National University of Computer and Emerging Sciences

Lab Manual 10
Object Oriented Programming

Course Instructor	Mr. Bismillah Jan
Lab Instructor (s)	Mr. Saif Ali Mr. Dilawar Shabbir
Section	BCS-2E
Semester	Spring 2021

Department of Computer Science
FAST-NU, Lahore, Pakistan

Objectives

After performing this lab, students shall be able to understand and implement:

- Dependency, Association, Aggregation, Composition

Task 1

Exercise 1:

Make a class **Student** that has following data members:

```
char name[50];
char rollNo[8];
float cgpa;
```

Students class provides a constructor with default arguments and a function **Print** that prints students name and roll number on screen in following format:

StudentName (RollNo.)

For Example, Aslam Baig (12L9356)

Do we need a Destructor for this class? **Mention Reason**

Exercise 2:

Write the following piece of code in your main function: it should create six students with information provided.

```
Student s1("12L1111", "Hashim Amla", 3.99);
Student s2("13L1121", "Virat Kohli", 3.45);
Student s3("13L1126", "Quinton de Kock", 2.98);
Student s4("14L1361", "Joe Root", 2.99);
Student s5("14L1124", "Martin Guptil", 3.09);
Student s6("15L1314", "Rohit Sharma", 3.19);
```

Exercise 3:

A Student **Society** has a president and five members from students. Make a class Society that has following private data members:

```
char name[50];
Student* president;
Student* members[5];
```

The Society class has a constructor with default arguments that takes the name of society as parameter. Why are we keeping Student pointers in Society class and what should the constructor do?

Exercise 4:

Write a member function of Society class **PrintInfo** that prints name of society and details of its members and president using the **Print** function of Student class. What should function do if some member does not exist?

Exercise 5:
Add following lines in your main function it should give following output:

```
Society sports ("Sports");

sports.PrintInfo();
```

Output:

```
Society Name:   Sports
President:      Not Available
Member 1:       Not Available
Member 2:       Not Available
Member 3:       Not Available
Member 4:       Not Available
Member 5:       Not Available
Press any key to continue . . .
```

Why is it displaying Not Available in members' information? Because president and members pointers are currently pointing to NULL. We need to point these pointers to students' objects in order to create association between sports society and students.

Exercise 6:
Make a member function **AppointPresident** in Society class that takes a student object by reference and appoints it to president's position if the position is vacant and the cgpa of student is above 3.00. Display appropriate error message otherwise. Do you need to add Getters in Students class to accomplish this task?

Exercise 7:
Add following lines in your main function and verify the output:

```
sports.AppointPresident(s3);
sports.AppointPresident(s1);
sports.AppointPresident(s2);
```

Output:

```
...
Quinton de Kock cannot be appointed as President. CGPA criteria not met.
Hashim Amla has been appointed as President.
Virat Kohli cannot be appointed as President. President position is NOT vacant.
Press any key to continue . . .
```

Note: AppointPresident need to call GetName of student in order to print this message.

Exercise 8:
Make a member function **AddMember** in Society class that takes a student pointer and adds it in the list of members if there is any position vacant and displays the error message otherwise. Also

the president cannot be in the list of society members. Also a student cannot be added as a member more than once.

Exercise 9:
Add following lines in your main function and verify the output:

```
cout << endl << endl << endl;
Student s7("15L1334", "Robert Elen", 2.19);
sports.AddMember(s3);
sports.AddMember(s2);
sports.AddMember(s3);
sports.AddMember(s1);
sports.AddMember(s4);
sports.AddMember(s5);
sports.AddMember(s6);
sports.AddMember(s7);
sports.PrintInfo();
```

Output:

```
...
Quinton de Kock has been added to members list successfully.
Virat Kohli has been added to members list successfully.
Quinton de Kock already exists in Members list.
President cannot be added in Members list.
Joe Root has been added to members list successfully.
Martin Guptil has been added to members list successfully.
Rohit Sharma has been added to members list successfully.
Robert Elen cannot be added to members list. Member position is NOT vacant.

Society Name:   Sports
President:      Hashim Amla (12L1111)
Member 1:    Quinton de Kock (Roll No here)
Member 2:    Virat Kohli (Roll No here)
Member 3:    Joe Root (Roll No here)
Member 4:    Martin Guptil (Roll No here)
Member 5:    Rohit Sharma (Roll No here)
Press any key to continue . . .
```

You may keep currentMembersCount in your Society class. Should it be a static data member or non-static?
Note: This is responsibility of Student Class to print its name and roll number, use Print of Student class here.

Task 2

```
class Tyre
{
private:
      private:
      int* width;
      int* aspect_ratio;
      int* diameter;
public:
//Constructors, Getters and Destructor
void PrintTyre();
};
class Car
{
public: private:
      int* model;
      char* company;
      tyre* t1;
public:
Constructors, Destructor
void PrintCar();
};
```

Your **main()** should contains following lines. You can add more code, but these lines should be included. Define and use display function of both the classes such that it shows Car, Tyre relationship.

```
tyre tNew(12, 10, 13);
car cNew(2016,"Honda",tNew);
```

TASK 3

Exercise 1:
Define and implement a class Point in files Point.h and Point.cpp, respectively. This class should provide:

- Two private integer data members x and y which will store the x and y coordinates of a point
- A default constructor which takes two parameters to initialize the x and y coordinates and prints "Point() called" on the screen.
- A function print() which prints out the point on the screen in the format (x,y)
- A destructor which prints "~Point() called" on the screen.

Exercise 2:

Now define and implement a class Circle in files Circle.h and Circle.cpp. This class should contain:

- A private data member center which will be an instance of the Point class
- A private float data member radius that will store the radius of the circle
- A constructor which takes three parameters (x and y coordinates of the center of the circle, and the radius) and initializes the data members accordingly and also prints "Circle() called" on the screen.
- A destructor which prints "~Circle() called" on the screen.
- A function print() which prints the information (center and radius) of the circle on the screen

To call the constructor of class Point from the constructor of class Circle, you can use the following syntax.

```
Circle::Circle(int x, int y, float r): center(x,y) { … };
```

Add another file Lab.cpp in your project. Copy the following piece of code in that file, compile and then execute. Note down the output of the program and write it in comments in the code.

```
#include "Circle.h"

void main()
{
	Circle c (3,4,2.5);
	c.print();
}
```

Exercise 3:

Define and implement a class Quadrilateral in files Quadrilateral.h and Quadrilateral.cpp. This class should provide:

- Four private data members w, x, y and z (Point type) which will be indicating the four corners of the quadrilateral.
- A constructor which takes eight parameters (x and y coordinates of the four corners) and initializes the data members accordingly and prints "Quadrilateral () called" on the screen.
- A destructor which prints "~Quadrilateral called" on the screen.
- A function print () which prints out the information (i.e. the coordinates of its four corners) of the quadrilateral object on the screen.

Exercise 4:

Modify the Lab.cpp file to instantiate an object of class Quadrilateral called obj with parameters for points (1, 0) (0, 1), (1, 1) and (0, 0) and call its print function. Note down the output of the program and write it in comments in the code.