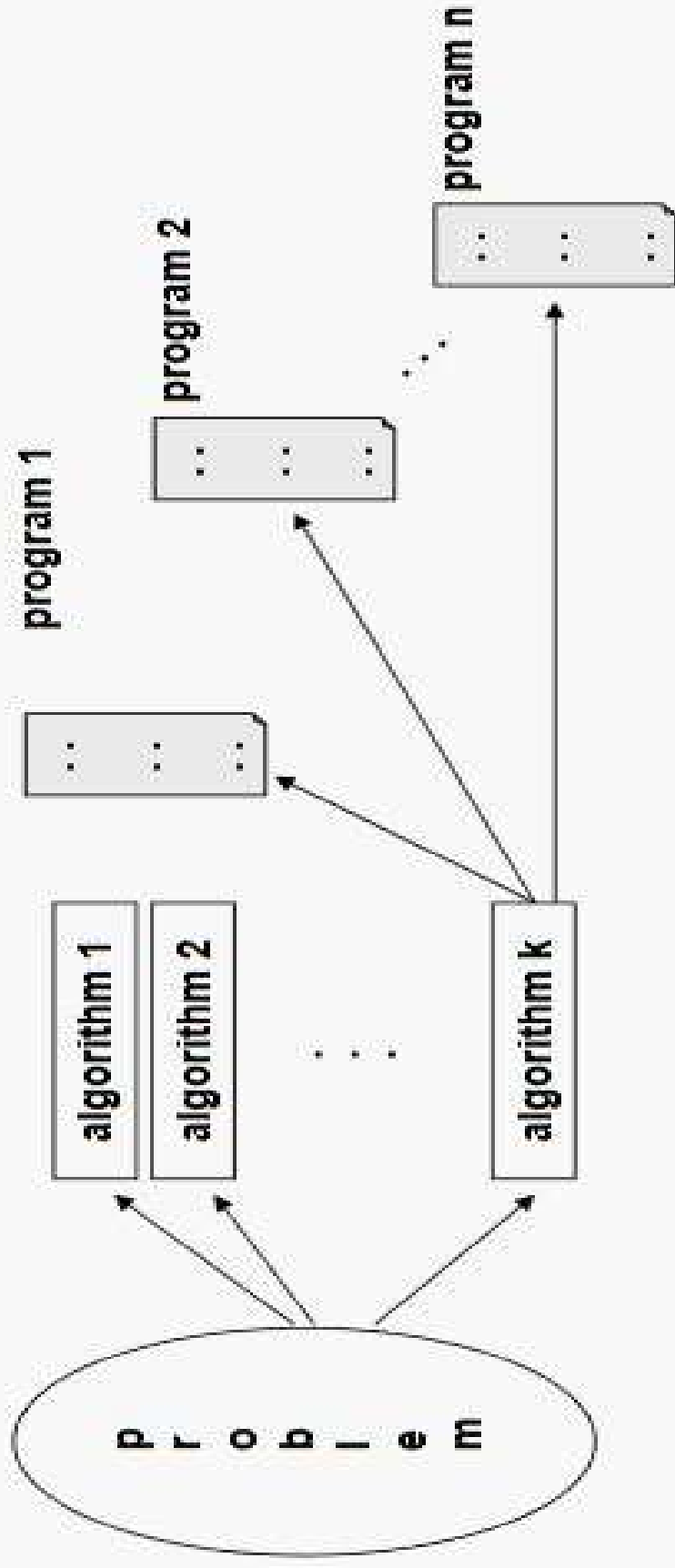
An algorithm must possess the following properties:

- finiteness:* The algorithm must always terminate after a finite number of steps.
- definiteness:* Each step must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case.
- input:* An algorithm has zero or more inputs, taken from a specified set of objects.
- output:* An algorithm has one or more outputs, which have a specified relation to the inputs.
- effectiveness:* All operations to be performed must be sufficiently basic that they can be done exactly and in finite length.

- Problems Vs Algorithms Vs Programs

For each problem or class of problems, there may be many different algorithms.

For each algorithm, there may be many different implementations (programs).



# • Common Elements of Algorithm

## *acquire data* (input)

some means of reading values from an external source; most algorithms require data values to define the specific problem (e.g., coefficients of a polynomial)

## *computation*

some means of performing arithmetic computations, comparisons, testing logical conditions, and so forth...

## *selection*

some means of choosing among two or more possible courses of action, based upon initial data, user input and/or computed results

## *iteration*

some means of repeatedly executing a collection of instructions, for a fixed number of times or until some logical condition holds

## *report results* (output)

some means of reporting computed results to the user, or requesting additional data from the user

# HOW TO WRITE ALGORITHMS

**Step 1 Define your algorithms input:** Many algorithms take in data to be processed, e.g. to calculate the area of rectangle input may be the rectangle height and rectangle width.

**Step 2 Define the variables:** Algorithm's variables allow you to use it for more than one place. We can define two variables for rectangle height and rectangle width as HEIGHT and WIDTH (or H & W). We should use meaningful variable name e.g. instead of using H & W use HEIGHT and WIDTH as variable name.

