

Lab #7:

Name: Hassan Farooq Siddiqi

Roll No: BSSEM-S24-099

Section: BSSE-3A

Subject: Data Structure & Algorithm

**Code:**

#include<iostream>

using namespace std;

class Node{

    public:

    int data;

    Node\* prev;

    Node\* next;

    public:

    Node(int d = 0){

        data = d;

        prev = nullptr;

        next = nullptr;

    }

};

class DublyList{

    Node\* head;

    Node\* tail;

    public:

    DublyList(){

        head = nullptr;

    }

    void insertAtFirst(int d){

        Node\* newNode = new Node(d);

        if(head == nullptr){

            head = newNode;

        }

        else {

            newNode->next = head;   //head is not empty it have some values so head have value so we store it in the next of newNode

            head->prev = newNode;

            head = newNode;

        }

    }

    void insertAtLast(int d){

        Node\* newNode =  new Node(d);

        if(head == nullptr){

            head = newNode;

        }else{

            Node\* temp = head;

            while(temp->next != nullptr){

                temp=temp->next;

            }

            newNode->prev = temp; //storing the address of lastNode in previous of newNode

            temp->next = newNode; //Now linked  the next of last Node to NewNode

        }

    }

    void insertAtCenter(int d){

        Node\* newNode = new Node(d);

        if(head == nullptr){

            head = newNode;

        }

        else {

            Node\* temp = head; //For traveresing & counting

            int count = 1;

            while (temp->next != nullptr)

            {

                temp = temp->next;

                count++;

            }

            temp = head;

            for(int i  =1 ; i<=count/2;i++){

                temp = temp->next;

            }

            newNode->next= temp->next ;

            newNode->prev =temp;

            temp->next->prev = newNode;

            temp->next =newNode;

        }

    }

    void insertAtSpecficPosistion(int d ,int pos){

        Node\* newNode =  new Node(d);

        if(head == nullptr){

            head = newNode;

        }else if(pos < 1){

            cout<<"Invalid Position\n";

        }else if(pos == 1){

        newNode->next = head;

        head->prev = newNode;

        head = newNode;

        }

        else {

            Node\* temp = head;

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

                temp = temp->next;

                if(temp == nullptr){

                    break;

                }

            }

            newNode->prev = temp;  // Prevoius of newNode is temp

            newNode->next = temp->next;   //Linked the next of newNode is temp of next

            temp->next->prev = newNode;   // now linked the Prevous of temp-> next is newNode address

            temp->next = newNode;        ///now linked the  neNode to temp of next

        }

    }

    void display\_In\_Order(){

        Node\* temp = head;

        cout<<"Doubly List : ";

        while(temp != nullptr){

            cout<<temp->data<<" ";

            temp = temp->next;

            if(temp!= nullptr){

               cout<< " , ";

            }

        }

        cout<<endl;

    }

    void displayInreverse(){

        Node\* temp = head;

        cout<<"Revered Doubly List : ";

        while(temp->next != nullptr){

            temp = temp->next;

        }

        while(temp != nullptr){

            cout<<temp->data<<" ";

            if(temp->prev != nullptr){

                cout<<" , ";

            }

            temp = temp->prev;

        }

        cout<<endl;

    }

};

int main(){

    DublyList d;

    d.insertAtFirst(23);

    d.insertAtLast(40);

    d.insertAtLast(90);

    d.insertAtLast(55);

    d.insertAtSpecficPosistion(37,2);

    d.display\_In\_Order();

    d.displayInreverse();

    d.insertAtCenter(8);

    cout<<"-----------------------------------------.\n";

    cout<<"After Adding the Center Node.\n";

    d.display\_In\_Order();

    d.displayInreverse();

    return 0;

}

**Output:**

