|  |
| --- |
| **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 12 A. 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**

C++ is a powerful object-oriented programming language that provides a range of features for building robust and extensible software applications. These features include inheritance, aggregation, polymorphism, composition, encapsulation, and dynamic memory allocation, which enable developers to create complex class hierarchies, implement flexible data structures, and manage resources efficiently. By utilizing these concepts, developers can build high-performance systems that provide secure and reliable functionality for a range of applications, including online flight booking systems.

**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]**

You are tasked with creating an Online Flight Booking System that allows users to search for flights, book tickets, and manage their bookings. The system should also allow administrators to manage flight schedules and view booking information.

Steps:

* Define a Flight class hierarchy that contains properties such as the flight ID, departure and arrival locations, departure time, and a vector to store the ticket reservations. Define a Ticket class hierarchy that contains properties such as the ticket ID, passenger name, flight ID, and price. Define a Reservation class that contains properties such as the reservation ID, the passenger’s name, the flight ID, and the number of tickets reserved.
* Implement inheritance in the Ticket class hierarchy by creating two derived classes: EconomyTicket and BusinessTicket. The EconomyTicket class should contain properties such as the seat number and baggage allowance, while the BusinessTicket class should contain properties such as the seat number, meal preference, and lounge access.
* Implement aggregation and polymorphism in the Flight class hierarchy by creating a function to add a Ticket object to a specific flight. The function should take a Ticket object as a parameter and add it to the Flight object's vector of ticket reservations. The function should also use dynamic casting to determine whether the Ticket object is an instance of EconomyTicket or BusinessTicket, and update the flight's properties accordingly.
* Implement composition in the Reservation class by creating a Flight object as a member variable. The Flight object should represent the flight associated with the reservation. Implement encapsulation in the Flight and Reservation classes by making their properties private and providing public getter and setter functions.
* Implement dynamic memory allocation by using new and delete operators to allocate and deallocate memory for Flight, Ticket, and Reservation objects.

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