**Assignment NO:-1.5.2**

**Assignment Name:- Implementation of Program based on Circular Linked List**

**Name: Rutik Vasanta Kharche**

**Roll No : 112**

#include<iostream.h>

#include<conio.h>

class NODE

{

    public:

        NODE \*next;

        int data;

};

class LIST\_31

{

    private:

        NODE \*start,\*end;

    public:

        LIST\_31();

        void ADD\_FIRST(int);

        void ADD\_END(int);

        int DELETE\_FIRST();

        int DELETE\_END();

        //void ADD\_POS(int,int);

        //int DELETE\_POS(int);

        void LISTALL();

};

LIST\_31::LIST\_31()

{

    start=end=NULL;

}

void LIST\_31::ADD\_FIRST(int ele)

{

    //create New Node

    NODE \*NEW =new NODE();

    //Populate New Node

    NEW->data=ele;

    NEW->next=NULL;

    //set the Links

    if(start==NULL)

    {

    start=end=NEW;

    end->next=start;

    }

    else

    {

      NEW->next=start;

      start=NEW;

      end->next=start;

    }

}

void LIST\_31::ADD\_END(int ele)

{

    //create New Node

    NODE \*NEW = new NODE();

    //Populate New node

    NEW->data=ele;

    NEW->next=NULL;

    //Set the Links

    if (start==NULL)

    {

        start=end=NEW;

        end->next=start;

    }

    else

    {

        end->next=NEW;

        end=NEW;

        end->next=start;

    }

}

int LIST\_31::DELETE\_FIRST()

{

    NODE \*TEMP;

    int ele;

    if(start==NULL)

    {

        cout<<"List is empty"<<endl;

        return NULL;

    }

    if(start==end)

    {

         ele=start->data;

         TEMP=start;

         start=end=NULL;

    }

    else

    {

         ele=start->data;

         TEMP=start;

         start=start->next;

         end->next=start;

    }

    delete TEMP;

    return ele;

}

int LIST\_31::DELETE\_END()

{       int ele;

    NODE \*TEMP;

    if(start==NULL)

    {

        cout<<"List is empty"<<endl;

        return NULL;

    }

    if(start==end)

    {

         ele=start->data;

         TEMP=start;

         start=end=NULL;

    }

    else

    {

        NODE \*ptr1=start;

        NODE \*ptr2=NULL;

        while(ptr1->next!=start)

        {

            ptr2=ptr1;

            ptr1=ptr1->next;

        }

        ele=ptr1->data;

        TEMP=ptr1;

        ptr2->next=start;

        ptr2=end;

    }

    delete TEMP;

    return ele;

}

/\*void LIST\_31::ADD\_POS(int pos,int ele)

{

    //create Node

    NODE \*NEW = new NODE();

    //Populate Node

    NEW->data=ele;

    NEW->next=NULL;

    //Set Links

    if(pos==1)

    {

        NEW->next=start;

        start=NEW;

    }

    else

    {

      NODE \*ptr1=start;

      NODE \*ptr2=NULL;

      int count=1;

      while(count<pos)

      {

        ptr2=ptr1;

        ptr1=ptr1->next;

        count=count+1;

      }

      NEW->next=ptr1;

      ptr2->next=NEW;

    }

}

int LIST\_31::DELETE\_POS(int pos)

{

    NODE \*TEMP;

    int ele;

    if(start==NULL)

    {

        cout<<"List is Empty"<<endl;

        return NULL;

    }

    if(pos==1)

    {

        ele=start->data;

        TEMP=start;

        start=start->next;

    }

    else

    {

        NODE \*ptr1=start;

        NODE \*ptr2=NULL;

        int count=1;

        while(count<pos)

        {

        ptr2=ptr1;

        ptr1=ptr1->next;

        count=count+1;

        }

        ele= ptr1->data;

        TEMP=ptr1;

        ptr2->next=ptr1->next;

    }

    delete TEMP;

    return ele;

} \*/

void LIST\_31::LISTALL()

{

    if(start==NULL)

        cout<<"List is Empty"<<endl;

    else

    {

        NODE \*ptr;

        ptr=start;

        while(ptr->next!=start)

        {

             cout<<ptr->data<<" ";

             ptr=ptr->next;

        }

        cout<<ptr->data<<" ";

    }

}

void MENU()

{

   int ch,ele,pos;

   LIST\_31 obj;

   do

   {

    cout<<"\n1.ADD\_FIRST";

    cout<<"\n2.ADD\_END";

    cout<<"\n3.DELETE\_FIRST";

    cout<<"\n4.DELETE\_END";

    /\*cout<<"\n5.ADD\_POS";

    cout<<"\n6.DELETE\_POS";\*/

    cout<<"\n5.LISTALL";

    cout<<"\n6.Exit";

    cout<<"\nEnter your choice"<<endl;

    cin>>ch;

    switch(ch)

    {

        case 1: cout<<"Enter element"<<endl;

            cin>>ele;

            obj.ADD\_FIRST(ele);

            break;

        case 2: cout<<"Enter element"<<endl;

            cin>>ele;

            obj.ADD\_END(ele);

            break;

        case 3: ele=obj.DELETE\_FIRST();

            if(ele!=NULL)

                cout<<ele<<" is deleted"<<endl;

            break;

        case 4: ele=obj.DELETE\_END();

            if(ele!=NULL)

                cout<<ele<<" is deleted"<<endl;

            break;

        /\*case 5: cout<<"Enter Position"<<endl;

            cin>>pos;

            cout<<"Enter Element"<<endl;

            cin>>ele;

            obj.ADD\_POS(pos,ele);

            break;

        case 6: cout<<"Enter Position"<<endl;

            cin>>pos;

            ele=obj.DELETE\_POS(pos);

            if(ele!=NULL)

                cout<<ele<<" is deleted"<<endl;

            break;\*/

        case 5: obj.LISTALL();

            break;

        case 6:

            return;

        default:cout<<"Invalid Case"<<endl;

    }

   }while(1);

}

void main()

{

    clrscr();

    MENU();

    getch();

}