National University of Computer and Emerging Sciences 0348

Lahore Campus

Object Oriented Paradigm Lab (CL1004)

Final Exam

Date: May 22, 2025

Instructor(s)

Ms. Amina Qaiser
Mr. Aqib Zeeshan
Mr. Durraiz Waseem
Ms. Eisha Khan
Mr. Hashir Mohsineen
Mr. Muhammad Hasan
Mr. Zaeem Ahmed

Total Time (Hrs):	**2.5**
Total Marks:	**40**
Total Questions:	**2**

Do not write below this line

Submission Path

Windows + R: \\Exam\Final Exam\Object Oriented Programming\Submissions\[your section]

Important Note

- Submit a **single .cpp file** named using your roll number e.g. **24L-1234.cpp**
- After submitting, **verify** your file is **visible** in the **submission folder of your section**, and check that the **file size is shown** next to it (to ensure it's not an empty 0 KB file)

Attempt all the questions.

CLO #3

Q1: Campus Transport Management System [25 marks]

Instructions:

- You may not use any STL containers (e.g., vector, map) or built-in string manipulation functions.
- Use dynamic memory wherever appropriate.
- No global variables allowed.
- No use of subscript notation []. Use pointer arithmetic instead.

Your university plans to launch a Campus Transport Management System to manage transport routes, drivers, and buses. You are hired to develop a C++ application for the administration department to manage this system using object-oriented principles.

1. System Requirements:
   - **Bus** [2 marks]
     - Unique Bus ID
     - Capacity
     - Assigned to a Route
     - Assigned to a Driver
   - **Driver** [1.5 marks]
     - Name
     - CNIC
     - License Number
   - **Route** [1.5 marks]
     - Route ID
     - Source
     - Destination
     - Timings
     - Total Distance in KMs
2. OOP Concepts and Functionalities to implement:
   - Constructors (default, parameterized, copy) [3+3+3 = 9 marks]
   - Destructors 3/ [1+1+1 = 3 marks]
   - Clean use of pointer arithmetic and input validations 2 [2 marks]
   - Implement == **operator** in the Route class. Two routes are considered equal if both their **source** and **destination match** (case-sensitive comparison). 2 [2 marks]
   - Assign a route and driver to a bus. 2 [1+1 = 2 marks]
   - Display all buses with their assigned driver and route details. 2 [2 marks]
   - Inheritance /Composition/ Aggregation (whichever is applicable)
   - Use char pointers as data members instead of strings (if applicable)
   - You must demonstrate the **creation of multiple Bus, Driver, and Route objects in your main code** (using dynamic memory). Use **dummy test data** directly in your code instead of prompting the user for input.

CLO #2

Q2: Autonomous Robot Control System [15 marks]

You are designing a control system for various autonomous robots deployed in specialized environments such as **medical facilities**, **agricultural fields**, and **combat zones**. Your design must reflect good object-oriented principles such as inheritance, polymorphism, aggregation/composition, and templates.

0348

1. Create an **abstract base class** Robot with the following **pure virtual functions**: [2 marks]
   - void performTask(): Each robot will carry out a unique mission-specific operation.
   - void statusReport(): Each robot will print its type and the status of its attached module.
2. Derive the following concrete classes: [6 marks]
   - MedicalRobot
   - AgriculturalRobot
   - CombatRobot

   Each class should **override** the two virtual functions:

   - performTask() should display a unique operation. For example:
     - MedicalRobot: "Assisting in surgery and patient monitoring."
     - AgriculturalRobot: "Monitoring soil health and planting seeds."
     - CombatRobot: "Scanning enemy zone and deploying countermeasures."
   - statusReport() should print the robot's **type** and **module** details (see Part 3).

3. Define a class **Module** representing an attached component such as a sensor or battery: [3 marks]
   - string name (e.g., "Thermal Sensor", "High-Capacity Battery")
   - int powerRating (Watts)

   Each robot must be initialized with its specific module at construction time.
   The module should be a **crucial internal component** of the robot, and its details should be included in the statusReport().

4. In **main()**, allow users to: [2 marks]
   - Add robots dynamically (prompt for type and module info)
   - View status reports of all robots

   Use a **dynamic array of pointers to Robot**, and **Memory management must be handled properly.**

5. Write a **templatized** function **analyzePerformance**(T& robot) that accepts any type of robot (Medical, Combat, etc.) and displays: [2 marks]

   "Running diagnostics on [robot's type]. It is used for [robot's task]"