MODULE 4.1 :- C++ Basic

Q1. WAP to print “Hello World” using C++.

```
A1. #include <iostream>

using namespace std;

int main()

{

    cout << "Hello World" << endl;

    return 0;

}
```

Q2. What is OOP? List OOP concepts.

A2. Object-oriented programming aims to implement real-world entities like inheritance, polymorphism, etc in programming. The main aim of OOP is to make data safe by binding the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOP’s Concepts:

- **Class**: A blueprint or template for creating objects. It defines a type of object according to the data structure and the methods that objects of this type can perform.

- **Object**: An instance of a class. Objects are created using the class, and they can store data and perform operations defined in the class.

- **Inheritance**: A mechanism where one class (subclass or derived class) inherits the attributes and methods of another class (superclass or base class). It promotes code reusability.

- **Encapsulation**: The concept of bundling data (attributes) and methods (functions) that operate on the data into a single unit or class. It restricts direct access to some of the object's components, which can prevent accidental interference and misuse.

- **Polymorphism**: The ability of different objects to respond, each in its way, to identical messages (or methods calls). It allows for methods to be used interchangeably as long as they follow a common interface or superclass.

- **Abstraction**: The concept of hiding the complex implementation details and showing only the necessary features of an object. It helps in reducing complexity and increases efficiency.

Q3. What is the difference between OOP and POP?

A3.

Feature	Object-Oriented Programming (OOP)	Procedural-Oriented Programming (POP)
Focus	Objects and classes.	Procedures or functions.
Organization	Uses objects and classes.	Organizes code using functions.
Data Access	Data is accessed through objects.	Data is accessed globally by functions.
Inheritance	Supports inheritance.	Does not support inheritance.
Encapsulation	Promotes encapsulation.	Does not emphasize encapsulation.
Polymorphism	Allows methods to be interchangeable.	Does not support polymorphism directly.
Examples	C++, Java, Python, C#.	C, Pascal.