**Assignment No:-1.6**

**Assignment Title:-Implementation of Program based on Singly/Linear link list.**

-----------------------------------------------------------------------------------------------------------------#include<iostream.h>

#include<conio.h>

class NODE

{

public:

int data;

NODE \*next;

};

class LIST

{

private:

NODE \*start;

public:

LIST();

void ADD\_FIRST(int ele);

int DEL\_FIRST();

void ADD\_END(int ele);

int DEL\_END();

void ADD\_POS(int ele,int pos);

int DEL\_POS(int pos);

void LIST\_ALL();

};

void LIST::LIST()

{

start=NULL;

}

void LIST::ADD\_FIRST(int ele)

{

//create a node

NODE \*NEW =new NODE();

if(NEW==NULL)

{

cout<<"LIST is full";

return;

}

//fill up the data

NEW->data= ele;

NEW->next=NULL;

//set the link

NEW->next=start;

start=NEW;

}

int LIST::DEL\_FIRST()

{

if(start==NULL)

{

cout<<"LIST is empty";

return NULL;

}

else

{

int ele=start->data;

NODE \*TEMP=start;

start=start->next;

delete TEMP;

return ele;

}

}

void LIST::ADD\_END(int ele)

{

//create a node

NODE \*NEW =new NODE();

if(NEW==NULL)

{

cout<<"LIST is full";

return;

}

//fill up the data

NEW->data= ele;

NEW->next=NULL;

//set the link

NODE \*ptr;

if(start==NULL)

{

start=NEW;

}

else

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=NEW;

}

int LIST::DEL\_END()

{

if(start==NULL)

{

cout<<"LIST is empty";

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)

ptr2->next=NULL;

else

{

start=NULL;

}

delete Temp;

return ele;

}

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

{

//create a node

NODE \*NEW =new NODE();

if(NEW==NULL)

{

cout<<"LIST is full";

return;

}

//fill up the data

NEW->data= ele;

NEW->next=NULL;

//Assuming given 'pos' in valid

int count;

if(pos==1)

{

NEW ->next=start;

start=NEW;

}

else

{

NODE \*ptr1=start;

NODE \*ptr2=NULL;

count=1;

while(count<pos)

{

ptr2=ptr1;

ptr1=ptr1->next;

count=count+1;

}

NEW ->next=ptr1;

ptr2->next=NEW;

}

}

int LIST::DEL\_POS(int pos)

{

if(start==NULL)

{

cout<<"LIST is empty";

return NULL;

}

NODE \*Temp;

int ele;

//search a node

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:: LIST\_ALL()

{

if(start==NULL)

{

cout<<"LIST is empty";

}

NODE \*ptr;

ptr=start;

while(ptr != NULL)

{

cout<<ptr->data<<" ";

ptr=ptr->next;

}

}

void MENU()

{

LIST obj;

int opt,ele,pos;

do

{

cout<<"\n 1. ADD\_FIRST";

cout<<"\n 2. DEL\_FIRST";

cout<<"\n 3. ADD\_END";

cout<<"\n 4. DEL\_END";

cout<<"\n 5. ADD\_POS";

cout<<"\n 6. DEL\_POS";

cout<<"\n 7. LIST\_ALL";

cout<<"\n 8.EXIT";

cout<<"\n Enter your option: ";

cin>>opt;

switch(opt)

{

case 1:

cout<<"Enter your elemet: ";

cin>>ele;

obj.ADD\_FIRST(ele);

break;

case 2:

ele=obj.DEL\_FIRST();

if(ele != NULL)

{

cout<<ele<<"is deleted";

}

break;

case 3:

cout<<"Enter your elemet: ";

cin>>ele;

obj.ADD\_END(ele);

break;

case 4:

ele=obj.DEL\_END();

if(ele != NULL)

{

cout<<ele<<"is deleted";

}

break;

case 5:

cout<<"Enter your elemet: ";

cin>>ele;

cout<<"Enter your position: ";

cin>>pos;

obj.ADD\_POS(ele,pos);

break;

case 6:

cout<<"Enter your elemet: ";

cin>>pos;

ele=obj.DEL\_POS(pos);

if(ele != NULL)

{

cout<<ele<<"is deleted";

}

break;

case 7:

obj.LIST\_ALL();

break;

case 8:

return;

default:

cout<<"INVALID OPTION";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}