**Assignment No : 1.6**

**Title : Implementation of program based on Linear/Singly Linked list.**

**Name : Patil Leena Arun**

**Roll No : 82**

#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);

void ADD\_END(int ele);

int DELETE\_FIRST();

int DELETE\_END();

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

int DELETE\_POS(int);

void LIST\_ALL();

};

LIST :: LIST()

{

start=NULL;

}

void LIST :: ADD\_FIRST(int ele)

{

NODE \*NEW = new NODE();

if(NEW==NULL)

{

cout<<"List is full";

}

NEW -> data = ele;

NEW -> next = NULL;

NEW -> next = start;

start = NEW;

}

void LIST :: ADD\_END(int ele)

{

NODE \*NEW = new NODE();

NEW->data=ele;

NEW->next=NULL;

if(NEW==NULL)

{

cout<<"List is full";

return;

}

if(start==NULL)

start=NEW;

else

{

NODE \*ptr;

ptr= start;

while(ptr -> next !=NULL)

{

ptr=ptr->next;

}

ptr->next=NEW;

}

}

int LIST :: DELETE\_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;

}

}

int LIST :: DELETE\_END()

{

if(start==NULL)

{

cout<<"List is empty";

return NULL;

}

else

{

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)

{

NODE \*NEW= new NODE();

NEW->data=ele;

NEW->next=NULL;

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 :: DELETE\_POS(int pos)

{

int ele;

NODE \*TEMP;

if(start==NULL)

{

cout<<"List is Empty";

return NULL;

}

else

{

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";

}

else

{

NODE \*ptr;

ptr = start;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void MENU()

{

int opt,ele,pos;

LIST obj;

do

{

cout<<"\n1.Add at begin";

cout<<"\n2.Add at end";

cout<<"\n3.Delete at begin";

cout<<"\n4.Delete at end";

cout<<"\n5.Add at position";

cout<<"\n6.Delete at position";

cout<<"\n7. LIST\_ALL";

cout<<"\n8.Exit";

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

cin>>opt;

switch(opt)

{

case 1:

cout<<"\n Enter Element to Add at begin :\t";

cin>>ele;

obj.ADD\_FIRST(ele);

break;

case 2:

cout<<"\n Enter element to add at end :\t";

cin>>ele;

obj.ADD\_END(ele);

break;

case 3:

ele=obj.DELETE\_FIRST();

if(ele!=NULL)

cout<<ele<<"is deleted";

break;

case 4:

ele=obj.DELETE\_END();

if(ele!=NULL)

cout<<ele<<"is deleted";

break;

case 5:

cout<<"\n Enter element to add at position :\t";

cin>>ele;

cout<<"\n Enter position :";

cin>>pos;

obj.ADD\_POS(ele,pos);

break;

case 6:

cout<<"\n Enter position :";

cin>>pos;

ele=obj.DELETE\_POS(pos);

if(ele!=NULL)

cout<<ele<<" is Deleted"<<endl;

break;

case 7:

obj.LIST\_ALL();

break;

case 8:

return;

default:

cout<<"INVALID OPTION";

}

} while(1);

}

void main()

{

clrscr();

MENU();

getch();

}