

Part A: Conceptual Questions

Definition of a Class and an Object

1. **What is a class in object-oriented programming?**

A class is a blueprint or template for creating objects. It defines data members and member functions or methods that objects of that class will have.

2. **What is an object, and how does it relate to a class?**

An object is an instance of a class, It has its own values for the attributes defined by the class and can perform actions using the class methods.

Constructors and Destructors

3. **Define a constructor. What is its role in a class?**

A constructor is a special function in a class that is automatically called when an object of the class is created. Its role is to initialize the object's attributes and ensure it starts in a valid state.

4. **Define a destructor. Why is it important in managing an object's lifecycle?**

A destructor is a special function that is automatically called when an object is destroyed or goes out of scope. It is important because it releases any allocated resources (e.g., memory, file handles) to prevent resource leaks.

Object Lifecycle

5. **Briefly describe the lifecycle of an object from instantiation to destruction.**

- **Instantiation:** The object is created using the class constructor, initializing its attributes.
- **Usage:** The object performs operations using its methods.
- **Destruction:** When the object goes out of scope or is explicitly deleted, the destructor is called to clean up resources before the object is removed from memory.

6. **Why is it important for a class to manage its resources (e.g., memory) during its lifecycle?**

Proper resource management prevents memory leaks, dangling pointers, and other issues that can lead to inefficient or unstable programs. In languages like C++, where garbage collection is not automatic, managing memory manually is crucial.

Part B: Minimal Coding Example (C++ Code)

```
class Creature {
private:
    string name;
    int health;

public:
    // Constructor
    Creature(string creatureName, int creatureHealth) {
        name = creatureName;
        health = creatureHealth;
        cout << "Creature " << name << " is created with " << health << " health." << endl;
    }

    // Method to display creature details
    void display() {
        cout << "Creature Name: " << name << ", Health: " << health << endl;
    }

    // Destructor
    ~Creature() {
        cout << "Creature " << name << " is being destroyed." << endl;
    }
};
```

The Creature class demonstrates the object lifecycle:

Constructor: Initializes name and health when an object is created.

Destructor: Prints a message when the object is destroyed.

Object Lifecycle: The Creature object is created inside main(), used for displaying details, and then destroyed automatically when the program ends.

Part C: Reflection & Short Answers

Importance of Constructors:

Constructors ensure that an object starts with valid values, preventing uninitialized variables and maintaining program stability. Without constructors, objects might contain garbage values leading to unpredictable behavior.

Role of Destructors:

Destructors are necessary for releasing dynamically allocated resources such as memory and file handles.

Destructors help prevent memory leaks and resource exhaustion.

Lifecycle Management:

If a class does not properly manage its resources, it can lead to memory leaks, crashes, or excessive resource consumption.

- example, failing to delete dynamically allocated memory will cause memory leaks, reducing performance over time.