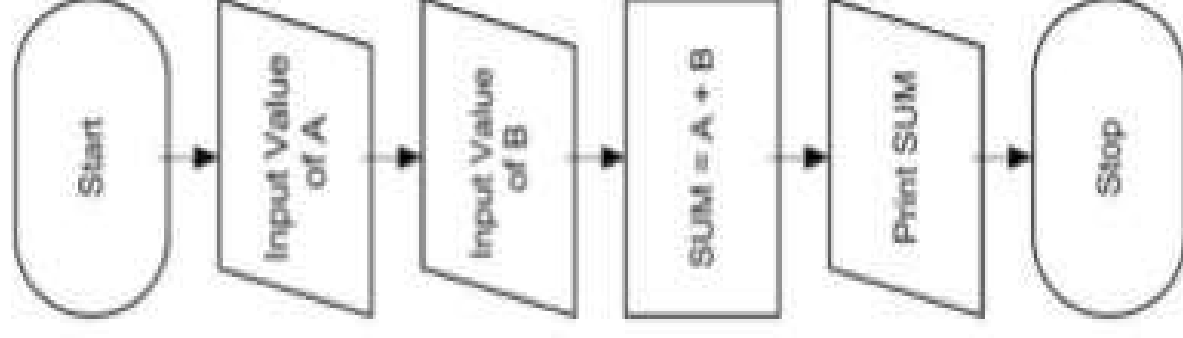
**Step 3 Outline the algorithm's operations:** Use input variable for computation purpose, e.g. to find area of rectangle multiply the HEIGHT and WIDTH variable and store the value in new variable (say) AREA. An algorithm's operations can take the form of multiple steps and even branch, depending on the value of the input variables.

**Step 4 Output the results of your algorithm's operations:** In case of area of rectangle output will be the value stored in variable AREA. if the input variables described a rectangle with a HEIGHT of 2 and a WIDTH of 3, the algorithm would output the value of 6.

## Algorithm & Flowchart to find the sum of two numbers

### Algorithm

- Step-1 Start
- Step-2 Input first numbers say A
- Step-3 Input second number say B
- Step-4  $SUM = A + B$
- Step-5 Display SUM
- Step-6 Stop



## Algorithm

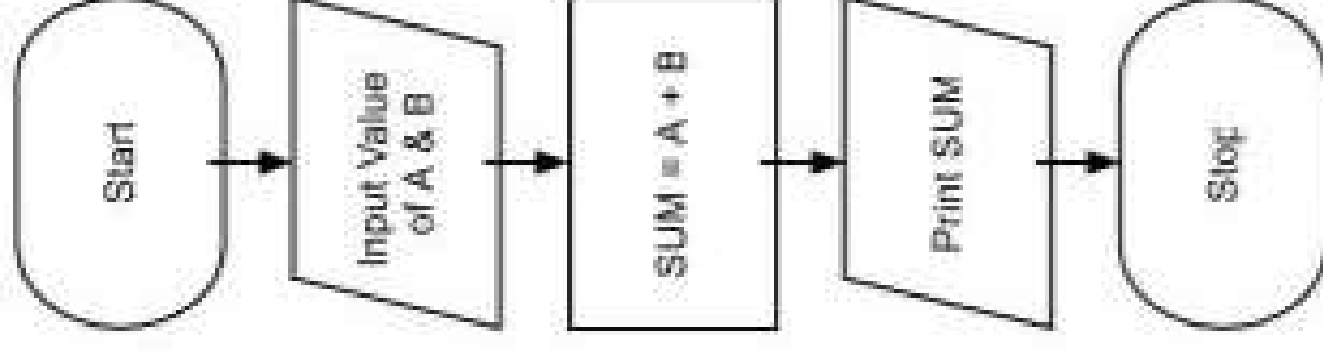
Step-1 Start

Step-2 Input two numbers say A & B

Step-3  $SUM = A + B$

Step-4 Display SUM

Step-5 Stop





# Advantages of writing algorithm

**Effective communication:** Algorithm is written using English like language, algorithm is better way of communicating the logic to the concerned people

**Effective Analysis:** with the help of algorithm, problems can be analyzed effectively, There can be number of algorithms for solving given problem

**Proper Documentation:** The selected algorithm has to be documented. The algorithm serves as a good documentation which is required to identify the logical errors

**Easy and Efficient Coding:** The algorithm act as blueprint during program development

**Program Debugging:** Debugging is nothing but identifying the logical errors in algorithm /program. Algorithms help in debugging so that we can identify the logical errors easily




**Program maintenance:** Maintaining the software becomes much easier

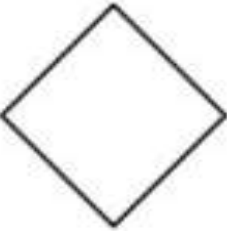
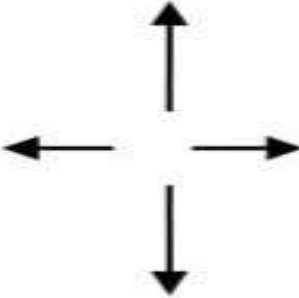

