INTRODUCTION TO ALGORITHMS

Learning Objectives

By the end of this lesson, students should be able to:

1. Define what an algorithm is and identify its key properties.
2. Explain the difference between an algorithm, a program, and pseudocode.
3. Understand what makes an algorithm correct and efficient.
4. Represent algorithms using pseudocode and flowcharts.

🧠 1. What Is an Algorithm?

Definition:

An **algorithm** is a finite, well-defined sequence of steps or instructions to solve a specific problem or perform a computation. e.g Find the sum of two numbers

Example: Finding the largest number in a list, sorting names alphabetically, or calculating factorial(n).

Characteristics of an Algorithm:

1. **Input:** Zero or more quantities are externally supplied.
2. **Output:** At least one result is produced.
3. **Definiteness:** Each instruction must be clear and unambiguous.
4. **Finiteness:** The algorithm must terminate after a finite number of steps.
5. **Effectiveness:** Each operation must be basic enough to be performed exactly in a finite time.

Algorithm vs Program

Algorithm	Program
Logical sequence of steps.	Implementation of an algorithm in a programming language.
Language-independent.	Written in a specific programming language (Python, C++, etc.).
Conceptual and abstract.	Concrete and executable.

Example: A recipe (algorithm) vs. a meal prepared using that recipe (program)

🧩 3. Representing Algorithms

1. **Natural language** – easy to understand, but ambiguous.
2. **Flowcharts** – visual diagrams showing control flow.
3. **Pseudocode** – structured description close to programming syntax

PSEUDOCODE

1. What is Pseudocode?

Pseudocode is a **simple, informal way of writing algorithms** using plain English mixed with programming-like keywords.
It describes what a program does **without following any specific programming language syntax**.

👉 In simple terms,

Pseudocode is “**fake code**” — it looks like a program but is meant for humans to understand, not computers.

2. Importance of Pseudocode

Pseudocode helps programmers to:

- **Plan** the logic of a program before coding.
- **Understand** and **communicate** algorithms clearly.
- **Translate** ideas easily into real programming languages (like Python, C++, etc.).
- **Avoid syntax errors** while focusing on logic first.

3. Characteristics of Good Pseudocode

A good pseudocode should be:

- **Clear and readable**
- **Language-independent**
- **Simple and structured**
- **Step-by-step and logical**
- **Easy to translate into actual code**

4. Common Keywords Used in Pseudocode

Keyword	Meaning
START / END	Marks the beginning and end of the program
INPUT / READ	To receive data from the user
OUTPUT / PRINT / DISPLAY	To show data or results
SET / ASSIGN	To store a value in a variable (e.g., SET Sum = A + B)
IF … THEN … ELSE	For decision making
WHILE … DO	Loop that repeats while a condition is true
FOR … TO …	Loop that repeats for a fixed number of times
REPEAT … UNTIL	Loop that continues until a condition is true
CALL / FUNCTION / PROCEDURE	For calling subroutines
COMMENT	Explanation or note (often written as // or in brackets)

5. Rules for Writing Pseudocode

1. Write **one statement per line**.
2. Use **indentation** to show hierarchy or blocks.
3. Use **simple English** words.
4. Show logical flow from top to bottom.
5. Keep statements **concise and meaningful**.
6. Use **capital letters** for keywords for clarity.

Example 1 – Add Two Numbers

Algorithm:

1. Start
2. Input A, B
3. Sum = A + B
4. Print Sum
5. End

Pseudocode:

```
START
    INPUT A, B
    SET Sum = A + B
    OUTPUT Sum
END
```

Example 2 – Find the Largest of Two Numbers

Algorithm:

```
1. Start
2. Input A, B
3. IF A > B THEN
       Print “A is the largest”
     ELSE
       Print “B is the largest”
4. End
```

Pseudocode:

```
START
   INPUT A, B
   IF A > B THEN
      OUTPUT "A is the largest"
   ELSE
      OUTPUT "B is the largest"
   ENDIF
END
```

Example 3 – Calculate Average of Three Numbers

Pseudocode:

```
START
    INPUT A, B, C
    SET Total = A + B + C
    SET Average = Total / 3
    OUTPUT Average
END
```

9. Advantages of Pseudocode

- Easy to understand and learn.
- Focuses on **logic**, not syntax.
- Can be converted into any programming language.
- Helps in **team communication** and documentation.

Simplifies debugging and testing.

FLOWCHARTS

1. What is a Flowchart?

A **flowchart** is a **diagrammatic representation of an algorithm**.

It uses **symbols** to show the sequence of steps or operations involved in solving a problem.

👉 In simple terms,

A flowchart shows how a program or process flows — from **start** to **finish** — using arrows and shapes.

