University of Central Punjab

**Faculty of Information Technology**

**Data Structures and Algorithms Spring 2023**

|  |  |  |
| --- | --- | --- |
| **Lab 05** | |  |
| **Topic** | * LinkedList * Circular linked List * Linked lest using Stack |
| **Objective** | * The basic purpose of this lab is to implement ADT of Linked List 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 List with following: (You can use your previous code as well)

**Functions:**

**virtual void insertAtFront(Type) = 0; virtual void insertAtLast(Type) = 0; virtual void removeAtFront(Type) = 0; virtual void removeAtLast(Type) = 0;**

**virtual void insertAtSpecificPosition(Type,Type) = 0;** Adds the element of type Type at the head of the linkedlist. **virtual void insertNext(Type Nodevalue,Type ) = 0;**

Adds the element of type at next of the given node value only if next of the node is Null, otherwise the value will be added between the targeted node and next of targeted node

Exmaple: list is **head->1->5->8->9->NULL** new value is 10 and targeted node is 9 List will

**Head->1->5->8->9->10->** NULL. If the targeted not is 5 then list will be 1->5->10->8->9-

>NULL

## virtual void insertSorted(Type) = 0;

Adds the element of type at the right place in the lined list Example:

* New value is 10 then Head->10->NULL
* New Value is 2 then Head->2->10->NULL
* New Value is 6 then Head->2->6->10->NULL
* And so on

## virtual void sortLinkedList(Type) = 0;

Its sorts the linked list in ascending order.

# virtual Type removeFromPosition(Type) =0;

Removes and returns the first element of the linked list, and reduces size of the linked list by 1.

# virtual Type removeNext() =0;

Removes the element of type next to the given node value only if next of the node exists. If next node is NULL, it shows appropriate message and returns -1

# virtual void reverseListNode() =0;

Reverse the whole link list.

# Task 2

For the creation of node create a C++ template based class named as Node with the following:

**Attributes:**

* Type Data;
* Node <Type> \*Next;

## Note: class Node should be created in a separate file named Node.h Implement the constructor and destructors

Now create a C++ generic class named MyList by using the class list created in Task 1 with following:

**Attributes:**

* Node <Type> \*head;

**Functions:**

**bool isEmpty() --**Returns whether the list is empty(1) or not(0).

**int sizeOfList()** – Returns the number of elements in the list.

* Implement all the functions declared in class List
* Create appropriate constructor
* Implement the constructor which actually deletes the list node by node

**Constraint:**

**For this task there only Head pointer you are not allowed to use Tail pointer**

**Task 3**

Now create a C++ generic class named MyCircularList by using the class list created in Task 1 and Node class in Task 2 with following:

Implement All above function on TASK 1 and Task 2 in Circular Linked List.

**Attributes:**

* Node <Type> \*tail;

**Functions:**

**bool isEmpty() --**Returns whether the list is empty(1) or not(0).

**int sizeOfList()** – Returns the number of elements in the list.

* Implement all the functions declared in class List
* Create appropriate constructor
* Implement the constructor which actually deletes the list node by node

**Constraint:**

**For this task there only Tail pointer you are not allowed to use Head pointer**

**Task 4**

Given a singly linked list, check whether the given list is palindrome or not. If it palindrome then return true otherwise return false.

**Hint:** Do this task by using Stack

**Input**: C🡪 I 🡪 V 🡪 I 🡪 C

**Output**: Given list is Palindrome

**Input**: S🡪 O 🡪 O 🡪 N

**Output**: Given list is Palindrome

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***