# 

|  |  |  |
| --- | --- | --- |
| **Lab 08** | |  |
| **Topic** | * Abstract Classes * Singly LinkedList * Singly Circular LinkedList * Doubly LinkedList * Doubly Circular LinkedList * LinkedList Application |
| **Objective** | * The basic purpose of this lab is to implement ADT of LinkedList, and test its applications. |

**Instructions:**

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.
* Name of the program should be same as the task name. i.e. the first program should be Task\_1.cpp

# void main() is not allowed. Use int main()

* **You have to work in multiple files. i.e separate .h and .cpp files**

# You are not allowed to use system("pause")

* **You are not allowed to use any built-in functions**

# You are required to follow the naming conventions as follow:

* + **Variables:** firstName; (no underscores allowed)
  + **Function:** getName(); (no underscores allowed)
  + **ClassName:** BankAccount (no underscores allowed)

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

**Task 1**

Create a C++ generic abstract class named as **LinkedList**, with the following:

**Attributes:**

1. Node \*head
2. Node \*tail

**Functions:**

virtual void insertAtTail(int) = 0;

* + Should add element at the tail node of linkedList.

virtual void deleteAtTail() = 0;

* + Should delete element at the tail node of linkedList.

LinkedList();

# Singly + Singly Circular LinkedList:

Linked List is type of data structure which contains some objects called nodes that are randomly stored in the memory. In singly linkedlist, node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory. The last node of the list contains pointer to the null.

In singly Circular linkedlist, a node contains two fields i.e. data stored and pointer which contains the address of the next node. While the next pointer of the tail node points to the head node.

1. **Use the above class to make a derived class named as SinglyLinkedList, having following additional functionalities:**

**void** [**display()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Should display the elements stored in linkedList.

* *Make a function to search and display the position(s) of a given element in the linkedList.*

**void search(Type &inputLinkedList)**

Example:

Linked List: 2 6 1 6 3 8 5 0

INPUT: Search element = n. Where ‘n’ can be any number, lets say n = 6.

OUTPUT:- Element ‘6’ is at position: 2, 4.

* *Make a function to insert a new node at the Nth position.*

**void insertAtNth(Type &inputLinkedList)**

Example:

Linked List: 2 3 5 7 9

Input: Insert new node at ‘3rd’ position.

Output LinkedList: 2 3 ‘n’ 5 7 9 (where ‘n’ represents new inserted node, containing any data)

1. **Now use the above class to make another derived class named as SinglyCircularLinkedList, having following additional functionalities:**

* *Make a function to insert a node at the head of linkedlist.*

**void insertAtHead(int);**

Example:

Linked List: 2 6 1 6 3

After inserting ‘8’ at the head.

Output LinkedList: 8 2 6 1 6 3

* *Make function to delete the head node of linkedlist.*

**void DeleteAtHead(int);**

Example:

Linked List: 2 6 1 6 3

After deleting data ‘2’ at the head.

Output LinkedList: 6 1 6 3

**Now create menu based program to perform the following operations**

1. Press 1 to add a new element to the tail of linkedlist. **void insertAtTail(int)**
2. Press 2 to delete the element from tail end of linkedlist. **void deleteAtTail()**
3. Press 3 to display the LinkedList.
4. Press 4 to search and display the position of a given element in Singly LinkedList.
5. Press 5 to insert new node at the Nth position in the Singly LinkedList.
6. Press 6 to insert new node at the head in Singly Circular LinkedList. **void insertAtHead(int)**
7. Press 7 to delete the head node in Singly Circular LinkedList. **void deleteAtHead(int)**
8. Press 0 to exit.

# Task 2

Create a C++ generic abstract class named as **LinkedList**, with the following:

**Attributes:**

1. Node \*head
2. Node \*tail

**Functions:**

virtual void insertAtTail(int) = 0;

* + Should add element at the tail node of linkedList.

virtual void deleteAtTail() = 0;

* + Should delete element at the tail node of linkedList.

LinkedList();

# Doubly + Doubly Circular LinkedList:

Doubly Linked List is type of data structure in which a node contains three fields i.e. data stored at that particular address, next pointer which contains the address of the next node and previous pointer which contains the address of the previous node in the memory. The last node’s head pointer and first node’s previous pointer contains pointer to the null.

Doubly Circular LinkedList node also contains three fields, data stored, next and previous pointer. While the next pointer of tail node contains address of head node, and previous pointer of head node contains the address of the tail node.

1. **Use the above class to make another class named as DoublyLinkedList, having following additional functionalities:**

**void** [**display()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Should display the elements stored in linkedList.

* *Make a function to swap head and tail nodes in the linkedlist.*

**void swapNodes(Type &inputLinkedList)**

Example:

Linked List: 2 ↔ 6 ↔ 1 ↔ 4 ↔ 3

After swapping: 3 ↔ 6 ↔ 1 ↔ 4 ↔ 2

1. **Use the above class to make another class named as DoublyCircularLinkedList, having following functionalities:**

* *Make a function to check if the list is palindrome or not.*

**void CheckPalindrome(Type &inputLinkedList)**

Example:

Input Linked List:  **2 5 3 6 3 5 2 OR 4 6 3 3 6 4**

Output: Palindrome

Input Linked List: **2 5 3 6 5 2 OR 4 4 9 5 8**

Output: Not a Palindrome

**Provide main function for above classes.**

**Now create menu based program to perform the following operations**

1. Press 1 to add a new element to the tail of linkedlist. **void insertAtTail(int)**
2. Press 2 to delete the element from tail end of linkedlist. **void deleteAtTail()**
3. Press 3 to display the LinkedList.
4. Press 4 to swap Head and Tail nodes in Doubly LinkedList.
5. Press 5 to check palindrome in Doubly Circular LinkedList.
6. Press 0 to exit.