**Quiz 1**

**Data Structures And Algorithms Lab**

**CSL-221**

|  |  |
| --- | --- |
| Name:  \_ABDULLAH\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Enrolment #:  \_\_\_\_\_\_\_\_\_01-134232-013\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Question:**

A stack can be efficiently implemented using a linked list, where the first node represents the "top" of the stack. The push operation adds a node at the head (top) of the linked list, while the pop operation removes the node at the head.

Design and implement a Stack class using a linked list. Your implementation should include the following standard stack operations:

* push(): Adds an element to the top of the stack.
* pop(): Removes and returns the element from the top of the stack.
* peek(): Returns the element at the top without removing it.
* isEmpty(): Checks if the stack is empty.
* size(): Returns the number of elements in the stack.

**Note:**  
When implementing the pop() operation, be especially careful when removing the head node. Ensure that you properly handle the case where the stack is empty to avoid accessing null pointers or creating memory leaks.

Write the necessary code for the class and implement all standard member functions.

**CODE:**

**Main**

#include <iostream>

#include "List.h"

using namespace std;

int main() {

List link

link.pop();

link.push();

link.isEmpty();

link.peak();

link.size();

return 0;

}

Node.h

#pragma once

#include<iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node();

~Node();

};

Node.cpp

#include "Node.h"

Node::Node()

{

data = 0;

next = NULL;

}

Node::~Node()

{

}

List.h

#pragma once

#include "Node.h";

using namespace std;

class List

{

public:

Node\* head;

List();

~List();

bool isEmpty();

void push(int);

void pop(int);

void peak(int);

void size();

};

List.cpp

#include "List.h"

List::List()

{

head = 0;

}

bool List::isEmpty() {

if (head = NULL) {

return true;

}

else {

return false;

}

}

List::~List()

{

}

void List::push(int data) {

Node\* temp = new Node;

temp->next = 0;

temp->data = data;

if (head == 0) {

head = temp;

}

else {

temp->next = head;

head = temp;

}

}

void List::pop(int val) {

if (head == NULL) {

cout << "list is empty";

}

Node\* ptr = head;

Node\* temp = NULL;

if (ptr->data == val) {

head = ptr->next;

delete ptr;

return;

}

while (ptr->data != val) {

head = NULL;

}

}

void List::peak(int data) {

Node\* temp = new Node;

temp->next = 0;

temp->data = data;

if (head == 0) {

head = temp;

}

else {

temp->next = head;

head = temp;

}

}

void List::size(int data) {

Node\* temp = new Node;

temp->next = 0;

temp->data = data;

cout << "size of list";

}