Part A: Conceptual Questions

1. **Definition of a Class and an Object**
   - Class **-** A programmer-defined data type that encapsulates related data, or member variables, and functions that operate on that data (methods).
   - Object **-** An instance of a class that holds data and behaves according to the class's methods.
2. **Constructors and Deconstructors**
   - Constructor - a public class method that is used to initialize data members.
   - Deconstructor - a public class method that releases data held by an object after it is destroyed. Destructors free up dynamically allocated memory to avoid memory leaks.
3. **Object Lifecycle**
   - An object is instantiated (created) by using the class to initialize the object's attributes, which involves allocating memory for the object. The object then calls its methods and performs its intended function in a program. When the object is no longer needed, a destructor is used to deallocate the memory held by the object.
   - It's important for a class to manage its resources to prevent memory leaks, avoid unexpected behavior, and ensure proper performance.

Part B: Minimal Coding Example

```
class Creature{
private:
    int health;

public:
    Creature(int h = 100){
        health = h;
    }

    ~Creature(){
        cout << "Creature is being destroyed" << endl;
    }

    void displayState(){
        cout << "Creature's health: " << health << endl;
    }
};
```

My Creature class uses a constructor to initialize health when an object is created. A destructor is used to ensure that the Creature object's data is deallocated when

destroyed. The constructor and destructor manage the object lifecycle by handling object initialization and memory deallocation.

Part C: Reflection & Short-Answer

1. **Importance of Constructors:**
    - Constructors ensure that an object starts its lifecycle in a valid state by initializing its data members with proper values. They assign objects with default or user-specified values, preventing the objects from being incomplete and guaranteeing that the object is ready for use.
2. **Role of Destructors:**
    - Destructors are necessary because they ensure proper deallocation of resources when an object's lifecycle is over. By releasing dynamically allocated memory as well as other resources held by no longer needed objects, destructors prevent memory leaks and ensure system stability.
3. **Lifecycle Management:**
    - If a class does not properly manage its resources during its lifecycle, several issues could arise such as memory leaks, undefined behavior, and security risks. These risks are why constructors and destructors are essential for ensuring stability and reliability in programs.