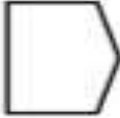




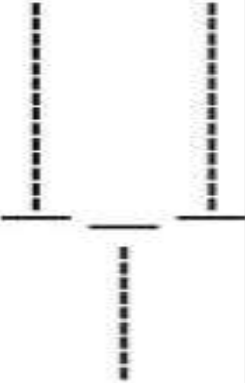


**Flowchart** is diagrammatic /Graphical representation of sequence of steps to solve a problem.

To draw a flowchart following standard symbols should use

Symbol Name	Symbol	function
Oval		Used to represent start and end of flowchart
Parallelogram		Used for input and output operation
Rectangle		Processing: Used for arithmetic operations and data-manipulations

Diamond		Decision making. Used to represent the operation in which there are two/three alternatives, true and false etc
Arrows		Flow line Used to indicate the flow of logic by connecting symbols
Circle		Page Connector
		Off Page Connector

		<p>Predefined Process /Function Used to represent a group of statements performing one processing task.</p>
		<p>Preprocessor</p>
		<p>Comments</p>

**GO TO** statement also called unconditional transfer of control statement is used to transfer control of execution to another step/statement. . e.g. the statement GOTO n will transfer control to step/statement n.

**Note: We can use keyword INPUT or READ or GET to accept input(s) /value(s) and keywords PRINT or WRITE or DISPLAY to output the result(s).**

The algorithm and flowchart include following three types of control structures.

1. **Sequence:** In the sequence structure, statements are placed one after the other and the execution takes place starting from up to down.
2. **Branching (Selection):** In branch control, there is a condition and according to a condition, a decision of either TRUE or FALSE is achieved. In the case of TRUE, one of the two branches is explored; but in the case of FALSE condition, the other alternative is taken. Generally, the 'IF-THEN' is used to represent branch control.
3. **Loop (Repetition):** The Loop or Repetition allows a statement(s) to be executed repeatedly based on certain loop condition e.g. WHILE, FOR loops.



The language used to write algorithm is simple and similar to day-to-day life language. The variable names are used to store the values. The value store in variable can change in the solution steps. In addition some special symbols are used as below

**Assignment Symbol** (  $\leftarrow$  or  $=$  ) is used to assign value to the variable.

e.g. to assign value 5 to the variable HEIGHT, statement is

HEIGHT  $\leftarrow$  5

or

HEIGHT = 5

The statement  $R = R + 1$  means that add 1 to the value stored in variable R and then assign/store the new value in variable R, in other words increase the value of variable R by 1

**Mathematical Operators:**

Operator	Meaning	Example
+	Addition	$A + B$
-	Subtraction	$A - B$
*	Multiplication	$A * B$
/	Division	$A / B$
^	Power	$A^3$ for $A^3$
%	Reminder	$A \% B$

**Relational Operators**

Operator	Meaning	Example
<	Less than	$A < B$
<=	Less than or equal to	$A <= B$
= or ==	Equal to	$A = B$
# or !=	Not equal to	$A \# B$ or $A != B$
>	Greater than	$A > B$
>=	Greater tha or equal to	$A >= B$

## Logical Operators

Operator	Example	Meaning
AND	A < B AND B < C	Result is True if both A<B and B<C are true else false
OR	A< B OR B < C	Result is True if either A<B or B<C are true else false
NOT	NOT (A >B)	Result is True if A>B is false else true

## Selection control Statements

Selection Control	Example	Meaning
IF ( Condition ) Then ... ENDIF	IF ( X > 10 ) THEN Y=Y+5 ENDIF	If condition X>10 is True execute the statement between THEN and ENDIF
IF ( Condition ) Then ... ELSE ..... ENDIF	IF ( X > 10 ) THEN Y=Y+5 ELSE Y=Y+8 Z=Z+3 ENDIF	If condition X>10 is True execute the statement between THEN and ELSE otherwise execute the statements between ELSE and ENDIF

## Loop control Statements

Selection Control	Example	Meaning
WHILE (Condition) DO .. .. ENDDO	WHILE ( X < 10) DO print x x=x+1 ENDDO	Execute the loop as long as the condition is TRUE
DO .... ... UNTILL (Condition)	DO print x x=x+1 UNTILL ( X > 10)	Execute the loop as long as the condition is false

**GO TO** statement also called unconditional transfer of control statement is used to transfer control of execution to another step/statement. . e.g. the statement GOTO n will transfer control to step/statement n.

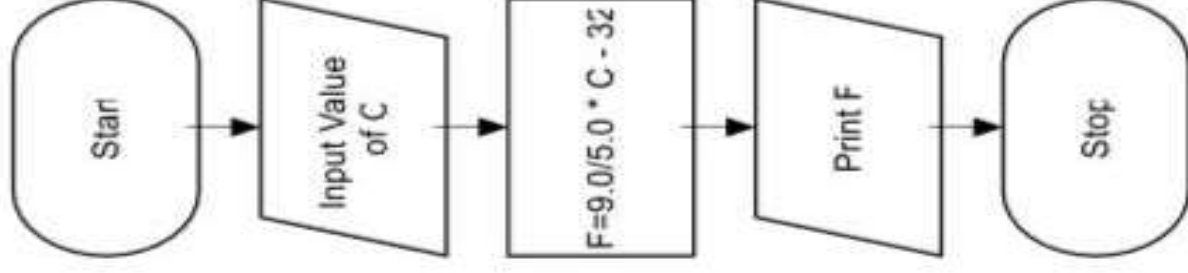
**Note:** We can use keyword **INPUT** or **READ** or **GET** to accept input(s) /value(s) and keywords **PRINT** or **WRITE** or **DISPLAY** to output the result(s).

