Data Structures and Algorithms — Lab 6

Objective

- Understanding Linked List
- Working with Linked List
- Applications of Linked List

Tasks

Students are required to complete the following tasks in lab timings.

Linked List Explanation

A linked list is a data structure in which each node contains two fields: one to store the data, and another to store a pointer to the next node in the list. In a singly linked list, each node only points to the next node, while in a doubly linked list, each node points to both the next and the previous nodes.

When implementing a linked list using head and tail pointers, we add two pointers to the class: one to point to the first node in the list (head pointer) and another to point to the last node in the list (tail pointer). This allows for more efficient insertion and deletion operations at both ends of the list.

It's important to note that not all linked lists have both head and tail pointers. In this lab, we are working with a linked list with both a head and a tail pointer for more efficient insertion and deletion operations at both ends.

The LinkedList class has two attributes:

1. `Node<T> *head` - This is a pointer to the first node in the linked list.
2. `Node<T> *tail` - This is a pointer to the last node in the linked list.

There are several member functions inside the LinkedList class:

1. `void insertAtHead(T data)` - This function is used to insert a new node at the beginning of the linked list.
2. `void insertAtTail(T data)` - This function is used to insert a new node at the end of the linked list.
3. `void insertSorted(T data)` - This function is used to insert a new node at the appropriate position in a sorted linked list.
4. `bool deleteFromHead()` - This function is used to delete the first node from the linked list.
5. `bool deleteFromTail()` - This function is used to delete the last node from the linked list.
6. `bool deleteValue(T data)` - This function is used to delete a specific value from the linked list.
7. `bool isEmpty()` - This function is used to check if the linked list is empty.
8. `bool search(T data)` - This function is used to search for a specific value in the linked list.
9. `void display()` - This function is used to display the contents of the linked list.

To use the `LinkedList<T>` class, you need to create your own class, `MyLinkedList<T>`, which inherits from the `LinkedList<T>` class. `MyLinkedList<T>` should be a template class that takes a type `T` as a parameter.

Task 1

Write a C++ program that implements a singly linked list using head and tail pointers. The linked list should have the following functions. The program should provide a menu-based interface with the following options:

1. Add node to the linked list
2. Insert node in sorted order
3. Delete a node from the linked list
4. Search for a node in the linked list
5. Display the linked list
6. Exit

Explanation of the menu:

1. When the user selects option 1, the program should prompt the user to enter the data of the node and add it to the linked list.
2. When the user selects option 2, the program should prompt the user to enter the data of the node and add it to the linked list in sorted order.
3. When the user selects option 3, the program should prompt the user to enter the data of the node to be deleted and delete it from the linked list.
4. When the user selects option 4, the program should prompt the user to enter the data of the node to be searched and display whether or not it is present in the linked list.
5. When the user selects option 5, the program should display all the nodes in the linked list.
6. The program should exit when the user selects option 6.

Note: You are required to implement the following functions as well, though, there isn't a menu entry for performing these operations:

1. `void insertAtHead(T data`
2. `void insertAtTail(T data)`
3. `bool deleteFromHead()`
4. `bool deleteFromTail()`

Task 2

Write a C++ program that implements a stack using a linked list. The stack should have the following functions: `push(T), pop(), top(), isEmpty(),` and `display()`. The program should provide a menu-based interface with the following options:

1. Push element onto the stack
2. Pop element from the stack
3. Display the top element of the stack
4. Display the stack
5. Check if the stack is empty
6. Exit

Explanation of the menu:

1. When the user selects option 1, the program should prompt the user to enter the data of the element and add it to the top of the stack.
2. When the user selects option 2, the program should remove the top element from the stack.
3. When the user selects option 3, the program should display the top element of the stack without removing it.
4. When the user selects option 4, the program should display all the elements in the stack.
5. When the user selects option 5, the program should check if the queue is empty or not and display the result.
6. The program should exit when the user selects option 6.

Task 3

Write a C++ program that implements a queue using a linked list. The queue should have the following functions: `enqueue(T), dequeue(), front(), isEmpty(),` and `display()`. The program should provide a menu-based interface with the following options:

1. Enqueue an element to the queue
2. Dequeue an element from the queue
3. Get the front element of the queue
4. Check if the queue is empty
5. Display the queue
6. Exit

Explanation of the menu:

1. When the user selects option 1, the program should prompt the user to enter the data of the element and add it to the queue.
2. When the user selects option 2, the program should remove the first element from the queue and display it.
3. When the user selects option 3, the program should display the first element of the queue without removing it.

4. When the user selects option 4, the program should check if the queue is empty or not and display the result.
5. When the user selects option 5, the program should display all the elements in the queue.
6. The program should exit when the user selects option 6.

Task 4

Write a C++ program that deletes duplicate values from a linked list. The program should provide a menu-based interface with the following options:

1. Add a value to the linked list
2. Delete duplicate values from the linked list
3. Display the linked list
4. Exit

Explanation of the menu:

1. When the user selects option 1, the program should prompt the user to enter the value and add it to the linked list.
2. When the user selects option 2, the program should delete all duplicate values from the linked list and display the updated list.
3. When the user selects option 3, the program should display all the values in the linked list.
4. The program should exit when the user selects option 4.

Task 5

Write a C++ program that counts the number of nodes in a linked list. The program should provide a menu-based interface with the following options:

1. Add node to the linked list
2. Count the number of nodes in the linked list
3. Display the linked list
4. Exit

Explanation of the menu:

1. When the user selects option 1, the program should prompt the user to enter the data of the node and add it to the linked list.
2. When the user selects option 2, the program should count the number of nodes in the linked list and display the count.
3. When the user selects option 3, the program should display all the nodes in the linked list.
4. The program should exit when the user selects option 4.