C++ Programming Assignment: Inheritance and Composition (Aggregation)

Objective:

The goal of this assignment is to introduce the concepts of inheritance and composition (aggregation) in C++. You will create a simple program that demonstrates both inheritance and aggregation by modeling a relationship between objects.

Difficulty Level:

Intermediate – This assignment is intended for students who have a basic understanding of object-oriented programming (OOP) concepts in C++ and want to practice inheritance and composition.

Assignment Description:

You are required to write a C++ program that implements inheritance and composition (aggregation). The program should model a simple system, such as a university or a library, where one class is a part of another class, and another class extends the functionality of an existing class.

Instructions:

1. Program Overview:

- You will design a program that involves at least three classes:
  - A base class.
  - A derived class that inherits from the base class.
  - A class that represents a part of another class (demonstrating composition).
- The program should demonstrate how inheritance and composition work together in object-oriented programming.

2. Step-by-Step Guide:

1. Design the Classes:

- Base Class: Create a base class that represents a generic entity. For example, if you are modeling a university, you could create a `Person` class that stores common attributes like name and age.

- Derived Class (Inheritance): Create a derived class that extends the base class. For example, a `Student` class that inherits from `Person`, and adds attributes like `studentID` and `major`.

- Class with Composition (Aggregation): Create a class that includes another class as its part. For example, a `Course` class that contains an object of the `Instructor` class. This demonstrates composition, as a `Course` is composed of an `Instructor`.

2. Demonstrate Inheritance:

- Show how a derived class inherits properties and methods from the base class, and how it can extend the functionality with its own properties or methods.

3. Demonstrate Composition (Aggregation):

- Show how an object of one class is used as a member (part) of another class.

3. Example Scenario:

You could model a university system with three classes:

- Class 1: `Person` (Base Class)**:

  - Attributes: `name`, `age`

  - Methods: `printInfo()`

- Class 2: `Student` (Derived Class using Inheritance)**:

  - Inherits from `Person`.

  - Additional Attributes: `studentID`, `major`

  - Methods: `printStudentInfo()`

- Class 3: `Course` (Class using Composition)**:

  - Attributes: `courseName`, `instructor`

  - `instructor` will be an object of the `Person` class (or you can create a separate `Instructor` class).

  - Methods: `printCourseDetails()`

4.Program Requirements:

- The program must compile and run without errors.

- Demonstrate both inheritance and composition.

- Include proper comments and indentation for readability.

- Output must be meaningful and clearly demonstrate the use of inheritance and aggregation.

Deliverables:

1. Source Code: Submit your C++ code in a `.cpp` file.

2. Program Output: Include a screenshot of the output from your program after it has been successfully executed.

Grading Criteria:

- Correctness of code (40%)

- Proper use of inheritance and composition (30%)

- Code readability, structure, and comments (15%)

- Output clarity (15%)

Good luck with your assignment!

Input:

```
Enter Instructor's Name: Sumanth
Enter Instructor's Age: 25
Enter Course Name: CISC125
Enter Student's Name: Yourname
Enter Student's Age: 20
Enter Student's ID: 1234
Enter Student's Major: Compsci
```

Output:

```
Course: CISC125
Instructor: Name: Sumanth, Age: 25
Name: Yourname, Age: 20
Student ID: 1234, Major: Compsci
```