Unit 3: Algorithm and Flowcharts

Subject: Introduction to IT System

Algorithm

An algorithm is a finite set of well-defined, step-by-step instructions used to solve a problem or perform a specific task.

In simple terms:

- It is a procedure or method for solving a problem logically.

Example of an Algorithm (Simple)

Problem: Add two numbers

Algorithm:

- Start
- Input two numbers A and B
- Compute Sum = A + B
- Display Sum
- Stop

Characteristics of an Algorithm

A good algorithm must satisfy the following characteristics:

1. Input

- An algorithm can have zero or more inputs
- Inputs are the values required to solve the problem
  Example: Numbers, user data, etc.

2. Output

- Must produce at least one output
- Output is the result of processing
  Example: Sum, result, decision

3. Definiteness (Clarity)

- Each step must be clear and unambiguous
- Instructions should not confuse the user
  Example:

⊗ "Do some calculation" (Wrong)
✓ "Add A and B" (Correct)

4. Finiteness

- Algorithm must terminate after a finite number of steps
- It should not run forever

5. Effectiveness

- Each step must be basic and executable
- Should be possible to perform in real time

6. Generality

- Algorithm should work for all valid inputs, not just one case
  Example: Adding any two numbers, not just specific values

Advantages of Algorithms

- Easy to understand and write
- Helps in problem-solving
- Language independent (can be implemented in any programming language)
- Easy to debug and modify

Disadvantages of Algorithms

- Time-consuming for complex problems
- Not visual (harder to understand flow compared to flowcharts)

Flowchart

A flowchart is a graphical (diagrammatic) representation of an algorithm using symbols and arrows.

It shows the flow of control step-by-step.

Common Flowchart Symbols

FLOWCHART SYMBOLS – ALL IN ONE

No.	Symbol Name	Shape	Use / Function	Example
1.	Terminator (Start/End)	Start / End	• Represents the beginning or end of a process. • Indicates Start or Stop.	Example: Start Stop
2.	Input / Output	Input / Output	• Used for taking input or displaying output. • Read Data / Print Data.	Example: Input A Print Result
3.	Process	Process	• Represents calculations or operations. • Any processing step.	Example: Sum = A + B
4.	Decision	Decision	• Used for decision making. • Checks a condition. • Has Yes/No (True/False) paths.	Example: Yes A > B ? No
5.	Flow Line	→	• Shows the direction of flow. • Connects one symbol to another.	Example: A → B
6.	Connector	○	• Connects different parts of a flowchart. • Useful for large flowcharts.	Example: 1 - - - - - 1
7.	Predefined Process (Optional)	Predefined Process	• Represents a subroutine or predefined function. • A module called in the program.	Example: Function Call

Example of Flowchart (Addition of Two Numbers)

Steps Explained:

- Start
- Input A and B
- Add A + B
- Display result
- End

Advantages of Flowcharts

- Easy to understand (visual representation)
- Helps in debugging
- Improves communication between developers
- Useful in program planning

Disadvantages of Flowcharts

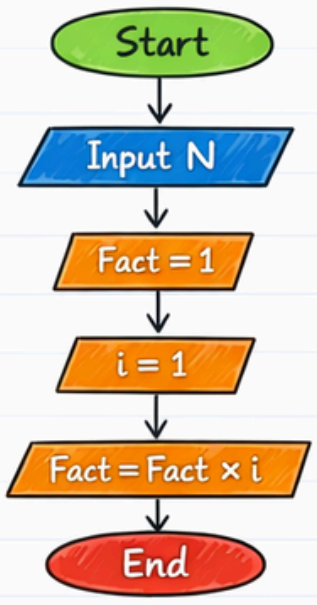
- Time-consuming to draw
- Difficult to modify for large programs
- Requires proper symbols knowledge

Difference Between Algorithm and Flowchart

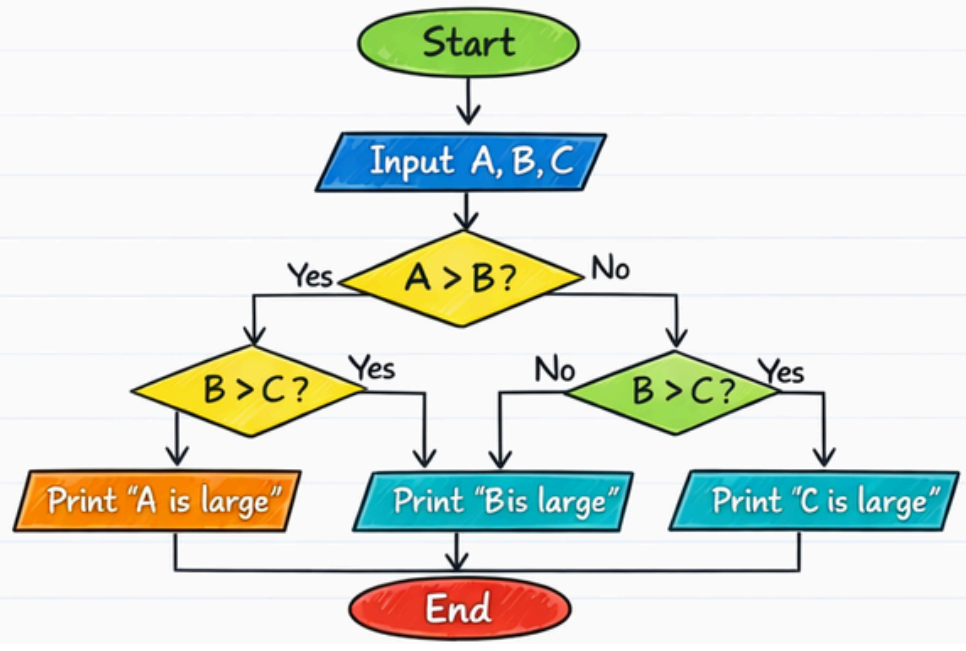
Algorithm	**Flowchart**
Written in steps	Drawn using symbols
Text-based	Visual/diagram-based
Easy to write	Easy to understand
Takes less time	Takes more time to draw

