A purple text on a black background

AI-generated content may be incorrect.

**LAB 7 - Doubly Linked List (Insert & Display Nodes)**

Name: Omar Afef Mohammed

Roll number: SU92-BSSEM-S24-086

Section: 3A

Teacher: Mr. Rasikh Ali

Subject: DSA LAB

Date: February 19, 2025

Task: Implement functions to insert node at first, last, Nth location, and centre of a doubly linked list. And display in order and display in reverse order.

**ANSWER**

**CODE**

#include <iostream>

using namespace std;

class Node{

public:

int data;

Node\* next;

Node\* prev;

Node(int data){

this->data=data;

next=NULL;

prev=NULL;

}

};

class LinkedList{

public:

Node\* head;

LinkedList(){

head=NULL;

}

//display

void display(){

Node\* temp=head;

while(temp!=NULL){

if(temp->prev==NULL){

cout<<"NULL | ";

}

else{

cout<<temp->prev<<" | ";

}

cout<<temp->data;

if(temp->next==NULL){

cout<<" | NULL";

}

else{

cout<<" | "<<temp->next;

}

if(temp->next!=NULL){

cout<<" -> ";

}

temp=temp->next;

}

cout<<endl;

}

// display reverse

void DisplayReverse() {

Node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

while (temp != NULL) {

if(temp->next==NULL){

cout<<"NULL | ";

}

else{

cout<<temp->next<<" | ";

}

cout<<temp->data;

if(temp->prev==NULL){

cout<<" | NULL";

}

else{

cout<<" | "<<temp->prev;

}

if(temp->prev!=NULL){

cout<<" -> ";

}

temp = temp->prev;

}

cout << endl;

}

//insert at start

void InsertAtStart(int d){

Node\* newNode= new Node(d);

if(head==NULL){

head=newNode;

return;

}

newNode->next=head;

head->prev=newNode;

head=newNode;

}

//insert at end

void InsertAtEnd(int d){

Node\* newNode= new Node(d);

if(head==NULL){

head=newNode;

return;

}

Node\* temp=head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=newNode;

newNode->prev=temp;

}

//insert at any

void InsertAt(int d, int num){

Node\* newNode= new Node(d);

if(head==NULL){

if(num==1){

head=newNode;

}

else{

cout<<"Invaild Number/n";

delete newNode;

newNode=NULL;

}

return;

}

if(num==1){

newNode->next=head;

head->prev=newNode;

head=newNode;

return;

}

Node\* temp=head;

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

temp=temp->next;

if(temp=NULL){

cout<<"Invaild Number\n";

return;

}

}

newNode->prev=temp;

newNode->next=temp->next;

temp->next->prev=newNode;

temp->next=newNode;

}

//insert at center

void InsertAtCenter(int d){

Node\* newNode=new Node(d);

if(head==NULL){

head=newNode;

return;

}

int num=0;

Node\* temp=head;

while(temp->next!=NULL){

num++;

temp=temp->next;

}

num=num/2;

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

temp=temp->prev;

}

newNode->prev=temp;

newNode->next=temp->next;

temp->next->prev=newNode;

temp->next=newNode;

return;

}

};

int main(){

LinkedList list;

cout<<"After insert at start: \n";

list.InsertAtStart(1);

list.display();

Node\* head1=list.head;

cout<<"\nAfter insert at end: \n";

list.InsertAtEnd(4);

list.display();

cout<<"\nAfter insert at 2th: \n";

list.InsertAt(2,2);

list.display();

cout<<"\nAfter insert at center: \n";

list.InsertAtCenter(3);

list.display();

cout<<"\ndisplay in order: \n";

list.display();

cout<<"\ndisplay in revers order: \n";

list.DisplayReverse();

}

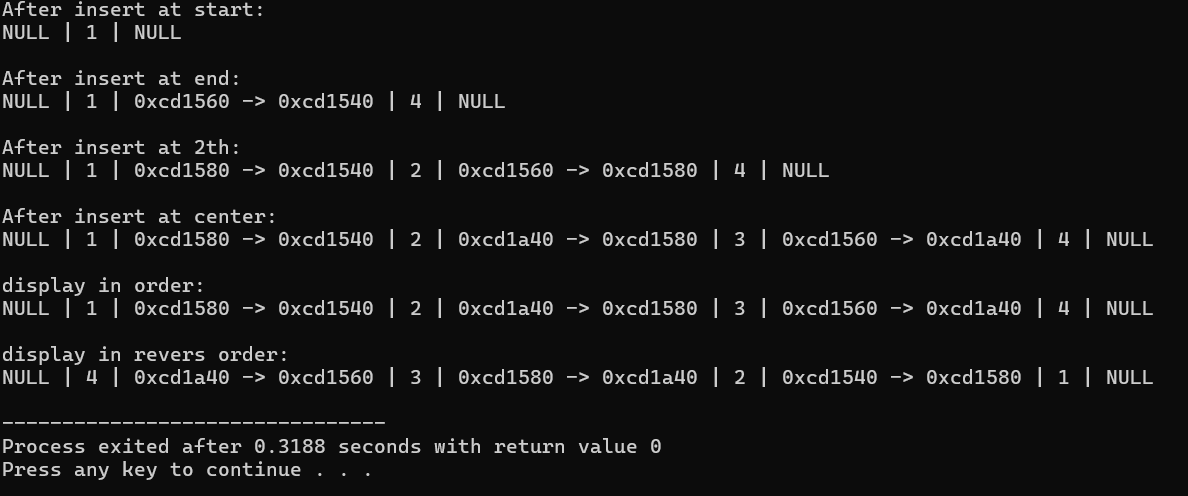
**How it works:**

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 backward to that position.**
5. **Display reverse:**
   * **Traverse until next == NULL.**
   * **Cout the node.**
   * **Traverse backward until node== NULL.**

**Why it works:**

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

**OUTPUT**

****