**Assignment No:- 1.6.3**

**Title :- Implementation of program based on Doubly Linked List**

#include<iostream.h>

#include<conio.h>

class NODE

{

    public:

        NODE \*next,\*prev;

        int data;

};

class LIST

{

    private:

        NODE \*start;

    public:

        LIST();

        void ADD\_FIRST(int);

        int DEL\_FIRST();

        void ADD\_END(int);

        int DEL\_END();

        void ADD\_POS(int,int);

        int DEL\_POS(int);

        void LISTALL();

};

LIST::LIST()

{

    start=NULL;

}

void LIST::ADD\_FIRST(int ele)

{

    //create node

    NODE \*NEW = new NODE();

    //Populate node

    NEW->data=ele;

    NEW->next=NULL;

    NEW->prev=NULL;

    //set links

    NEW->next=start;

    if(start!=NULL)

        start->prev=NEW;

    start=NEW;

}

int LIST::DEL\_FIRST()

{

    if(start==NULL)

    {

        cout<<"List is Empty"<<endl;

        return NULL;

    }

    int ele=start->data;

    NODE \*TEMP=start;

    start=start->next;

    if(start!=NULL)

        start->prev=NULL;

    delete TEMP;

    return ele;

}

void LIST::ADD\_END(int ele)

{

    //create node

    NODE \*NEW = new NODE();

    //populate node

    NEW->data=ele;

    NEW->next=NULL;

    NEW->prev=NULL;

    //set links

    if(start==NULL)

        start=NEW;

    else

    {

        NODE \*ptr=start;

        while(ptr->next!=NULL)

            ptr=ptr->next;

        ptr->next=NEW;

        NEW->prev=ptr;

    }

}

int LIST::DEL\_END()

{

    if(start==NULL)

    {

        cout<<"List is empty"<<endl;

        return NULL;

    }

    NODE \*ptr1=start;

    NODE \*ptr2=NULL;

    while(ptr1->next!=NULL)

    {

        ptr2=ptr1;

        ptr1=ptr1->next;

    }

    int ele=ptr1->data;

    NODE \*TEMP=ptr1;

    if(ptr2==NULL)

        start=NULL;

    else

        ptr2->next=NULL;

    delete TEMP;

    return ele;

}

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

{

    //create node

    NODE \*NEW = new NODE();

    //populate node

    NEW->data=ele;

    NEW->next=NULL;

    NEW->prev=NULL;

    //set links

    if(start==NULL)

        start=NEW;

    else

    {

        if(pos==1)

        {

          NEW->next=start;

          start->prev=NEW;

          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;

            if(ptr1!=NULL)

                ptr1->prev=NEW;

            ptr2->next=NEW;

            NEW->prev=ptr2;

        }

    }

}

int LIST::DEL\_POS(int pos)

{

    NODE \*TEMP;

    int ele;

    if(start==NULL)

    {

        cout<<"List is empty"<<endl;

        return NULL;

    }

    else

    {

        if(start->next==NULL)

        {

            ele=start->data;

            TEMP=start;

            start=NULL;

        }

        else

        {

            if(pos==1)

            {

                ele=start->data;

                TEMP=start;

                start=start->next;

                start->prev=NULL;

            }

            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;

               if(ptr2->next!=NULL)

                 ptr1->next->prev=ptr2;

            }

        }

        delete TEMP;

        return ele;

    }

}

void LIST::LISTALL()

{

    if(start==NULL)

        cout<<"List is empty"<<endl;

    else

    {

        NODE \*ptr=start;

        while(ptr!=NULL)

        {

            cout<<ptr->data<<" ";

            ptr=ptr->next;

        }

    }

}

void MENU()

{

    int ch,ele,pos;

    LIST obj;

    do

    {

        cout<<"\n1.ADD\_FIRST";

        cout<<"\n2.DEL\_FIRST";

        cout<<"\n3.ADD\_END";

        cout<<"\n4.DEL\_END";

        cout<<"\n5.ADD\_POS";

        cout<<"\n6.DEL\_POS";

        cout<<"\n7.LISTALL";

        cout<<"\n8.EXIT";

        cout<<"\nEnter your choice";

        cin>>ch;

        switch(ch)

        {

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

                cin>>ele;

                obj.ADD\_FIRST(ele);

                break;

            case 2: ele=obj.DEL\_FIRST();

                if(ele!=NULL)

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

                break;

            case 3: cout<<"Enter element"<<endl;

                cin>>ele;

                obj.ADD\_END(ele);

                break;

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

                if(ele!=NULL)

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

                break;

            case 5: cout<<"Enter element"<<endl;

                cin>>ele;

                cout<<"Enter Position"<<endl;

                cin>>pos;

                obj.ADD\_POS(ele,pos);

                break;

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

                cin>>pos;

                ele=obj.DEL\_POS(pos);

                if(ele!=NULL)

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

                break;

            case 7: obj.LISTALL();

                break;

            case 8:

                return;

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

        }

    }while(1);

}

void main()

{

    clrscr();

    MENU();

    getch();

}