Some Examples

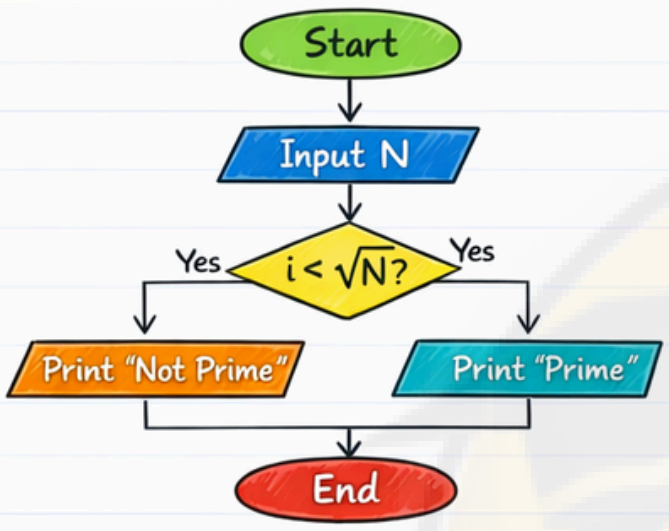
Find Factorial of a Number



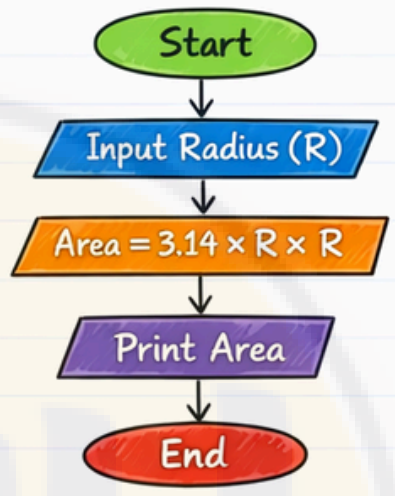
Find the Largest of Three Numbers



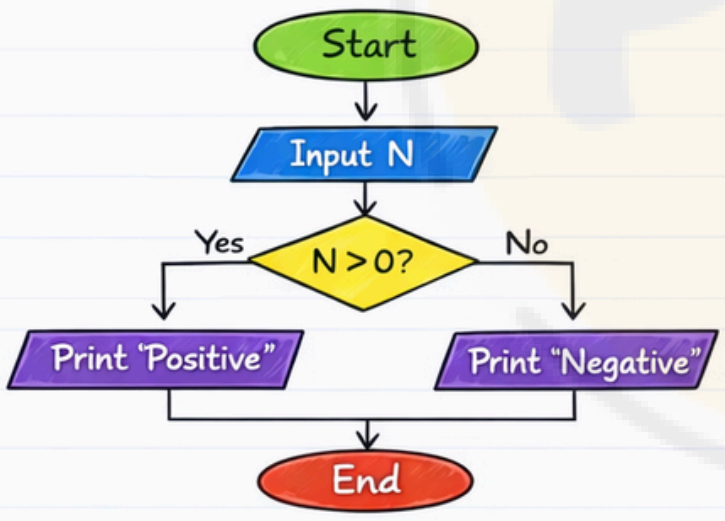
Check if Number is Prime



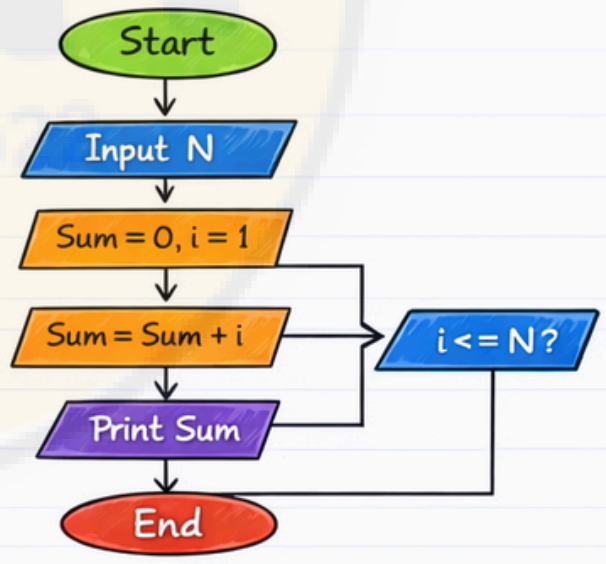
Calculate Area of a Circle



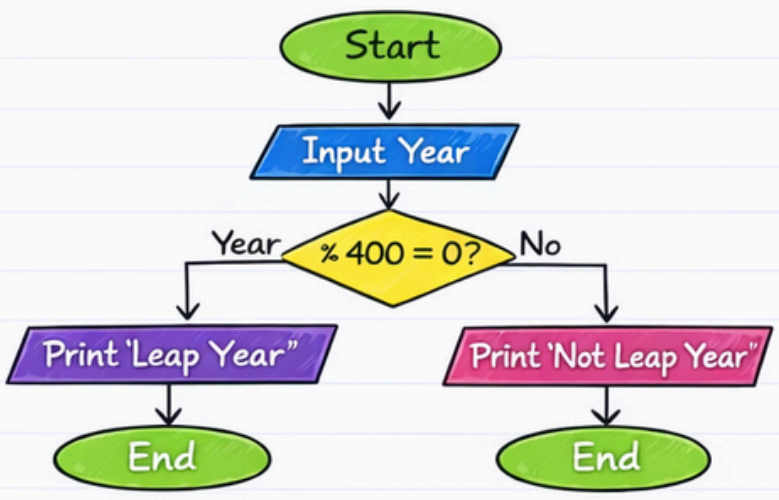