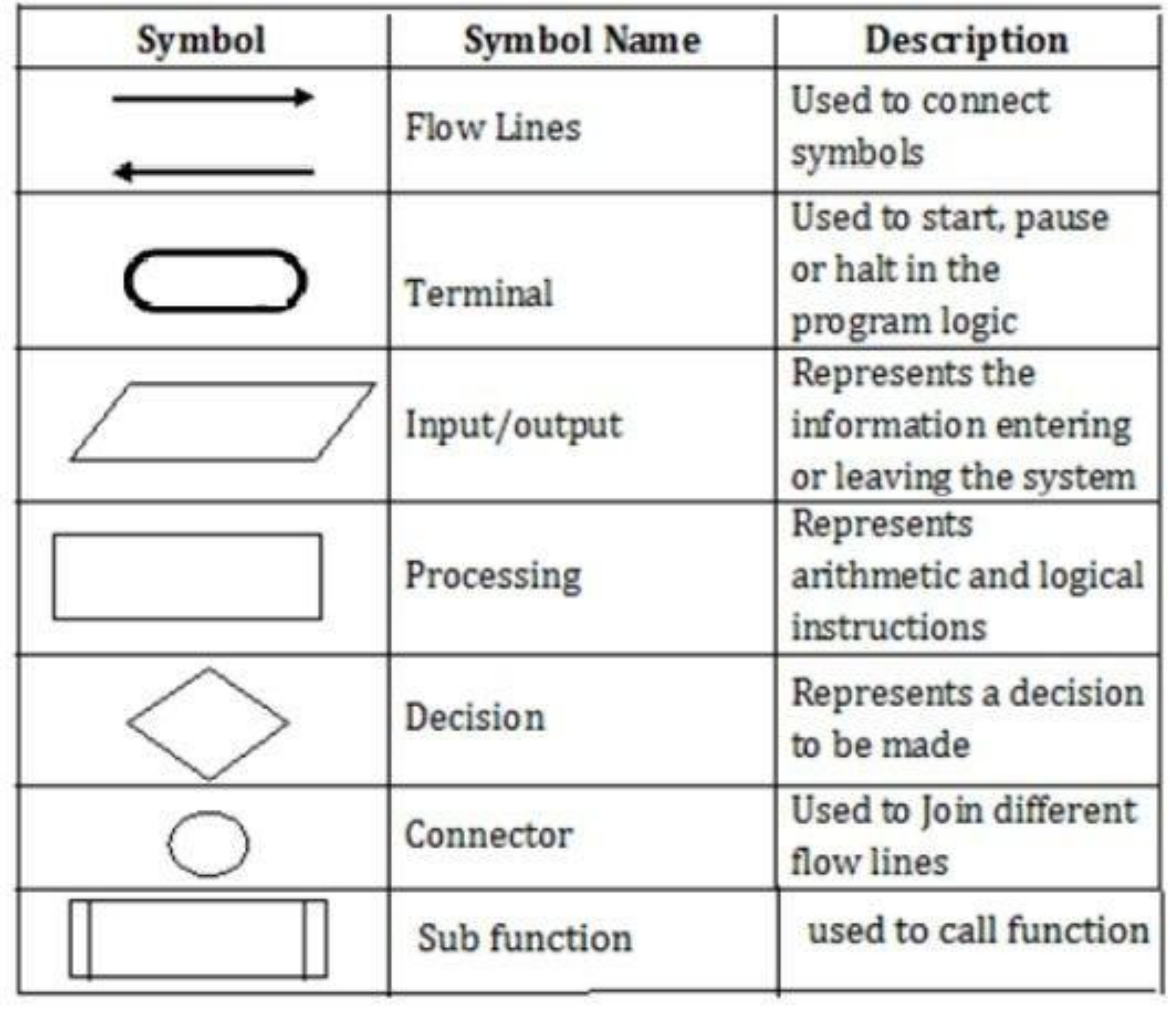
2. Importance of Flowcharts

Flowcharts help programmers and problem solvers:

- **Visualize** the logic of a program.
- **Communicate** ideas clearly and easily.
- **Debug and improve** algorithms before writing code.

Document how a process or program works.

3. Basic Flowchart Symbols



Symbol	Symbol Name	Description
→ ←	Flow Lines	Used to connect symbols
⬭	Terminal	Used to start, pause or halt in the program logic
▱	Input/output	Represents the information entering or leaving the system
▭	Processing	Represents arithmetic and logical instructions
◇	Decision	Represents a decision to be made
○	Connector	Used to Join different flow lines
	Sub function	used to call function

4. Rules for Drawing Flowcharts

1. Use **standard symbols**.
2. Flow should generally go **from top to bottom** or **left to right**.
3. Use **arrows** to indicate direction.
4. Each process or decision should be **clear and concise**.
5. Start and end points must be **clearly marked**.

Avoid crossing lines; use connectors when necessary

exercise 1 – Flowchart to Add Two Numbers

Algorithm:

1. Start
2. Input A, B
3. SUM = A + B
4. Print SUM
5. Stop

Flowchart Diagram:?????

Exercise 2 – Flowchart to Find the Largest of Two Numbers

Algorithm:

```
1. Start
2. Input A, B
3. IF A > B
        → Print A is Largest
        → Else Print B is Largest
4. Stop
```

Flowchart Diagram:???