A purple text on a black background

AI-generated content may be incorrect.

**LAB 6 - Singly Linked List (Delete Nodes)**

Name: Omar Afef Mohammed

Roll number: SU92-BSSEM-S24-086

Section: 3A

Teacher: Mr. Rasikh Ali

Subject: DSA LAB

Date: February 11, 2025

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

**ANSWER**

**CODE**

#include <iostream>

using namespace std;

class Node{

public:

int data;

Node\* next;

Node(int data){

this->data=data;

next=NULL;

}

};

class LinkedList{

public:

Node\* head;

LinkedList(){

head=NULL;

}

void insertAtStart(int d){

Node\* newNode=new Node(d);

newNode->next=this->head;

this->head=newNode;

}

//at start

void DeleteAtFirst(){

if(head!=NULL){

Node\* temp = head;

head = head->next;

delete temp;

}

}

// at end

void DeleteAtEnd(){

if(head!=NULL){

Node\* temp=head;

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

temp=temp->next;

}

delete temp->next;

temp->next=NULL;

return ;

}

}

//display

void display(){

Node\* tem=head;

while(tem!=NULL){

cout<<tem->data<<"->";

tem=tem->next;

}

cout<<"Null\n\n";

}

// at specifc number:

void DeleteAt(int num){

if (head == NULL) {

return ;

}

Node\* temp = head;

if (num == 1) { // Fix: Use == for comparison

head = head->next;

delete temp;

return ;

}

for(int i=1; i<num-1;i++){

temp=temp->next;

if (temp == NULL) { // If num is too big

cout << "big number" << endl;

return ;

}

}

Node\* prev = temp->next;

temp->next = temp->next->next;

delete prev;

return ;

}

// at center

void DeleteAtCenter(){

Node\* tem=head;

int num=0;

while(tem->next!=NULL){

num++;

tem=tem->next;

}

tem=head;

for(int i=0;i<(num/2)-1;i++){

tem=tem->next;

}

Node\* tem2=tem->next;

tem->next=tem->next->next;

delete tem2;

tem2=NULL;

}

};

int main(){

LinkedList link;

link.insertAtStart(10);

link.insertAtStart(20);

link.insertAtStart(30);

link.insertAtStart(40);

link.insertAtStart(50);

link.insertAtStart(60);

cout << "Linked List Contents before:" << endl;

link.display();

cout << "Linked List Contents after delete at 2th:" << endl;

link.DeleteAt(2);

link.display();

cout << "Linked List Contents after delete at center:" << endl;

link.DeleteAtCenter();

link.display();

cout << "Linked List Contents after delete at end:" << endl;

link.DeleteAtEnd();

link.display();

cout << "Linked List Contents after delet at start:" << endl;

link.DeleteAtFirst();

link.display();

}

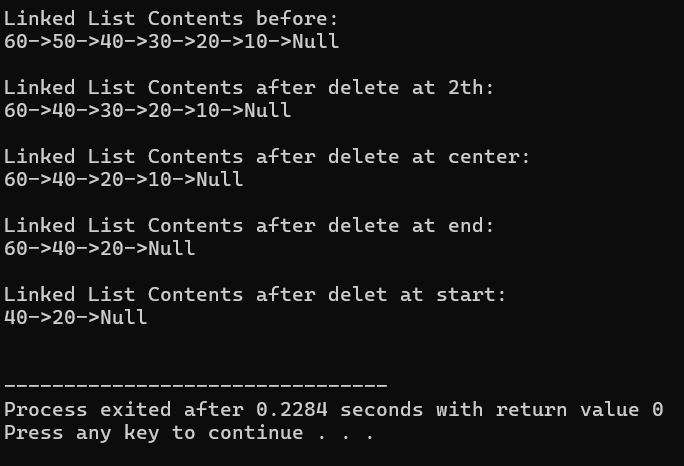
**How it works:**

1. **First Node: Print the head node.**
2. **Last Node: Traverse until next == NULL and delete the last node.**
3. **Nth Node: Traverse n-1 times, checking if n is out of range.**
4. **Center Node:**
   * **Find the total number of nodes.**
   * **Divide by 2 to get the middle index.**
   * **Traverse again to that position.**

**Why it works:**

* **O(1) for the first node, since we directly access head.**
* **O(n) for the last, Nth, and center nodes, since we traverse the list**

**OUTPUT**

****