|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object-Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated:** |
| **Teaching Assistant: Zain** | **Semester: Spring 2023** |
| **Lab Engineer: Rana Hamza Shakil** | **Batch: BSCE2022** |

# **Lab 13 B. Problem-Based Learning in C++**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
|  |  |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming in C++.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA), and lab attendants.

## **Theory and Backgrou****nd**

Polymorphism refers to the ability of objects to take on multiple forms. In C++, there are two types of polymorphism: run time polymorphism and compile time polymorphism. Run time polymorphism occurs when a function call is resolved at run time based on the actual type of the object that the function is called on. This is achieved through the use of virtual functions and the "virtual" keyword in C++. Compile time polymorphism occurs when a function call is resolved at compile time based on the number and types of arguments passed to the function. This is achieved through the use of function overloading and operator overloading.

Moreover, Function overloading is a feature of C++ that allows multiple functions with the same name to be defined in the same scope. The functions must have different parameter lists or different types of parameters. When a function call is made, the correct function is selected based on the number and types of arguments passed to the function. Function overriding occurs when a derived class provides its own implementation of a virtual function that is already defined in its base class. When a function call is made on an object of the derived class, the overridden function is called instead of the base class function.

**Lab Task**

**Task A [Marks: 5]**

Please follow the following steps before starting the below tasks:

1. Create a separate header file (**.h file**) for each class declaration of data members and member functions

2. Create a separate source file (**.cpp file**) for the implementation of the class member functions.

3. Create **main.cpp** file for creating objects of the class and other driving codes.

**Task B: [Marks: 35]**

Step 1: Implement a class hierarchy for geometric shapes

In this task, you are required to implement a class hierarchy for geometric shapes. You should define a base class called "Shape" that has virtual functions for calculating the area and perimeter of the shape. You should then create several derived classes, such as "Circle", "Square", and "Rectangle", that inherits from the Shape class and override the virtual functions to calculate the area and perimeter of each specific shape. This will demonstrate the use of run time polymorphism and function overriding.

Step 2: Create overloaded functions for manipulating geometric shapes

In this step, you will create overloaded functions for manipulating geometric shapes. You should create functions for calculating the area and perimeter of a single shape, as well as functions for calculating the area and perimeter of multiple shapes. You should also create functions for comparing the areas and perimeters of two shapes. This will demonstrate the use of compile-time polymorphism and function overloading.

Step 3: Define an abstract class for file operations

In this final step, you will define an abstract class for file operations. You should define a base class called "FileOperation" which has virtual functions for opening, reading, writing, and closing a file. You should then create several derived classes, such as "TextFile" and "BinaryFile", that inherit from the FileOperation class and override the virtual functions to implement the specific file operations for each type of file. You should also demonstrate the use of constructor overloading by creating deep and shallow copy constructors for the derived classes.

Note: You can use any additional features of C++ that you think are necessary to complete the lab task, such as templates, exceptions, and operator overloading.

|  |
| --- |
| // Paste your code here |

### Assessment Rubric for Lab

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva/Quiz | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_