**Part A: Conceptual Questions**

1. Definition of a Class and an Object
    - A class is a blueprint or template for creating objects. It defines the properties (attributes or data members) and behaviors (methods or functions) that its objects will have.
    - An object is an instance of a class. It represents a specific realization of the class blueprint with its own unique data. The class provides the structure, while the object holds the actual data.

2. Constructors and Destructors
    - A constructor is a special member function of a class that is automatically called when an object is created. Its role is to initialize the object's data members and establish a valid state for the object.
    - A destructor is a special member function of a class that is automatically called when an object is destroyed. It is used to release resources (memory, file handles) that the object might have acquired during its lifecycle, ensuring proper cleanup and preventing resource leaks.

3. Object Lifecycle
    - When an object is created, the constructor is invoked to initialize its state. The object is used to perform tasks, accessing its methods and modifying its attributes. When the object's lifetime ends (it goes out of scope), the destructor is invoked to release its resources.
    - Proper resource management prevents issues like memory leaks, dangling pointers, or resource contention. This ensures the program runs efficiently and avoids crashes or undefined behavior.

**Part B: Minimal Coding Example**

```
class Creature {
private:
   string name; // Private data member to hold the creature's name

public:
   // Constructor
   Creature(string creatureName) {
      name = creatureName;
   }
```

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

    // Public method to display the creature's state
    void displayState() const {
        cout << "This creature's name is: " << name << endl;
    }
};
```

- Explanation
    - The Creature class uses a constructor to initialize the name of the creature when the object is created. The destructor is automatically invoked when the object is destroyed, printing a message to indicate cleanup. The lifecycle is managed by ensuring the object's name is initialized during construction and cleaned up properly during destruction.

**Part C: Reflection & Short-Answer**

1. Importance of Constructors
    - Constructors ensure that an object starts in a valid and predictable state by initializing its attributes. This reduces the risk of bugs caused by uninitialized data.

2. Role of Destructors
    - Destructors are vital for releasing resources like dynamically allocated memory or file handles. In languages like C++, which don’t have automatic garbage collection, destructors help prevent memory leaks and ensure the program cleans up after itself.

3. Lifecycle Management
    - If a class does not properly manage its resources, issues like memory leaks, resource contention, or dangling pointers can occur. These issues can cause inefficiency, instability, or crashes in the program.