## Algorithm & Flowchart to convert temperature from Celsius to Fahrenheit

C : temperature in Celsius  
F : temperature Fahrenheit

### Algorithm

- Step-1 Start
- Step-2 Input temperature in Celsius say C
- Step-3  $F = (9.0/5.0 \times C) + 32$
- Step-4 Display Temperature in Fahrenheit F
- Step-5 Stop



# Algorithm & Flowchart to find Area and Perimeter of Square

L : Side Length of Square

AREA : Area of Square

PERIMETER : Perimeter of Square

## Algorithm

Step-1 Start

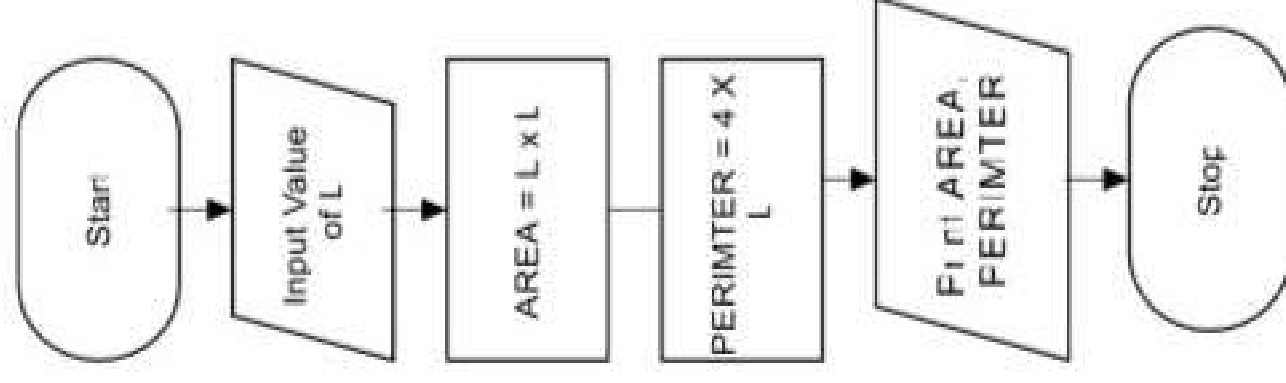
Step-2 Input Side Length of Square say L

Step-3 Area =  $L \times L$

Step-4 PERIMETER =  $4 \times L$

Step-5 Display AREA, PERIMETER

Step-6 Stop

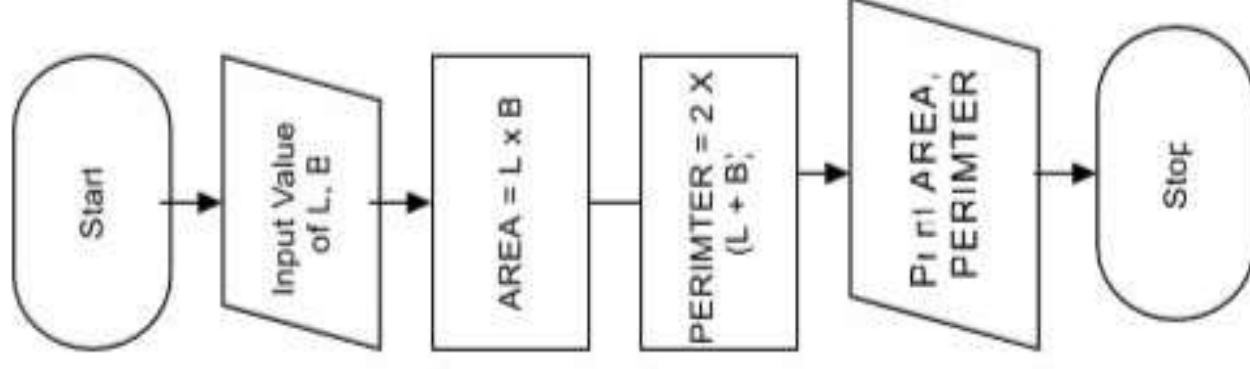


# Algorithm & Flowchart to find Area and Perimeter of Rectangle

L : Length of Rectangle  
B : Breadth of Rectangle  
AREA : Area of Rectangle  
PERIMETER : Perimeter of Rectangle

