University of Central Punjab
Faculty of Information Technology

DSA-Lab: Mid-Term Examination Fall-24

Course Code	CSCP2031-F24-BS-CS-S23-D7
Course Title	Data Structures and Algorithms - Lab
Credit Hours	4 (3+1)

CLO	CLO STATEMENT	Bloom's Taxonomy Level	PLO
1	Solve real-world problems skillfully with precision using programming constructs learned in theory with the course toolkit	P3	5

Course: Data Structures and Algorithms - Lab
Mid-Term Duration: 90 minutes
Total Marks: 20

Instructions:

1. This midterm consists of **three questions** which carry **20 marks**.
2. Complete the task within **90 minutes**.
3. Create a file with an **appropriate name**
4. Submit only **.h and .cpp** files on the portal.
5. Late submissions will **NOT** be considered
6. Create as many classes and functions as required. Remember **one function for one functionality.**
7. Take care, **plagiarism will not be tolerated in any case.**
8. **No . Rar/Zip** files are accepted
9. The **paper is a closed book and closed notes.** No cheat sheet allowed.
10. **Use meaningful** variable names, and take care of **naming conventions and indentation. 10% Marks will be deducted** for each thing if **not followed.**

Deadline:

- Submission **within the time no time will be extended**.
- Late submissions will incur a **15% mark deduction penalty will apply**.

Submission Platform:

- Upload your **`.cpp` file `.h`** and **`.doc`** with an **output** of **your** code then **submit it on your portal: https://horizon.ucp.edu.pk/**

Question 1: Develop a Book Management System Using Stack [6]

Scenario: A publishing company uses a stack data structure to manage books scheduled for printing. Each book contains the following information:

- **Book ID**: An integer that uniquely identifies the book.
- **Book Title**: A string representing the title of the book.

Task:

Design and **implement** a book management system using a class that employs stack operations with pointers for dynamic memory allocation. Your program should perform the following operations:

1. **Push**: **Add** a book to the stack.
2. **Pop**: **Remove** and **display** the book's details at the top of the stack.
3. **Peek**: **Retrieve** and **view** the details of the book currently at the top of the stack.
4. **Display**: **List** all books in the stack with their details.
5. **Size**: **Calculate** and **return** the total number of books currently in the stack.
6. **Exit**: **Terminate** the program.

Ensure the following conditions are handled:

- **Empty Stack**: Properly handle situations where the stack is empty during `Pop` or `Peek` operations.

```
1. Push -> Book ID: 101, Title: "Data Structures"
1. Push -> Book ID: 102, Title: "Algorithms"
3. Peek -> Output: Book ID: 102, Title: "Algorithms"
4. Display -> Output:
   Book ID: 102, Title: "Algorithms"
   Book ID: 101, Title: "Data Structures"
5. Size -> Output: 2
2. Pop -> Output: Book ID: 102, Title: "Algorithms"
5. Size -> Output: 1
6. Exit -> Output: Program Terminated.
```

Criteria	Marks
Correct Implementation of Operations **(Push, Pop, Peek, Display, Size)**	4
Proper Use of Pointers for Dynamic Memory Allocation	1
Handling Edge Cases (e.g., empty stack)	1

Question 2: Design a Toll Plaza Management System Using a Circular Queue [7]

Scenario: A toll plaza manages the flow of vehicles waiting to pay toll fees. Each vehicle has the following attributes:

- **Vehicle Number**: A string uniquely identifying the vehicle.
- **Vehicle Type**: A string indicating whether the vehicle is a "Car," "Truck," or "Bus."
- **Toll Fee**: An integer fee dependent on the vehicle type: $20 for Cars, $50 for Trucks, and $30 for Buses.

Task:
Construct and **implement** a toll plaza management system using a circular queue class with dynamic memory allocation. Include the following operations:

1. **Add Vehicle**: **Insert** a vehicle into the queue.
2. **Dispatch Vehicle**: **Remove** and **display** the vehicle at the front of the queue.
3. **Calculate Total Toll**: **Compute** and **display** the total toll fees collected from all processed vehicles.
4. **Search Vehicle**: **Locate** a vehicle by its number and **retrieve** its details.
5. **Display Queue**: **List** all vehicles in the queue with their details.
6. **Exit**: **Terminate** the program.

Ensure the following conditions are handled:

- **Full Queue**: Properly manage queue overflow.
- **Empty Queue**: Handle cases where operations like `Dispatch` or `Search` are performed on an empty queue.

Criteria	Marks
Correct Implementation of Operations **(Add, Dispatch, Calculate Total Toll, Search, Display)**	5
Proper Use of Pointers and Circular Logic	1
Handling Edge Cases (e.g., full/empty queue)	1

Question 3: Build an Employee Hierarchy Management System Using Singly Linked List [7]

Scenario: A company uses a singly linked list to maintain its employee hierarchy. Each employee has the following attributes:

- **Employee ID**: An integer uniquely identifying the employee.
- **Name**: A string containing the employee's name.
- **Designation**: A string indicating the role ("Manager," "Team Lead," "Developer").
- **Salary**: A float representing the employee's salary.

Task:
Develop and **implement** an employee management system using a class with a singly linked list. Perform the following operations:

1. **Add Employee**:
    - **Insert** Managers at the front of the list.
    - **Insert** Team Leads after the first half of the list.
    - **Insert** Developers at the end of the list.
2. **Remove Employee**: **Delete** an employee by their ID.
3. **Search Employee**: **Locate** an employee by their ID and **retrieve** their details.
4. **Update Salaries**: **Increase** salaries by:
    - 20% for Managers
    - 15% for Team Leads
    - 10% for Developers
5. **Sort Employees**: **Arrange** all employees in descending order of their salaries after updating salaries.
6. **Display Employees**: **List** all employees with their details.
7. **Exit**: **Terminate** the program.

Ensure the following conditions are handled:

- **Employee Not Found**: Appropriately manage cases where operations involve employees that do not exist in the list

Criteria	Marks
Correct Implementation of Operations **(Add, Remove, Search, Update Salaries, Sort, Display)**	5
Proper Use of Pointers for Dynamic Memory Allocation	1
Handling Edge Cases (e.g., non-existent employee)	1