|  |
| --- |
| **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 6B. Multiple Inheritance Vs Multilevel Inheritance**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 Background**

**Inheritance, Multiple Inheritance, and Multilevel Inheritance**

Inheritance is a fundamental concept in object-oriented programming, including in C++. It allows a new class to be based on an existing class, inheriting the properties and behaviors of the existing class and adding new ones as needed. The existing class is called the base class or parent class, and the new class is called the derived class or child class. In C++, inheritance is implemented using the syntax "class derived\_class : access\_specifier base\_class", where access\_specifier can be "public", "protected", or "private". This specifies the level of access that the derived class has to the members of the base class. The derived class can access the public and protected members of the base class as if they were its own members. Private members of the base class are not directly accessible by the derived class, but they can be accessed using public or protected member functions of the base class.

The main difference between multiple inheritance and multilevel inheritance is that multiple inheritance allows a class to inherit from two or more base classes directly. On the other hand, multilevel inheritance is a type of inheritance in which a derived class is created from another derived class, which is already derived from a base class. This creates a hierarchical structure, with each level of the hierarchy building upon the previous level. Multilevel inheritance can be useful for creating complex relationships between classes and can help in code reuse.

**Lab Task**

**Task A [Marks: 5]**

Please follow the following steps before starting 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 class and other driving code.

**Task B: Multiple Inheritance vs Multilevel Inheritance [Marks: 35]**

Step 1: Define the base class Person with protected data members and functions.

* Define the Person class with the following protected data members: name, age, and address.
* Define two protected member functions, setName() and setAge(), to set the name and age of the person.

Step 2: Implement multiple inheritance using classes Teacher and Student.

* Define the Teacher class with a protected data member, subject, and a protected member function, setSubject(), to set the subject that the teacher teaches.
* Define the Student class with a protected data member, rollNumber, and a protected member function, setRollNumber(), to set the roll number of the student.
* Use multiple inheritance to inherit both the Teacher and Student classes in a new class called TeachingAssistant.

Step 3: Implement multilevel inheritance using a class called GraduateTeachingAssistant.

* Define the GraduateTeachingAssistant class that inherits from the TeachingAssistant class.
* Define a protected data member, researchTopic, and a protected member function, setResearchTopic(), to set the research topic of the graduate teaching assistant.
* Test the GraduateTeachingAssistant class by creating an instance of it and calling its public member functions, including those inherited from the Person, Teacher, and Student classes. Verify that the data members are correctly set and displayed.

|  |
| --- |
| // 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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_