## Algorithm

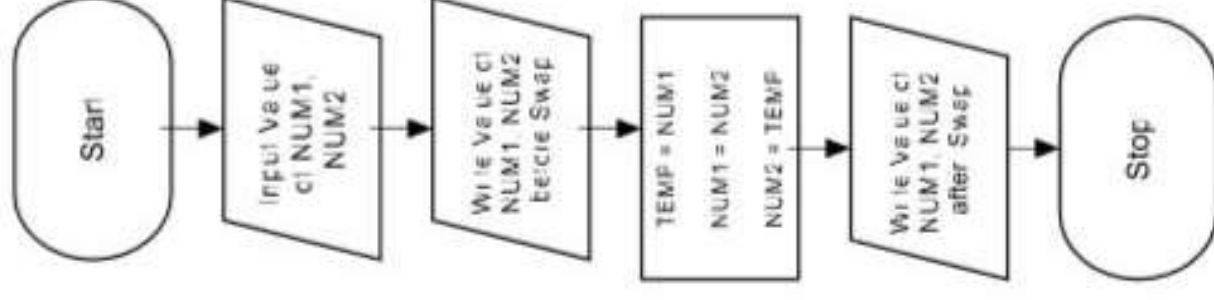
- Step-1 Start
- Step-2 Input Side Length & Breadth say L, B
- Step-3  $\text{Area} = L \times B$
- Step-4  $\text{PERIMETER} = 2 \times (L + B)$
- Step-5 Display AREA, PERIMETER
- Step-6 Stop



# Algorithm & Flowchart to Swap Two Numbers using Temporary Variable

## Algorithm

- Step-1 Start
- Step-2 Input Two Numbers Say NUM1, NUM2
- Step-3 Display Before Swap Values NUM1, NUM2
- Step-4 TEMP = NUM1
- Step-5 NUM1 = NUM2
- Step-6 NUM2 = TEMP
- Step-7 Display After Swap Values NUM1, NUM2
- Step-8 Stop





