Q-2. What is OOP? List OOP concepts?

Ans:- OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

List OOP Concept :-

1. Class
2. Objects
3. Encapsulation
4. Abstraction
5. Polymorphism
6. Inheritance
7. Templates

Q3. What is the difference between OOP and POP?

Ans:-

Object-Oriented Programming (OOP)

1. **Concept**: OOP is based on the concept of objects, which are instances of classes. A class defines the properties (attributes) and behaviors (methods) of the objects.
2. **Data and Functions**: In OOP, data and functions are bundled together as objects. The focus is on data rather than procedure.
3. **Encapsulation**: Data is encapsulated within objects, and access is restricted through public, private, and protected access specifiers. This helps in data hiding and abstraction.
4. **Inheritance**: OOP supports inheritance, where a new class can inherit properties and behaviors from an existing class. This promotes code reusability.
5. **Polymorphism**: OOP allows polymorphism, where a single function or method can operate in different ways based on the context, often implemented through method overloading and method overriding.
6. **Abstraction**: OOP emphasizes abstracting complex systems into manageable, interacting objects, hiding the complexity from the user.

Procedural-Oriented Programming (POP)

1. **Concept**: POP is based on the concept of procedure calls. Programs are structured as a sequence of procedures or routines (also known as functions or subroutines).
2. **Data and Functions**: In POP, data and functions are separate. The focus is on procedures or functions that operate on the data.

3. **Encapsulation**: Data is not encapsulated. It is freely passed around between functions. There is no concept of access specifiers like in OOP.
4. **Inheritance**: POP does not support inheritance. Code reusability is achieved through functions and procedures, but not through class hierarchies.
5. **Polymorphism**: POP does not naturally support polymorphism. Function overloading is possible in some languages, but it's not as integral to the paradigm as in OOP.
6. **Abstraction**: POP abstracts programs into a series of procedures. It does not naturally emphasize objects and data abstraction.