**National University of Computer and Emerging Sciences**



**Laboratory Manual**

*for*

**Object Oriented Programming Lab**

| Course Instructor | Mr. Uzair Naqvi |
| --- | --- |
| Lab Instructor(s) | M Hashir, Seemab Ayub |
| Section | BCS-2B |
| Date | Tuesday, 9 April 2024 |
| Semester | Spring 2024 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. operator overloading

# Associations:

**Exercise 1:**

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 2:**

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 3:**

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.

# Composition:

### Exercise 1:

Make a new application called Lab\_<your roll number>. 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:

Class Circle{

Point center;

Float radius;

};

* 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();

}

# Aggregation:

### Exercise 1:

Deep copy with Aggregation:

* + Design classes with aggregation relationships.
  + Implement a deepCopy() function in the container class that creates a complete copy of itself, including the contained objects.
  + Ensure proper memory management using appropriate constructors, destructors, and copy constructors (if relevant).

### Exercise 2:

Student and Address:

* + Design classes Student and Address using aggregation. Student should have attributes like name, roll number, and an Address object.
  + Implement member functions in Student to get/set student information and access address details (e.g., print address).