National University of Computer and Emerging Sciences

Object Oriented Programming Lab (CL1004)

Comment is your best friend

Date:

Course Instructor(s)

Mr. Muhammed Monis, Mr. Talha Shahid

Lab Mid Exam (B)

Total Time: 90 minutes

Total Marks: 40

Total Questions: **03**

Semester: SP-2025

Campus: Karachi

Dept: Computer Science

LLO # 1: Understand and implement static data members, static member functions, constant members with data immutability

Q1. [10 marks |5 weightage]

You are tasked with designing a Library Book Management System that keeps track of the library's collection and circulation of books. Each book is assigned a unique book ID that automatically increments when a new book is added, and the system must also track the total number of books currently available for borrowing.

Your implementation should:

1. **Unique Book Identification:** Automatically assign an immutable, incrementing book ID to each new book added to the collection.
2. **Library-wide Consistency:** Ensure that the library name remains consistent across all book records.
3. **Borrowing and Returns:** Allow borrowers to check out and return books, updating each book's availability status.
4. **Efficient Memory Management:** Dynamically allocate and deallocate book records as needed, ensuring no memory leaks occur.
5. **Real-time Count Updates:** Accurately update the total number of available books when books are added, borrowed, or returned.

Implement your solution in C++ and test it by simulating multiple book additions, checkouts, and returns. Ensure that the total available book count updates correctly and that all memory is properly managed.

LLO # 2: Understanding and implementing Has-A Relationship concepts between classes with dynamic memory and object management.

Q2 [16 marks | 8 weightage]

You are tasked with designing a Hospital Doctor Management System that tracks doctors and their assigned departments, emphasizing the Has-a relationship between the Doctor and Department classes. Your system should address the following requirements:

1. **Department Management:**
   - Ensure each department is uniquely identified.
   - Maintain an up-to-date count of the total number of hospital departments.
2. **Doctor Management:**
   - Automatically assign a unique, auto-incremented doctor ID to every doctor.
   - Ensure each doctor has an immutable name.
   - Associate each doctor permanently with a specific department upon creation (establishing a Has-a relationship).
3. **Memory Management:**
   - Dynamically allocate memory for doctor objects when they are added to the system.
   - Properly deallocate memory when a doctor resigns, ensuring no memory leaks occur.
4. **Data Integrity and Constraints:**
   - Enforce that a doctor's name and assigned department cannot be modified after creation.
   - Dynamically update the total count of doctors and departments as changes occur.

Considerations:

- How will you implement an auto-incrementing, immutable doctor ID?
- What strategy will you use to dynamically track and update the count of doctors and departments?
- How can you enforce the permanent association between a doctor and a department?
- What measures will you take to handle memory allocation and deallocation safely?

Implement your solution in C++ and validate it by simulating various scenarios such as doctor assignments, department updates, and doctor resignations. Ensure that all counts update correctly and that memory is managed efficiently.

***LLO # 3:** Apply Inheritance (Is-a Relationship) in C++ to build an extensible University Staff Management System, using dynamic memory allocation, pointer references, and pure virtual functions for polymorphism.*

Q3 [14 marks | 7 weightage]
You are tasked with designing an F1 Racing Team Management System that efficiently manages different types of team members, such as Drivers and Engineers. Since all team members share common attributes, your implementation should avoid redundancy.

Your system should:
1. Manage team members, ensuring each has a unique identifier and an unchangeable name.
2. Differentiate between roles, such as Drivers (who have a car number and track race wins) and Engineers (who have a specialization).
3. Enforce structured behavior, ensuring that every team member can display their details appropriately.
4. Efficiently allocate and deallocate memory, properly managing dynamically created objects.
5. Demonstrate move semantics by utilizing move constructors where appropriate to ensure efficient object transfers.

Think About:
a. How does inheritance help in reducing redundancy when managing different team members?
b. How can you ensure that every team member has a unique and immutable ID?
c. What mechanism will enforce that all team members implement a function to display their details?
d. How should memory be handled when objects are dynamically allocated?

Implement your solution in C++ and validate it by simulating multiple drivers and engineers, tracking their details, and ensuring memory is properly managed. Use comments in your code to explain how inheritance reduces redundancy and improves maintainability. Also create an object an move an original object details to another while also deleting the old object.