PART A:

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

A class is like a blueprint for creating objects. Think of it as a template that defines the characteristics (attributes) and behaviors (methods) of the class.Example: A "Date" class could define common attributes like day, month, and year, and behaviors like displayDate(), addDays(), and isLeapYear().

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

An object is an instance of a class, meaning it is a real-world entity created using the blueprint provided by the class.Example: If Date is the class, then specific dates like "March 18, 2025" and "January 1, 2000" are objects of that class, each with its own unique values for day, month, and year.

1. What is a Constructor? What is its Role in a Class?

A constructor is a special function that is called automatically when an object is created. Its main role is to initialize the object's attributes with default or specified values.

2. What is a Destructor? Why is it Important in Managing an Object's Lifecycle?

A destructor is a special function that is called automatically when an object is destroyed. It helps free resources (like memory) and ensures a clean exit.

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

Instantiation: The object is created using a constructor, which initializes its attributes.

Active State: The object is used, performing operations and interacting with other objects.

Destruction: When the object is no longer needed, the destructor is called to free up resources.

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

Prevent Memory Leaks: Avoid wasting memory by ensuring it's properly released when no longer needed.

Optimize Resource Usage: Ensure resources like files or connections are closed properly, preventing wastage.

Ensure Stability: Proper management avoids errors or crashes, keeping the program stable and efficient.

PART B

1. Design a simple class called Creature (or a D&D-themed class such as Goblin) that includes:

A private data member (e.g., health or name).

A constructor that initializes the data member(s).

A destructor that prints a message indicating the object is being destroyed.

A public method that displays the object's state.

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

using namespace std;

class Creature {
private:
    string name;  // Private data member for the creature's name
    int health;   // Private data member for the creature's health

public:
    // Constructor - Initializes the name and health of the creature
    Creature(string n, int h) : name(n), health(h)

    // Destructor - Prints a message when the object is destroyed
    ~Creature() {
    }
    // Public method to display the creature's state
    void displayState() {
    }
};

int main() {
    Creature goblin("Goblin", 100);  // Constructor is called
    goblin.displayState();

    return 0;
}
```

Constructor: Initializes the name and health when the object is created, ensuring it starts with valid data.

Destructor: Called automatically when the object goes out of scope, printing a message that it's being destroyed.

Object Lifecycle: The object is created, used, and then automatically destroyed when it goes out of scope, with the destructor cleaning up.

PART C

Importance of Constructors: Constructors ensure that an object is initialized with valid data by setting its attributes at creation. This prevents errors from uninitialized or incorrect values, making the object ready for immediate use.

The Role of Destructors: Destructors are essential, particularly in languages without automatic garbage collection. They manage resources (such as memory or file handles) when an object is no longer in use. Without destructors, these resources may remain unreleased, resulting in memory leaks or other complications.

Lifecycle Management: If a class doesn’t properly manage its resources, it can cause memory leaks, slow down the program, or even crash it. For example, if memory is allocated but not freed, the program will keep consuming more memory until it fails.