Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

Name: Tanzeela Asghar

Reg no: 2021 BSE 032

Section no: III-A

Course: Data Structure

Submitted to: Sir Rehan Ahmed Siddiqui

Lab#11

Code Task#1

Implement the (class) Circular Linked List to create a list of integers. You need to provide the implementation of the member functions as described in the following. Use DOUBLY LINKED LIST

CODE:

```
#include "stdafx.h"
#include<iostream>
using namespace std;
class CList
{
	struct node
	{
		int data;
		node *next;
		node *prev;
	}*head;
public:
	CList()
	{
		head=NULL;
	}
	bool emptylist()
	{
		if(head==NULL)
			return true;
		else
			return false;
	}
	void insert (int pos, int value)
	{
		node *p,*q;
		p=new node;
		q=head;
		p->data=value;
		int count=1;
		if(pos==0)
		{
			p->next=head;
			p->prev=head;
			head=p;
		}
		else
		{
			while(count!=pos)
			{
```

```
                count++;
                q=q->next;
            }
            p->next=q->next;
            p->prev=q;
            q->next->prev=p;
            q->next=p;
        }

    }
    void insert_begin(int value)
    {
        node *p;
        p=new node;
        p->data=value;
        if(emptylist())
        {
            head=p;
            head->next=head;
            head->prev=head;
        }
        else if(head->next==head)
        {
            p->next=head;
            p->prev=head;
            head->prev=p;
            head->next=p;
            head=p;
        }
        else
        {
            p->next=head;
            p->prev=head->prev;
            head->prev->next=p;
            head->prev=p;
            head=p;
        }
    }
    void insert_end(int value)
    {
        node *p;
        p=new node;
        p->data=value;
        if(emptylist())
        {
            head=p;
```

```
                head->next=head;
                head->prev=head;
        }
        else if(head->next==head)
        {
                p->prev=head;
                p->next=head;
                head->next=p;
                head->prev=p;
        }
        else
        {
                p->next=head;
                p->prev=head->prev;
                head->prev->next=p;
                head->prev=p;
        }
}
void delete_begin()
{
        node *p;
        p=head;
        if(emptylist())
        {
                cout<<"List is empty...\n";
        }
        else
        {
                head->next->prev=head->prev;
                head->prev->next=head->next;
                p=head->next;
                delete head;
                head=p;
        }
}
void delete_end()
{
        node *p;
        p=head;
        if(emptylist())
        {
                cout<<"List is empty...\n";
        }
        else
        {
                p=head->prev;
```

```
                        head->prev->prev->next=head;
                        head->prev=head->prev->prev;
                        delete p;
                }
        }
        void traverse()
        {
                node *p;
                p=head;

                if(emptylist())
                        cout<<"List is empty...\n";
                else
                {
                        while(p->next!=head)
                        {
                                cout<<p->data<<" ";
                                p=p->next;
                        }
                        if(p->next==head)
                        cout<<p->data<<endl;
                }
        }
        void traverse2()
        {
                node *p;
                p=head->prev;
                do
                {
                        cout<<p->data<<" ";
                        p=p->prev;
                }while(p!=head->prev);
                cout<<"\n";
        }
};

int _tmain(int argc, _TCHAR* argv[])
{
        CList cl1;
        cout<<"Insert at begin:\n";
        cl1.insert_begin(7);
        cl1.insert_begin(8);
        cl1.insert_begin(9);
        cl1.insert_begin(10);
        cl1.traverse();
        cout<<"Inserting after specific position\n";
```

```
        cl1.insert(2,90);
        cl1.traverse();
        cout<<"Insert at end: \n";
        cl1.insert_end(0);
        cl1.traverse();
        cout<<"Deleting first node\n";
        cl1.delete_begin();
        cl1.traverse();
        cout<<"Deleting last node\n";
        cl1.delete_end();
        cl1.traverse();
        cout<<"Printing list in reverse order: \n";
        cl1.traverse2();
        system("pause");
        return 0;
}
```

OUTPUT:

D:\codes\lab 11\Debug\lab 11.exe

```
Insert at begin:
10 9 8 7
Inserting after specific position
10 9 90 8 7
Insert at end:
10 9 90 8 7 0
Deleting first node
9 90 8 7 0
Deleting last node
9 90 8 7
Printing list in reverse order:
7 8 90 9
Press any key to continue . . .
```