1. Explain the concept of Object-Oriented Programming (OOP). How does it differ from Procedural Programming?
Ideal Answer:
Object-Oriented Programming (OOP) is a programming paradigm based on the concept of 'objects', which can contain data and code to manipulate the data. The four main principles of OOP are encapsulation, abstraction, inheritance, and polymorphism. OOP helps in organizing code into reusable components or classes.
In contrast, Procedural Programming is based on the concept of procedures or routines. It focuses more on the sequence of actions to be performed rather than the data. OOP promotes better modularity and makes code easier to maintain and scale compared to procedural programming.
Max Marks: 10

2. Describe the process of normalization in database design. Why is it important?
Ideal Answer:
Normalization is the process of organizing

data in a database to reduce redundancy and improve data integrity. It involves dividing a database into two or more tables and defining relationships between the tables. The main goal is to isolate data so that additions, deletions, and modifications can be made in just one table and then propagated through the rest of the database via relationships. Normalization typically progresses through normal forms: 1NF, 2NF, 3NF, and so on. It is important because it ensures consistency, eliminates duplicate data, and makes the database more efficient and easier to maintain.
Max Marks: 8

3. What is a deadlock in operating systems? Explain its necessary conditions and prevention techniques.
Ideal Answer:
A deadlock is a state in an operating system where a set of processes are blocked because each process is holding a resource and waiting for another resource held by another process.

The four necessary conditions for deadlock are:

- Mutual Exclusion
- Hold and Wait
- No Preemption
- Circular Wait

Deadlock can be prevented by ensuring that at least one of these conditions is not allowed. Techniques include resource ordering, avoiding hold and wait by requesting all resources at once, and allowing preemption.

Max Marks: 9

4. What are the key differences between compiler and interpreter?

Ideal Answer:

A compiler translates the entire source code into machine code before execution, while an interpreter translates and executes the code line by line.

Compiler:

- Faster execution after compilation
- Generates an executable file
- Errors are shown after full compilation

Interpreter:

- Slower execution since it translates during runtime
- No executable file generated
- Errors are shown line by line, making debugging easier
- Examples: C uses a compiler, Python uses an interpreter.

Max Marks: 7