BASIC PROGRAMMING

ASSIGNMENT

1. **What is an Algorithm?**
   An **algorithm** is a **step-by-step procedure or a set of rules** designed to perform a specific task or solve a particular problem. It takes an input, processes it through a series of well-defined steps, and produces an output.
2. **Discuss key Characteristics of an Algorithm:**

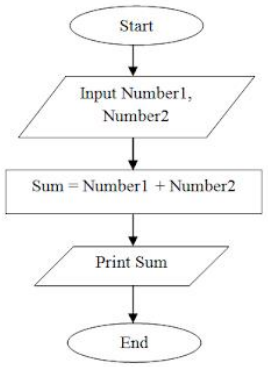
1. **Input**: Takes zero or more inputs.
2. **Output**: Produces at least one output.
3. **Definiteness**: Each step is clearly and unambiguously defined.
4. **Finiteness**: It terminates after a finite number of steps.
5. **Effectiveness**: Each step is basic enough to be carried out.

3. **Example of an Algorithm:**
   Problem: Add two numbers
   Algorithm:

1. Start
2. Read two numbers, say A and B
3. C=A+B /*Add A and B, store the result in C
4. Print C
5. Stop

4. **What is a Flowchart?**
   A **flowchart** is a **diagrammatic representation** of an algorithm or a process. It uses **symbols** to show the steps, decision points, and flow of control in a process, making it easier to understand and communicate how a system or algorithm works.

Example:



5. **What is Pseudocode?**
   Pseudocode is a **simplified, human-readable** way to describe an algorithm using plain language and basic programming-like structure. It is **not actual code** but acts as a blueprint to understand the logic of a program before writing it in a real programming language.

   Features of Pseudocode:

- Written in **plain English**
- Does **not follow strict syntax**
- Focuses on **logic, not language-specific rules**
- Easier to understand for both programmers and non-programmers

6. **Write Pseudocode: Area and Perimeter of a Rectangle**
   Problem:
   Given the **length** and **width** of a rectangle, find its **area** and **perimeter**.

   Pseudocode:

```
START
  INPUT length
  INPUT width
  area ← length × width
  perimeter ← 2 × (length + width)
  PRINT "Area = ", area
  PRINT "Perimeter = ", perimeter
END
```

7. **Machine Level Language**

Machine language is the **lowest-level programming language**, consisting of **binary code (0s and 1s)** that the computer's CPU can directly execute.

✅ **Characteristics:**

- Fastest execution
- Hardware-dependent
- Very difficult to read, write, and debug

- No abstraction

Example (binary instructions):

10110000 01100001

8. **What is Assembly Level Language**

Assembly language uses **mnemonics (short words)** to represent machine instructions. It is **one level above** machine language and must be converted to machine code by an **assembler**.

✅ **Characteristics:**

- Easier than machine code
- Still hardware-dependent
- One-to-one correspondence with machine instructions
- Requires knowledge of computer architecture

Example:

```
MOV AL, 61h   ; Move the hexadecimal value 61 into register AL
ADD AL, 1     ; Add 1 to the value in AL
```

9. **What is High-Level Language**

High-level languages are **closer to human languages** and abstract away the hardware details. They are **platform-independent** (mostly) and must be translated (using **compilers or interpreters**) into machine code.

✅ **Characteristics:**

- Easy to write, read, and debug
- Portable across systems
- Supports abstraction, functions, data structures
- Slower than low-level in some cases, but very productive

Example (in Python):

```
a = 97
a = a + 1
print(a)
```

10.Difference between machine, Assembly and High-level language.

Feature	Machine Language	Assembly Language	High-Level Language
Readability	Not human-readable	Slightly readable	Easy to read
Portability	No	No	Yes
Abstraction Level	None	Low	High
Speed	Fastest	Fast	Comparatively slower
Translation Tool	None (direct execution)	Assembler	Compiler / Interpreter