LAB TASK 12- COMPOSITION AND AGGREGATION

Aggregation:

CONCEPT: Aggregation occurs when a class contains an instance of another class.

Q1: When designing software, it sometimes makes sense to create an object from other objects.

For example, suppose you need an object to represent a **Course** that you are taking in college. You decide to create a **Course** class, which will hold the following information:

• The course name
• The instructor's last name, first name, and office number
• The textbook's title, author, and publisher

In addition to the course name, the class will hold items related to the **instructor** and the **textbook**. You could put attributes for each of these items in the Course class. However, a good design principle is to separate related items into their own classes.

Note:

In this question, an Instructor class could be created to hold the instructor-related data and a TextBook class could be created to hold the textbook-related data. Instances of these classes could then be used as attributes in the Course class.

Problem 1:

A class named ***Processor*** has

- Two attributes i.e. ***processName*** and ***price***
- A ***parameterized constructor*** to initialize attributes with user-defined values

Class ***MainMemory*** consists of

- Two attributes i.e. ***size*** and ***price***
- A ***parameterized constructor*** to initialize attributes with user-defined values

Class ***MotherBoard*** has

- a data member named ***compName*** of type string
- a ***no-argument*** constructor to initialize with default name ***intel***

Design a class named ***Computer*** that includes

- A data member named ***proc*** of type ***Processor***
- A data member named ***ram*** of type ***MainMemory***
- A data member named ***mboard*** of type ***MotherBoard***
- A ***parameterized constructor*** that accept two arguments of type ***Processor*** and ***MainMemory*** to initialize members of these types. Moreover, within this constructor, instantiate object of ***MotherBoard*** to initialize ***mboard*** data field.

Write ***main()*** in a way that it clearly describes aggregation and composition relationships between objects of implemented classes.

Problem 2:

Consider six classes i.e. ***Person***, ***Professor***, ***Researcher***, ***Department***, ***Laboratory***, and ***University*** having following specifications**.**

Class ***University*** has

- Two attributes of type string i.e. ***universityName*** and ***location***
- An attribute named ***dept*** of type ***Department***

Class ***Department*** has

- Two attributes i.e. ***deptID***, ***deptName***
- A ***two-argument constructor*** to initialize data fields with user-defined values
- A member function ***display()*** to show all attribute values

Class ***Laboratory*** contains

- Two attributes i.e. ***labID*** and ***experimentNo***
- A ***two-argument constructor*** to initialize data member with user-defined values

Class ***Person*** has

- Two attributes i.e. ***name*** and ***age***
- A ***parameterized constructor*** to initialize attributes with user-defined values
- A member function ***display()*** to show its attribute values

Class ***Professor*** is derived from class ***Person*** and has

- A data field named ***profName*** of type ***string***
- A data field named ***dept*** of type ***Department***
- A two-argument constructor to initialize both attributes of user-defined values

Class ***Researcher*** is derived from class ***Professor*** and has

- An additional attribute named ***lab*** of type ***Laboratory***
- A constructor to initialize ***lab*** with user-defined value

a) Implement all these classes while illustrating the concept of aggregation and composition in terms of ownership and life-cycle.

Write following functions.

1. Write appropriate getter setter function for each Class.
2. Add/delete/update Department in University class