Turram Toussaint
2143 OOP
Prof Griffin

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 attributes (data members) and behaviors (methods) that describe the object.

2. **What is an object, and how does it relate to a class?**
   An object is an instance of a class. It represents a specific entity with the attributes and behaviors defined by its class.

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 is instantiated. Its role is to initialize data members and set up any necessary state for the object.

4. **Define a destructor. Why is it important in managing an object's lifecycle?**
   A destructor is a special function in a class that is automatically called when an object goes out of scope or is deleted. It is important for releasing resources, such as memory or file handles, to prevent resource leaks.

Object Lifecycle

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

   Instantiation: The constructor is called, and memory is allocated for the object.

   Usage: The object performs its tasks through its methods and properties.

   Destruction: The destructor is called when the object goes out of scope or is explicitly deleted, releasing resources.

6. **Why is it important for a class to manage its resources (e.g., memory) during its lifecycle?**
   Proper resource management prevents memory leaks and ensures efficient use of system resources, improving performance and stability of the application.

Part B: Minimal Coding Example

```
#include <iostream>
#include <string>

class Creature {
private:
    std::string name;
    int health;
public:
    // Constructor
    Creature(std::string creatureName, int creatureHealth) : name(creatureName),
health(creatureHealth) {
        std::cout << "Creature " << name << " created with " << health << " health.\n";
    }

    // Method to display creature's state
    void display() {
        std::cout << "Creature: " << name << ", Health: " << health << "\n";
    }

    // Destructor
    ~Creature() {
        std::cout << "Creature " << name << " is being destroyed.\n";
    }
};

int main() {
    Creature goblin("Goblin", 100);
    goblin.display();
    return 0;
}
```

Explanation:
This class defines a creature with a name and health. The constructor initializes these attributes, ensuring that the object starts in a valid state. The destructor outputs a message when the object is destroyed, demonstrating resource cleanup. The object lifecycle is managed by automatic invocation of the constructor upon creation and the destructor upon destruction.

Part C: Reflection & Short-Answer

1. **Importance of Constructors:**
   Constructors ensure that objects start their being in a valid state by initializing essential data members, preventing uninitialized variables that could lead to it having an undefined behavior.

2. **Role of Destructors:**
   Destructors are necessary, especially in languages like C++, where memory management is manual. They help release dynamically allocated memory, close file handles, and free other resources to prevent leaks.

3. **Lifecycle Management:**
   If a class does not properly manage its resources, it can lead to issues like memory leaks, resource exhaustion, and undefined behavior. This can degrade system performance and lead to crashes or instability in programs.