**NAME : MALIHA SHAHID**

**ROLL NO:SU92-BSSEM-S24-071**

**LAB 6**

#include <iostream>

using namespace std;

class ListNode {

public:

int data;

ListNode\* nextNode;

ListNode(int val) {

data = val;

nextNode = NULL;

}

};

class LinkedList {

public:

ListNode\* first;

LinkedList() {

first = NULL;

}

void addAtBeginning(int val) {

ListNode\* newNode = new ListNode(val);

newNode->nextNode = first;

first = newNode;

}

void removeFirst() {

if (first == NULL) {

cout << "List is already empty.\n";

return;

}

ListNode\* temp = first;

first = first->nextNode;

delete temp;

}

void removeLast() {

if (first == NULL) {

cout << "List is already empty.\n";

return;

}

if (first->nextNode == NULL) {

delete first;

first = NULL;

return;

}

ListNode\* temp = first;

while (temp->nextNode->nextNode != NULL) {

temp = temp->nextNode;

}

delete temp->nextNode;

temp->nextNode = NULL;

}

void removeNthNode(int n) {

if (first == NULL) {

cout << "List is empty.\n";

return;

}

if (n == 1) {

removeFirst();

return;

}

ListNode\* temp = first;

for (int i = 1; temp != NULL && i < n - 1; i++) {

temp = temp->nextNode;

}

if (temp == NULL || temp->nextNode == NULL) {

cout << "Invalid position!\n";

return;

}

ListNode\* nodeToDelete = temp->nextNode;

temp->nextNode = temp->nextNode->nextNode;

delete nodeToDelete;

}

void removeMiddle() {

if (first == NULL || first->nextNode == NULL) {

removeFirst();

return;

}

ListNode\* slow = first;

ListNode\* fast = first;

ListNode\* prev = NULL;

while (fast != NULL && fast->nextNode != NULL) {

prev = slow;

slow = slow->nextNode;

fast = fast->nextNode->nextNode;

}

prev->nextNode = slow->nextNode;

delete slow;

}

void printList() {

if (first == NULL) {

cout << "LIST IS EMPTY\n";

return;

}

ListNode\* temp = first;

while (temp != NULL) {

cout << temp->data << " -> ";

temp = temp->nextNode;

}

cout << "NULL\n";

}

};

int main() {

LinkedList myList;

myList.addAtBeginning(10);

myList.addAtBeginning(20);

myList.addAtBeginning(30);

myList.addAtBeginning(40);

myList.addAtBeginning(50);

cout << "Original List: ";

myList.printList();

myList.removeFirst();

cout << "After deleting first node: ";

myList.printList();

myList.removeLast();

cout << "After deleting last node: ";

myList.printList();

myList.removeNthNode(2);

cout << "After deleting 2nd node: ";

myList.printList();

myList.removeMiddle();

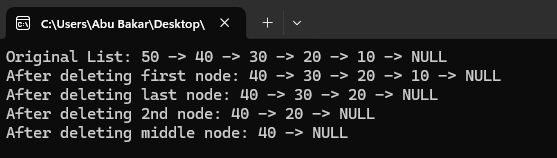
cout << "After deleting middle node: ";

myList.printList();

return 0;

}

**OUTPUT**

****

**EXPLANATION**

* ListNode Class – Represents a node with data and nextNode.
* LinkedList Class – Manages the list with first as the head.
* addAtBeginning – Inserts a node at the start.
* removeFirst / removeLast – Deletes the first or last node.
* removeNthNode(n) – Deletes the node at position n.
* removeMiddle – Removes the middle node using the two-pointer method.
* printList – Displays the linked list.
* main Function – Demonstrates insertions, deletions, and prints the updated list.