

Lab #6:

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 \*next;

    Node(){

        data = 0;

        next = nullptr;

    }

    Node(int d){

        this->data = d;

        this->next = nullptr;

    }

};

class LinkedList{

    Node \*head;

    public:

    LinkedList(){

        this->head = nullptr;

    }

    //insertation faciltators

    void insertAtStart(int d){

        Node \*newNode = new Node(d);

       newNode->next = head;

       head =newNode;

    }

    void insertAtLast(int d){

        Node \*newNode = new Node(d);

        Node \* temp = head;

        while(temp->next!= nullptr){

            temp = temp->next;

        }

        temp->next = newNode;

    }

    void insertAtAnyPosition(int d, int po){

        if(po < 1){

            cout<<"Invalid Positoin\n";

        }

        else if( po == 1){

            insertAtStart(d);

        }

        else{

            Node \*newNode = new Node(d);

            Node \*temp = head;

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

                if(temp == nullptr){

                    cout<<"Invalid Postion";

                    break;

                }

                temp = temp->next;

            }

          newNode->next = temp->next;

            temp->next = newNode;

        }

    }

    //Deletion facalitators

    void delete\_at\_Start(){

        if(head!= nullptr){

            Node\* temp = head;

            head = head->next;

            delete temp;

            temp = nullptr;

        }

    }

    void delete\_at\_position(int pos){

        if(head!= nullptr){

            if(pos<1){

                cout<<"Invalid Position\n";

            }

            else if(pos == 1){

                delete\_at\_Start();

            }

            else {

                Node\* temp = head;

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

                temp = temp->next;

                if(temp == nullptr){

                    break;

                }

            }

            Node\* temp2 = temp->next->next;

            delete temp->next;

            temp->next = temp2;

            }

       }

    }

    void delete\_lastNode(){

        if(head!= nullptr){

            Node\* temp = head;

            Node\* temp2 = nullptr;

            while(temp->next!= nullptr){

                temp2 = temp;

                temp = temp->next;

            }

            temp2->next = nullptr;

            delete temp;

            temp = nullptr;

        }

    }

    void delete\_Center\_Node(){

        if(head!= nullptr){

            Node\* temp = head;

            int count = 1;

            while (temp!= nullptr)

            {

                count++;

                temp = temp->next;

            }

            temp = head;

            Node\* temp2 = nullptr;

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

                temp2 = temp;

                temp= temp->next;

            }

            temp2->next = temp->next;

            delete temp;

        }

    }

    void delete\_a\_value (int v){

        if(head!= nullptr){

            Node\* temp = head;

            Node\* temp2 = nullptr;

            while(temp != nullptr && temp->data != v){

                temp2 =temp;

                temp = temp->next;

            }

            temp2->next = temp->next;

            delete temp;

        }

    }

    //Display facalitators

    void displayLinknedList(){

        Node \*temp = head;

        while(temp != nullptr){

            cout<<temp->data;

            if(temp->next != nullptr){

                cout<<",";

            }

            temp = temp->next;

        }

        cout<<endl;

    }

    void displayFirstNode(){

      cout<<head->data<<endl;

    }

    void displaylastNode(){

      Node \*temp = head;

      while(temp->next != nullptr){

        temp = temp->next;

      }

      cout<<temp->data<<endl;

    }

    void displayCenterNode(){

      Node \*temp = head;

      int count = 0;

      while(temp != nullptr){

        temp= temp->next;

        count++;

      }

       temp = head;

      for(int i = 1 ; i<count; i++){

        temp = temp->next;

        if(i == count/2){

          cout<<temp->data<<endl;

          break;

        }

      }

    }

};

int main(){

    LinkedList list;

    list.insertAtStart(9);

    list.insertAtStart(93);

    list.insertAtLast(0);

    list.insertAtLast(88);

    list.insertAtAnyPosition(8,2);

    list.insertAtAnyPosition(77,3);

    list.insertAtAnyPosition(18,2);

    cout<<"Before LinkedList : ";

    list.displayLinknedList();

    cout<<"First Linked List Node : ";

    list.displayFirstNode();

    cout<<"Last LinkedList Node : ";

    list.displaylastNode();

    cout<<"Center linkedList : ";

    list.displayCenterNode();

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

    cout<<"Deleteing the Last Node.\n";

    list.delete\_lastNode();

    cout<<"Last Linked Node :";

    list.displaylastNode();

    cout<<"Deleteing the First Node.\n";

    list.delete\_at\_Start();

    cout<<"First Linked List Node : ";

    list.displayFirstNode();

    cout<<"Deleteing the Center Node.\n";

    list.delete\_Center\_Node();

    cout<<" Center linkedList : ";

    list.displayCenterNode();

    cout<<"LinkedList : ";

    list.displayLinknedList();

    return 0;

}

**Output:**