# Algorithm & Flowchart to find the smallest of two numbers

## Algorithm

Step-1 Start

Step-2 Input two numbers say

NUM1,NUM2

Step-3 IF NUM1 < NUM2 THEN

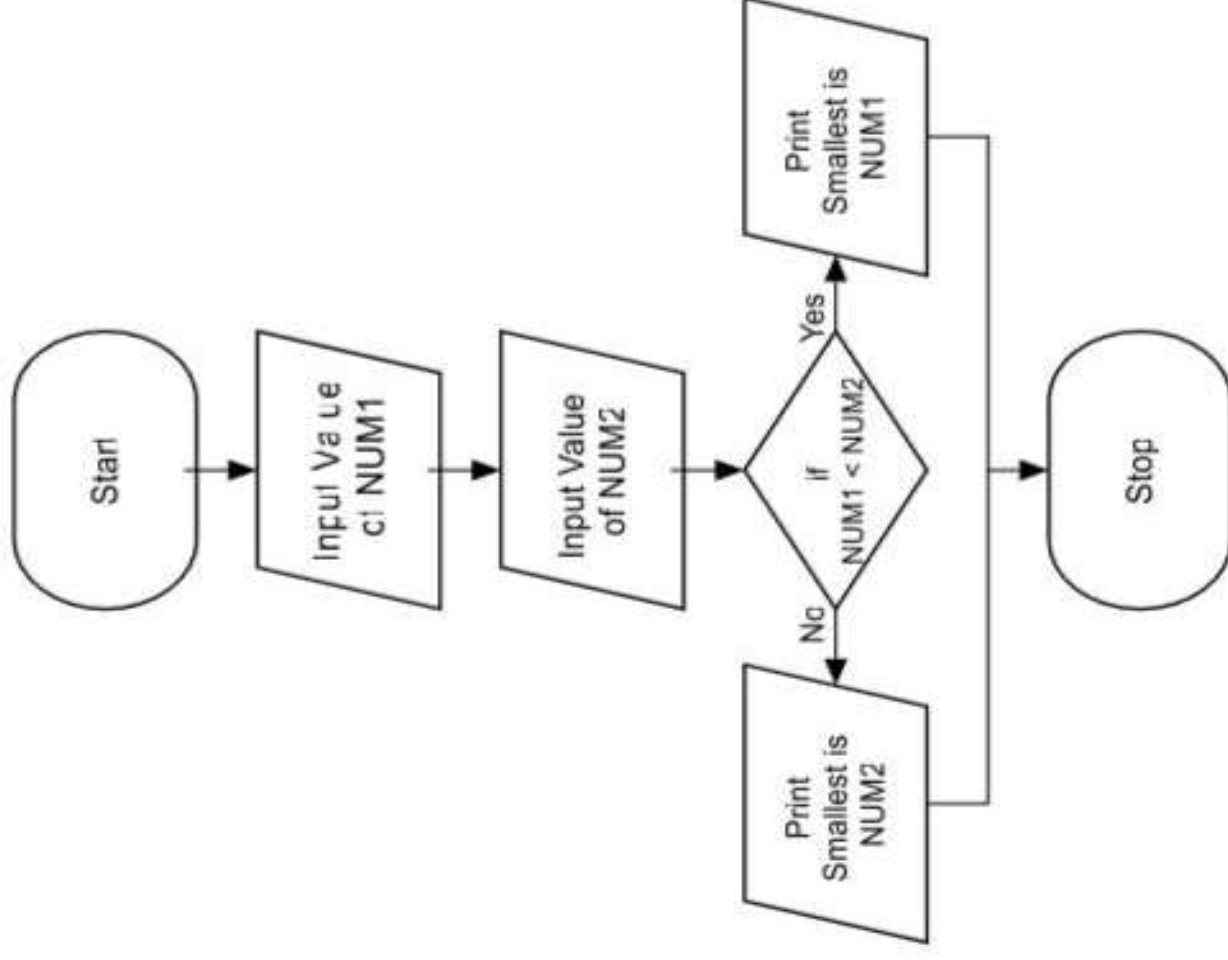
print smallest is NUM1

ELSE

print smallest is NUM2

ENDIF

Step-4 Stop



## Algorithm & Flowchart to find the largest of two numbers

### Algorithm

Step-1 Start

Step-2 Input two numbers say

NUM1, NUM2

Step-3 IF NUM1 > NUM2 THEN

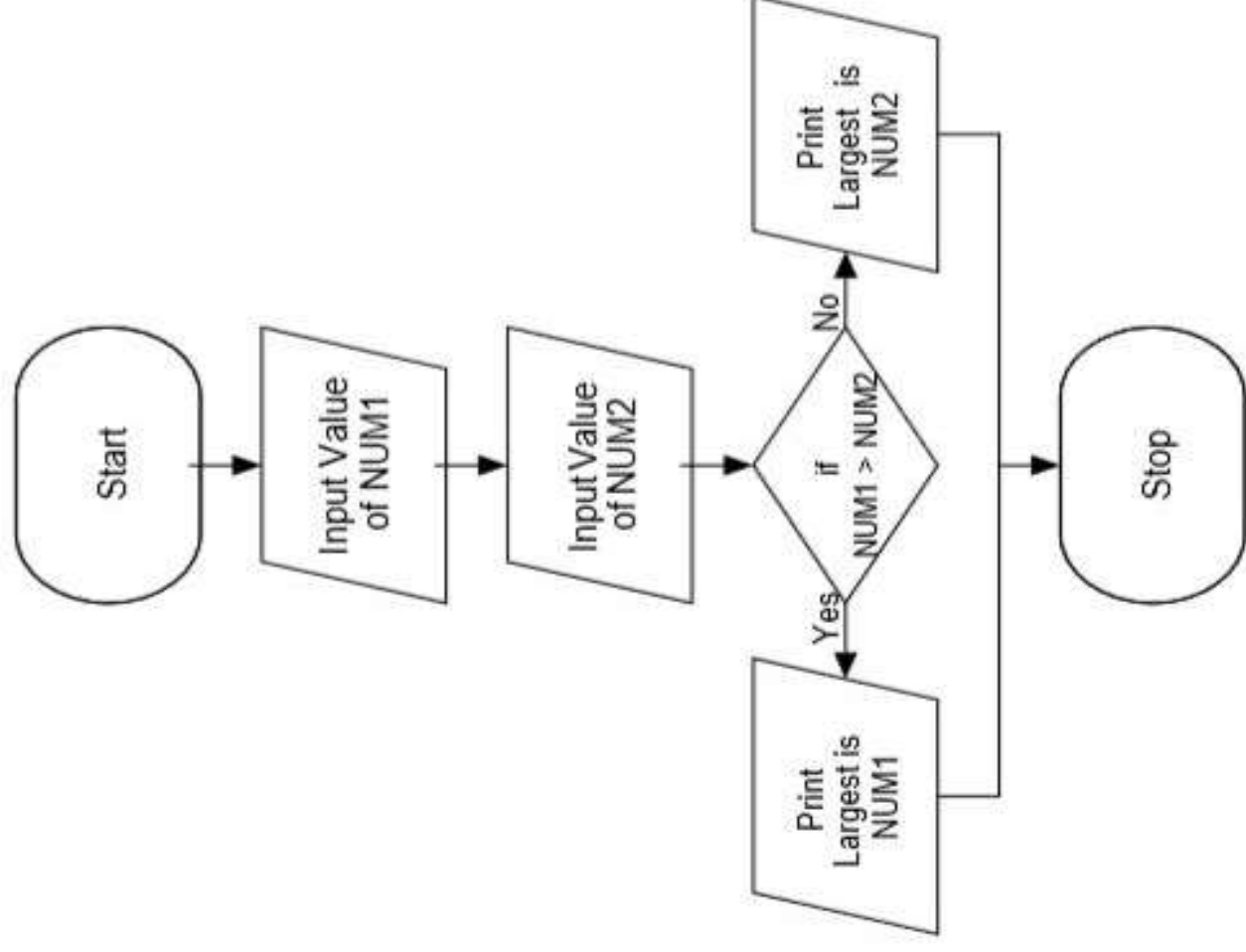
print largest is NUM1

ELSE

print largest is NUM2

ENDIF

Step-4 Stop



## Algorithm & Flowchart to find the largest of three numbers

### Algorithm

Step-1 Start