Check Positive, Negative or Zero



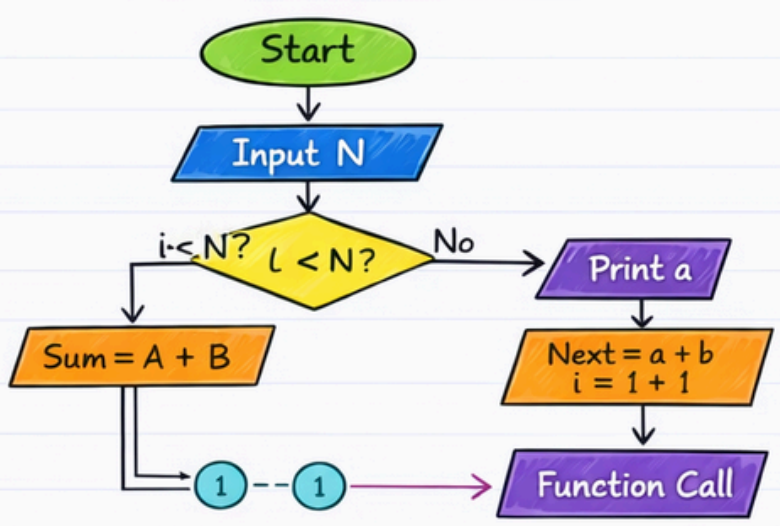
Calculate Sum of 1 to N Numbers



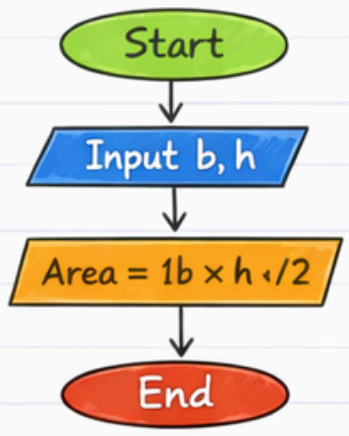
Check Leap Year



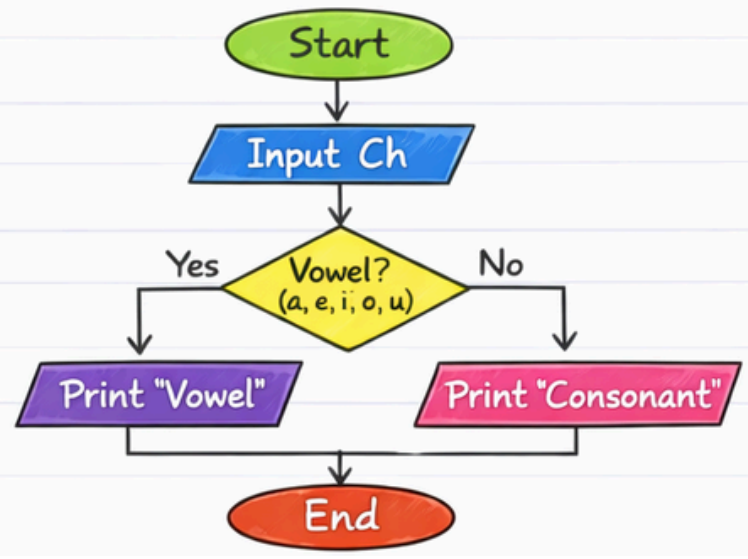
Generate Fibonacci Series



Find Area of Triangle



Check Vowel or Consonant



Determine Grade

Calculate Power of a Number