

Lab #9:

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 , \* prev ;

    Node(int d =0 ){

        data = d;

        next = prev = nullptr;

    }

};

class CircularList{

    Node\* head , \* tail ;

    public:

    CircularList(){

        head = tail = nullptr;

    }

    void insertAtStart(int d){

        Node\* newNode = new Node( d);

        if(head == nullptr && tail == nullptr){

            head = tail = newNode;

            tail->next = head;

            head->prev = tail;

        }else {

            newNode->next = head;

            newNode->prev = tail;

            head = newNode;

            newNode->next->prev = newNode;

            tail->next = head;

            head->prev = tail;

        }

    }

    void insertAtEnd(int d){

        Node\* newNode = new Node(d);

        if(head== nullptr){

            head = tail = newNode;

            head->prev = tail;

            tail->next = head;

        }else {

            tail->next = newNode;

            newNode->prev = tail;

            tail = newNode;

            tail->next = head;

            head->prev = tail;

        }

    }

    void insertAtCenter(int d){

        Node\* newNode = new Node(d);

        if(head!= nullptr){

            Node\* temp = head;

            int count = 1;

            while (temp->next != head)

            {

                count++;

                temp = temp->next;

            }

            temp = head;

            if(count %2 != 0){ //odd case

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

                {

                    temp = temp->next;

                }

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

                newNode->prev = temp->next;

                temp->next->next->prev = newNode;

                temp->next->next = newNode;

            }else { //even case

                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;

            }

        }

else {

            head = tail = newNode;

            head->prev = tail;

            tail->next = head;

        }

    }

    void display(){

        Node\* temp = head;

        if(head!= nullptr){

           do

           {

            cout<<" "<<temp->data<<" ";

            if(temp!= tail){

                cout<<" <> ";

            }

            temp = temp->next;

           } while (temp != head);

        }

        cout<<endl;

    }

    void displayReverse(){

        Node\* temp = head;

        if(head!= nullptr){

           do

           {

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

            if(temp->prev != head){

                cout<<" <> ";

            }

            temp = temp->prev;

           } while (temp != head);

            delete temp;

            temp = nullptr;

      }

        cout<<endl;

    }

    void insertAtPosition(int d, int pos){

        if (pos < 1)

        {

            cout<<"Invalid Position.\n";

        }

        else if(pos == 1 ){

            insertAtStart(d);

        }

        else {

            Node\* newNode = new Node(d);

            Node\* temp = head;

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

            {

                temp = temp->next;

                if(temp == head){

                    cout<<"Invalid Position.\n";

                    return ;

                }

            }

                newNode->next = temp;

                newNode->prev = temp->prev;

                temp->prev->next = newNode;

                temp->prev = newNode;

        }

    }

};

int main(){

    CircularList c;

    c.insertAtStart(9);

    c.insertAtStart(89);

    c.insertAtStart(99);

    c.insertAtStart(1);

    c.insertAtEnd(100);

    cout<<"\nNormal Display.\n";

    c.display();

    c.insertAtCenter(77);

    cout<<"\nAfter Adding 77 in Centre.\n";

    c.display();

    cout<<"\nAfter Adding 88 at position 3.\n";

    c.insertAtPosition(88,3);

    c.display();

    cout<<"\nAfter Revrersing .\n";

    c.displayReverse();

    return 0;

}

**Output:**