Step-2 Read three numbers say num1,num2, num3

Step-3 if num1>num2 then go to step-5

Step-4 IF num2>num3 THEN

print num2 is largest

ELSE

print num3 is largest

ENDIF

GO TO Step-6

Step-5 IF num1>num3 THEN

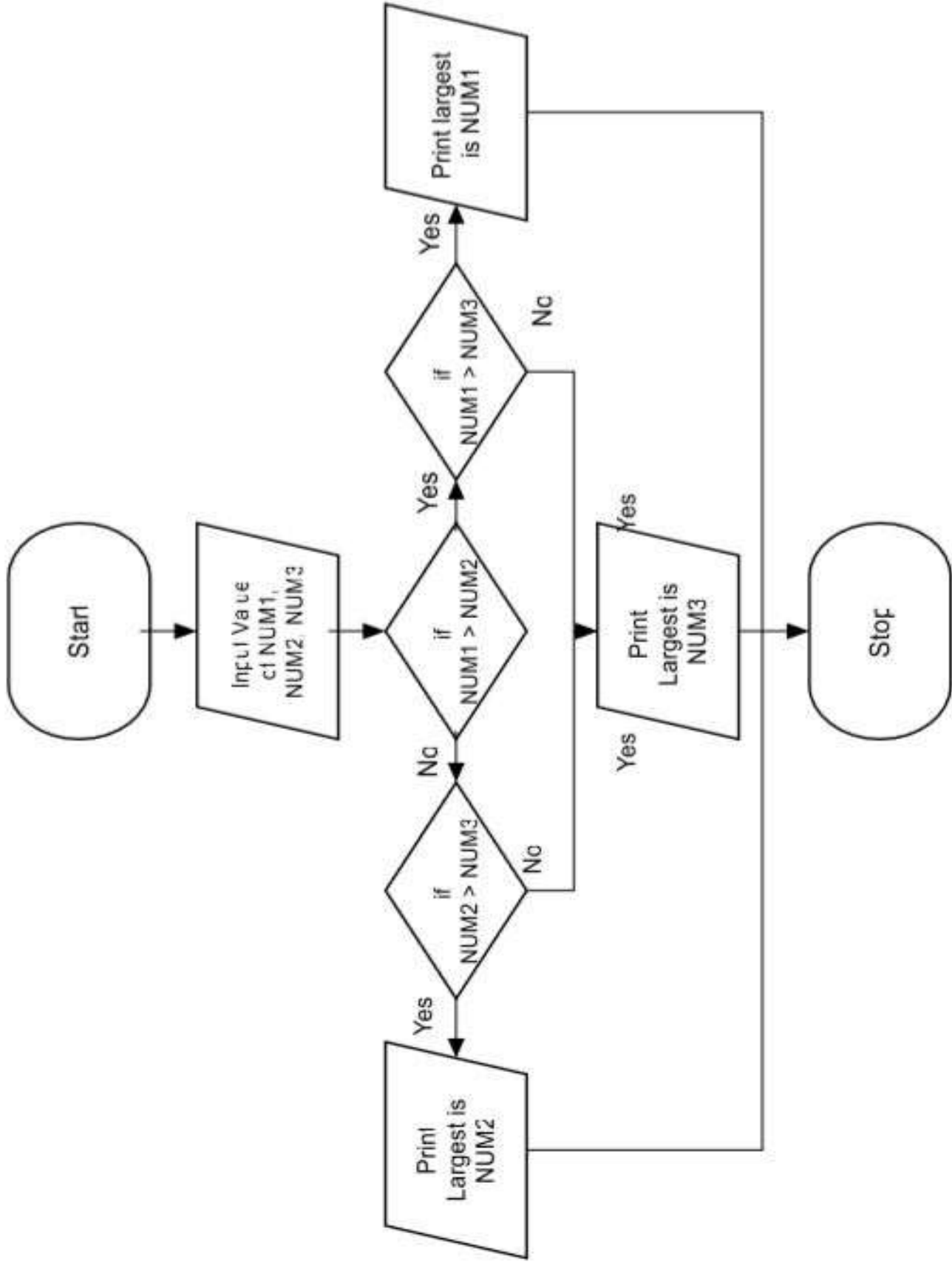
print num1 is largest

ELSE

print num3 is largest

ENDIF

Step-6 Stop



## Algorithm & Flowchart to find Even number between 1 to 50

### Algorithm

Step-1 Start

Step-2  $I = 1$

Step-3 IF ( $I > 50$ ) THEN  
GO TO Step-7

ENDIF

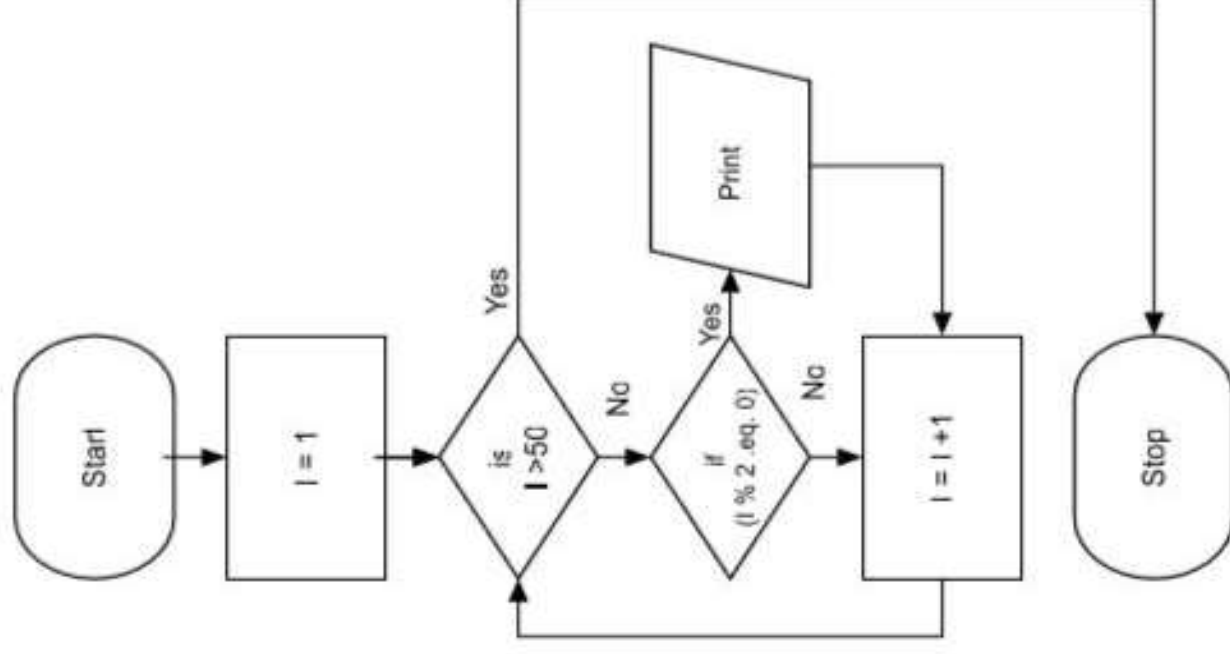
Step-4 IF ( ( $I \% 2$ ) = 0) THEN  
Display I

ENDIF

Step-5  $I = I + 1$

Step-6 GO TO Step--3

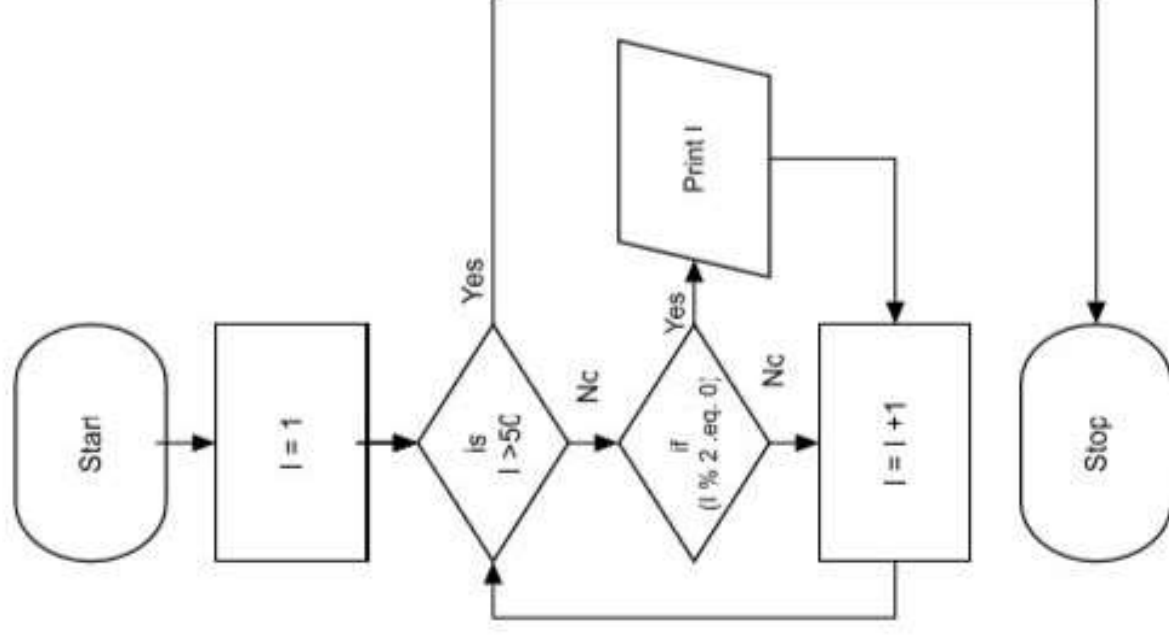
Step-7 Stop



## Algorithm & Flowchart to find Odd numbers between 1 to n where n is a positive Integer

### Algorithm

- Step-1 Start
- Step-2 Input Value of N
- Step-3  $I = 1$
- Step-4 IF ( $I > N$ ) THEN  
GO TO Step-8  
ENDIF
- Step-5 IF ( ( $I \% 2$ ) = 1 ) THEN  
Display I  
ENDIF
- Step-6  $I = I + 1$
- Step-7 GO TO Step-4
- Step-8 Stop



## Algorithm & Flowchart to find sum of series $1+2+3+\dots+N$

### Algorithm

- Step-1 Start
- Step-2 Input Value of N
- Step-3  $I = 1$ ,  $SUM = 0$
- Step-4 IF ( $I > N$ ) THEN  
GO TO Step-8  
ENDIF
- Step-5  $SUM = SUM + I$
- Step-6  $I = I + 1$
- Step-7 Go to step-4
- Step-8 Display value of SUM
- Step-9 Stop

