LAB TASK

20p-0480

Q1:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_?

Code:

#include<iostream>

using namespace std;

//class declaration

class DLL{

    class node

    {

        public:

        node\*next;

        node\*prev;

        int data;

        node(int val)

        {

            data=val;

            next=NULL;

            prev=NULL;

        }

    };

    private:

    node\*head;

    int length;

    public:

    DLL()

    {

        head=NULL;

        length=0;

    }

    int size()

    {

        return length;

    }

    void setsize(int l)

    {

        this->length=l;

    }

    node\*gethead()

    {

        node\*curr=head;

        return curr;

    }

    //using functions outside class

    void insert(int val,int pos);

    void remove(int pos);

    void print();

    void reverse();

    void splitlist();

};

int main()

{

     DLL li;

     cout<<"INSERTION"<<endl;

    li.insert(100,1);

    li.insert(1,2);

    li.insert(2,3);

    li.insert(3,4);

    li.insert(4,5);

    cout<<"PRINTING"<<endl;

    li.print();

    cout<<"DELETION"<<endl;

    li.remove(4);

    cout<<endl;

    li.print();

    cout<<"REVERSE"<<endl;

    li.reverse();

    cout<<endl;

    li.print();

    cout<<"SPLITLIST"<<endl;

    li.splitlist();

//use switch for menue

    // int menue\_option;

    // do

    // {

    //     cout<<"Enter the option number.Press 0 to exit."<<endl;

    //     cout<<"1)Insert()"<<endl;

    //     cout<<"2)remove()"<<endl;

    //     cout<<"3)print()"<<endl;

    //     cout<<"4)reverse()"<<endl;

    //     cout<<"5)splitlist()"<<endl;

    //     cin>>menue\_option;

    //     switch (menue\_option)

    //     {

    //     case 1:

    //         int val;

    //         int pos;

    //         cout<<"Enter the value :";

    //             cin>>val;

    //         cout<<"Enter position:"<<endl;

    //         cin>>pos;

    //             li.insert(val,pos);

    //             cout<<endl;

    //         break;

    //     case 2:

    //         cout<<"Enter the positoin :";

    //             cin>>pos;

    //             li.remove(pos);

    //             cout<<endl;

    //         break;

    //     case 3:

    //             li.print();

    //             cout<<endl;

    //         break;

    //     case 4:

    //             li.reverse();

    //             cout<<endl;

    //         break;

    //     case 5:

    //             li.splitlist();

    //             cout<<endl;

    //         break;

    //     default:

    //             cout<<"enter correct option"<<endl;

    //         break;

    //     }

    // } while (menue\_option !=0);

    return 0;

}

//INSERTION

void DLL:: insert(int val, int pos)

{

    if(pos>length+1 || pos< 1)

    {

        cout<<"Invalid position."<<endl;

        return;

    }

    node \*n=new node(val);

    if(pos==1)

    {

        n->next=head;

        if(head!=NULL)

           head->prev=n;

        head=n;

    }

    else

    {

        node \*curr=head;

        for(int i=1; i<(pos-1); i++)

        {

            curr=curr->next;

        }

        n->next=curr->next;

        n->prev=curr;

        if(curr->next!=NULL)

            curr->next->prev=n;

        curr->next=n;

    }

    length++;

}

//DELETION

void DLL::remove(int pos)

    {

        node\*curr=head;

        if(head == NULL)

        {

            cout<<"list is empty"<<endl;

        }

        if(pos==1)

        {

            head=curr->next;

            head->prev=NULL;

            delete curr;

            curr=NULL;

            length--;

        }

        else

        {

            node\*temp=NULL;

            for(int i=1;i<pos;i++)

            {

                temp=curr;

                curr=curr->next;

            }

            temp->next=curr->next;

            if(curr->next!=NULL)

            {

                curr->next->prev=temp;

                delete curr;

                curr=NULL;

                length--;

            }

        }

    }

//PRINTING

    void DLL:: print()

    {

        node\*curr=head;

        while(curr->next!=NULL)

        {

            cout<<curr->data<<"\t"<<endl;

            curr=curr->next;

        }

        cout << curr->data<<"\t"<<endl;

    }

    //REVERSE

    void DLL::reverse()

    {

        node\*curr=head;

        while (curr->next!=NULL)

        {

            curr=curr->next;

        }

        node\*r=head;

        int stop=length/2;

        for(int i=1;i<stop;i++)

        {

            int temp=curr->data;

            curr->data=r->data;

            r->data=temp;

            curr=curr->prev;

            r=r->next;

        }

    }

    //SPLITLIST

    void DLL::splitlist()

{

    if(length==1)

    {

        cout<<"NOt possible to split insert element"<<endl;

        return;

    }

    int size=0;

    node\*curr=head;

    //checking odd and even

    if(length%2==0)

    {

        size=length/2;

