SUPERIOR UNIVERSITY

NAME :ZOHAIB KHAN

ROLL_ NO : 022

SUBJECT: DSA LAB

LAB NO 6

Task: Implement functions to delete the first node, last node, Nth node, and centre node of a singly linked list.

```cpp
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
class LinkedList {
private:
    Node* head;
public:
    LinkedList() : head(nullptr) {}
    void insertAtEnd(int value);
    void deleteFirstNode();
    void deleteLastNode();
    void deleteNthNode(int n);
    void deleteCentreNode();
    void display();
};
void LinkedList::insertAtEnd(int value) {
    Node* newNode = new Node{value, nullptr};
    if (!head) head = newNode;
    else {
        Node* temp = head;
        while (temp->next) temp = temp->next;
        temp->next = newNode;
    }
}
void LinkedList::deleteFirstNode() {
    if (!head) cout << "List is empty!" << endl;
    else {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
}
void LinkedList::deleteLastNode() {
    if (!head) cout << "List is empty!" << endl;
    else if (!head->next) {
        delete head;
        head = nullptr;
    } else {
        Node* temp = head;
        while (temp->next->next) temp = temp->next;
        delete temp->next;
        temp->next = nullptr;
    }
}
void LinkedList::deleteNthNode(int n) {
    if (n < 0 || !head) {
        cout << "Invalid position or list is empty!" << endl;
```

```
    }
    if (n == 0) {
        deleteFirstNode();
        return;
    }
    Node* temp = head;
    for (int i = 0; i < n - 1 && temp; i++) temp = temp->next;
    if (!temp || !temp->next) {
        cout << "Position out of range!" << endl;
        return;
    }
    Node* nodeToDelete = temp->next;
    temp->next = temp->next->next;
    delete nodeToDelete;
}
void LinkedList::deleteCentreNode() {
    if (!head) {
        cout << "List is empty!" << endl;
        return;
    }
    Node *slow = head, *fast = head, *prev = nullptr;
    while (fast && fast->next) {
        prev = slow;
        slow = slow->next;
        fast = fast->next->next;
    }
    if (!prev) head = head->next;
    else prev->next = slow->next;
    delete slow;
}
void LinkedList::display() {
    Node* temp = head;
    while (temp) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "nullptr" << endl;
}
int main() {
    LinkedList list;
    list.insertAtEnd(10); list.insertAtEnd(20); list.insertAtEnd(30);
    list.insertAtEnd(40); list.insertAtEnd(50); list.display();
    list.deleteFirstNode();  list.display();
    list.deleteLastNode();   list.display();
    list.deleteNthNode(1);   list.display();
    list.deleteCentreNode(); list.display();
    return 0;
}
```

```
10 -> 20 -> 30 -> 40 -> 50 -> nullptr
20 -> 30 -> 40 -> 50 -> nullptr
20 -> 30 -> 40 -> nullptr
20 -> 40 -> nullptr
20 -> nullptr

--------------------------------
Process exited after 0.6125 seconds with return value 0
Press any key to continue . . .
```

1. insertAtEnd(int value): Inserts a node at the end of the linked list.
2. deleteFirstNode(): Deletes the first node of the linked list.
3. deleteLastNode(): Deletes the last node of the linked list.
4. deleteNthNode(int n): Deletes the node at the specified position n.