

**DATA STRUCTURES**

**Spring 2025**

**LAB 07**

**Learning Outcomes**

In this lab you are expected to learn the following:

**Doubly & Circular Linked List**



**Lab Task 1: Singly Linked List – Emergency Patient list in a Hospital**

Perform Following deletion operations in the Lab Task you have performed in the previous Lab **i.e. Singly Linked List**.

**Operations**

**Deletion:**

* **Discharge** the first patient after treatment (delete from the beginning).
* **Remove the last patient** if they leave without treatment.
* **Remove a specific patient** if they are transferred to another hospital.

### **Output Example**

**After discharging the first patient:**

ID: 101, Name: Alice, Severity: 2, Arrival: 10:00 AM

ID: 105, Name: Eve, Severity: 4, Arrival: 10:20 AM

ID: 102, Name: Bob, Severity: 3, Arrival: 10:05 AM

ID: 103, Name: Charlie, Severity: 1, Arrival: 10:10 AM

**After removing last patient:**

ID: 101, Name: Alice, Severity: 2, Arrival: 10:00 AM

ID: 105, Name: Eve, Severity: 4, Arrival: 10:20 AM

ID: 102, Name: Bob, Severity: 3, Arrival: 10:05 AM

**After transferring patient ID 102:**

ID: 101, Name: Alice, Severity: 2, Arrival: 10:00 AM

ID: 105, Name: Eve, Severity: 4, Arrival: 10:20 AM

**Lab Task 2: Doubly Linked List – Movie Ticket Booking System**

You are developing a simplified **Movie Ticket Booking System** for a small cinema. The cinema screens multiple movies daily, and each movie has a **linked list of booked seats** stored in a **doubly linked list**. Each node in the list represents a booked seat, containing details such as **seat number and customer name**.

To efficiently manage booked seats, implement the following operations using a doubly linked list:

**Operations**

1. **Traversal in a Doubly Linked List:**
   * Display all booked seats in forward and reverse order.
2. **Finding the Length of a Doubly Linked List:**
   * Calculate and display the total number of booked seats.
3. **Insertion in a Doubly Linked List:**
   * **At the start:** A customer books the first available seat.
   * **At the end:** A customer books the last available seat.
   * **At a specific position:** A customer books a specific seat in the cinema hall.
4. **Deletion in a Doubly Linked List:**
   * **At the start:** A customer cancels their earliest booked seat.
   * **At the end:** A customer cancels their last booked seat.
   * **At a specific position:** A customer cancels a specific seat booking.
5. **Searching for a Specific Value:**
   * Search for a customer’s booked seat by their name or seat number.

**Implementation Details:**

* Use Object Oriented Programming.
* Implement functions for all the above operations.
* Provide a **menu-driven system** where users can select an operation to perform on the booked seats list.

**Lab Task 3: Circular Linked List – Automated Customer Service Call Handling System**

A **telecom company** wants to automate its **customer service call handling system** to manage incoming support calls in a **circular queue-like manner**.

Each incoming customer call is represented as a **node in a Circular Linked List**. Since call centers operate on a **round-robin scheduling** method (where each call is handled one by one in a cyclic order), a **Circular Linked List** is the perfect data structure for this system.

**Operations**

1. **Traversal in a Circular Linked List (Processing Calls in Order)**

* Display the list of all active customer calls in a round-robin manner (from the current call to the last and then looping back).

1. **Finding the Length of the Circular Linked List (Total Active Calls)**
   * Determine and display the number of active calls currently in the system.
2. **Insertion in the Circular Linked List (New Call Arrivals):**
   * **At the start:** A new high-priority customer call is added at the beginning of the queue.
   * **At the end:** A new customer call joins the queue normally at the end.
   * **At a specific position:** A VIP customer’s call is inserted at a specific position in the queue for faster resolution.
3. **Deletion in the Circular Linked List (Call Completion or Drop-Off):**
   * **At the start:** The first call in the queue is completed and removed.
   * **At the end:** The last call in the queue is completed and removed.
   * **At a specific position:** A customer drops their call before getting assistance and is removed from the queue.
4. **Searching for a Specific Call (Tracking a Customer's Call Status):**
   * Search for a specific call using a **Customer ID** or **Phone Number** to see if it is still in the queue.

**Example Use Case:**

1. New customer calls arrive and get added to the queue → Insert at the end
2. A VIP customer call enters and gets priority handling → Insert at a specific position
3. A customer's call is completed, so they are removed → Delete from the start
4. A customer hangs up before getting assistance → Delete from a specific position
5. Find out how many calls are currently in the queue
6. Check if a specific customer's call is still in the system
7. Display the order of call handling using circular traversal (round-robin manner)
8. You can make use cases for other operations as well like insertion at start, deletion at end etc.

***UNGRADED HOME TASK:***

*Implement a* ***doubly circular linked list*** *with all the basic operations i.e. Insertion, Deletion, Traversal, Searching, Finding length of the list.*

**SUBMISSION GUIDELINES:**

* Create a zip file named it as zzi-xxxx\_Name.
* Include all the .cpp files in the above created zip file and submit it to GCR Only.
* Late Submission will Not be Graded.
* Submission via email will NOT be Considered.