    }

    else

    {

        size=(length/2)+1;

    }

    for(int i=1;i<size;i++)

    {

        curr=curr->next;

    }

    node\*newlist=curr->next;

    newlist->prev=NULL;

    curr->next=NULL;

    DLL s;

    s.head=newlist;

    //print

    cout<<"After split."<<endl;

    cout<<"List 1 : "<<endl;

    print();

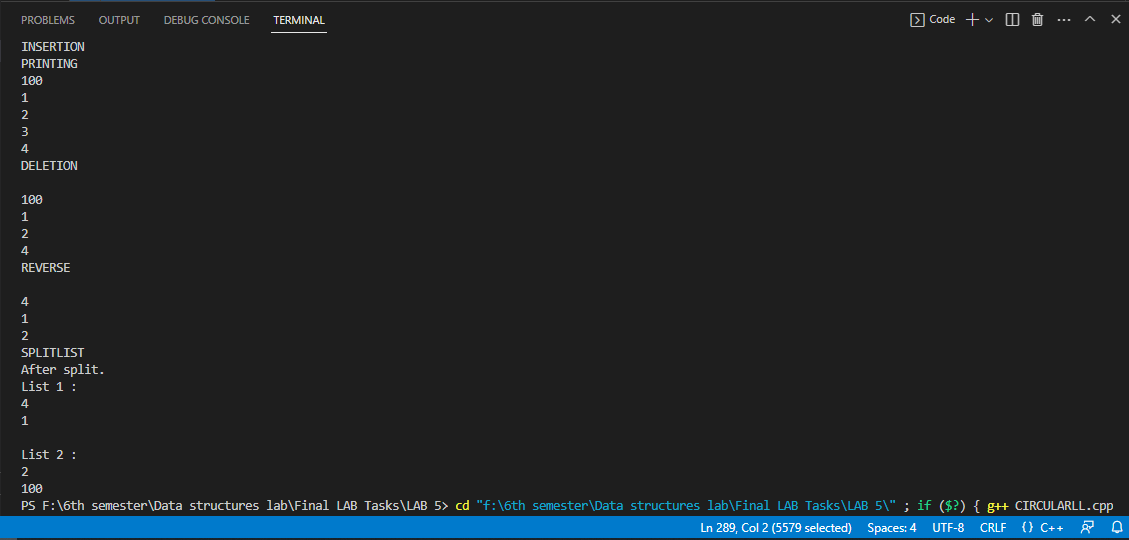
    cout<<endl;

    cout<<"List 2 : "<<endl;

    s.print();

}

Output:



Q2:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_?

Code:

#include<iostream>

using namespace std;

class node

{

    public:

    int data;

    node\*next;

    node(int val)

    {

        data=val;

        next=NULL;

    }

};

class CIRCULARLL

{

    private:

    int length;

    node\*head;

    public:

    CIRCULARLL()

    {

        length=0;

        head=NULL;

    }

    void insertAtbegin(int val);

    void deleteAtEnd();

    void printing();

};

void CIRCULARLL::insertAtbegin(int val)

{

    if(head == NULL)

    {

        node\*t=new node(val);

        head=t;

        t->next=head;

        length++;

    }

    else

    {

        node\*t=new node(val);

        node\*curr=head;

        while(curr->next!=head)

        {

            curr=curr->next;

        }

        t->next =head;

        head=t;

        curr->next=head;

        length++;

    }

}

void CIRCULARLL::deleteAtEnd()

{

    node\*curr=head;

    while(curr->next->next!=head)

    {

        curr=curr->next;

    }

    node\*t=curr->next;

    curr->next=head;

    delete t;

}

void CIRCULARLL::printing()

{

    if(length==0)

    {

        cout<<"LINKLIST is empty"<<endl;

    }

    node\*curr=head;

    while(curr->next!=head)

    {

        if(curr==head)

        {

            cout<<" ";

        }

        cout<<curr->data<<endl;

        curr=curr->next;

    }

    if(curr->next==head)

    {

        cout<<curr->data<<endl;

    }

}

int main()

{

    CIRCULARLL obj;

    int menue\_option;

    do

    {

        cout<<"Enter the option number.Press 0 to exit."<<endl;

        cout<<"1)InsertAtBegin()"<<endl;

        cout<<"2)deleteAtEnd()"<<endl;

        cout<<"3)printing()"<<endl;

        cin>>menue\_option;

        switch (menue\_option)

        {

        case 1:

            int val;

            cout<<"Enter the value :";

                cin>>val;

                obj.insertAtbegin(val);

                cout<<endl;

            break;

        case 2:

                obj.deleteAtEnd();

                cout<<endl;

            break;

        case 3:

                obj.printing();

                cout<<endl;

            break;

        default:

                cout<<"enter correct option"<<endl;

            break;

        }

    } while (menue\_option !=01);

    return 0;

}

Output:

