**ROLL NO:-45**

**NAME : Harshit Atul Chilvirwar**

**PRACTICAL NO:-**

**PRACTICAL NAME :- IMPLEMENTATION OF SINGAL LINKED LIST**

#include "iostream.h"

#include "conio.h"

class NODE

{

public:

int data;

NODE \*link;

};

class LIST

{

NODE \*start;

public:

LIST();

void ADD\_FIRST(int);

void ADD\_LAST(int);

void ADD\_POS(int,int);

int DEL\_FIRST();

int DEL\_LAST();

int DEL\_POS(int);

void VIEW();

int IS\_EMPTY();

};

LIST::LIST()

{

start = NULL;

}

void LIST::ADD\_FIRST(int ele)

{

NODE \*NN;

NN= new NODE();

if(NN == NULL)

{

cout<<endl<<"List is full";

return;

}

NN->data = ele;

NN->link = NULL;

if(start==NULL)

start=NN;

else

{

NN->link = start;

start = NN;

}

}

void LIST::ADD\_LAST(int ele)

{

NODE \*NN;

NN= new NODE();

NN->data = ele;

NN->link = NULL;

if(start == NULL)

start=NN;

else

{

NODE \*ptr;

ptr= start;

while(ptr->link != NULL)

{

ptr= ptr->link;

}

ptr->link = NN;

}

}

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

{

NODE \*NN;

NN= new NODE();

NN->data = ele;

NN->link = NULL;

if(pos == 1)

{

NN->link = start;

start = NN;

}

else

{

int count = 1;

NODE \*ptr = start;

NODE \*prev = NULL;

while(count < pos)

{

prev = ptr;

ptr = ptr->link;

count = count + 1;

}

NN->link = ptr;

prev->link = NN;

}

}

int LIST::DEL\_FIRST()

{

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

int ele = start->data;

NODE \* TEMP = start;

start = start->link;

delete TEMP;

return ele;

}

int LIST::DEL\_LAST()

{

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

NODE \*ptr = start;

NODE \*prev = NULL;

while(ptr->link != NULL)

{

prev = ptr;

ptr = ptr ->link;

}

int ele = ptr->data;

NODE \*TEMP = ptr;

if (prev ==NULL)

start =NULL;

else

prev->link = NULL;

delete TEMP;

return ele;

}

int LIST::DEL\_POS(int pos)

{

NODE \*TEMP;

int ele;

if(start == NULL)

{

cout<<endl<<"List is empty";

return NULL;

}

else

{

if(pos==1)

{

ele = start->data;

TEMP = start;

start = start->link;

}

else

{

NODE \*ptr, \*prev;

int count = 1;

ptr = start; prev = NULL;

while(count<pos)

{

prev = ptr;

ptr = ptr->link;

count = count+1;

}

ele = ptr->data;

TEMP = ptr;

prev->link = ptr->link;

}

delete TEMP;

return ele;

}

}

int LIST::IS\_EMPTY()

{

if(start == NULL)

return 1;

else

return 0;

}

void LIST::VIEW()

{

if(start == NULL)

{

cout<<endl<<"List is empty";

return;

}

NODE \*ptr = start;

cout<<endl<<"List elements are : ";

while(ptr != NULL)

{

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

ptr=ptr->link;

}

}

void MENU()

{

int ele, opt, pos;

LIST obj;

do

{

cout<<endl<<"1 Add at First";

cout<<endl<<"2 Add at Last";

cout<<endl<<"3 Add at Position";

cout<<endl<<"4 Delete from First";

cout<<endl<<"5 Delete from Last";

cout<<endl<<"6 Delete from Position";

cout<<endl<<"7 List All";

cout<<endl<<"8 Exit Menu";

cout<<endl<<"================\n";

cout<<endl<<"Enter your choice : ";

cin>>opt;

switch(opt)

{

case 1:

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

cin>>ele;

obj.ADD\_FIRST(ele);

obj.VIEW();

break;

case 2:

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

cin>>ele;

obj.ADD\_LAST(ele);

obj.VIEW();

break;

case 3:

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

cin>>ele;

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

cin>>pos;

obj.ADD\_POS(ele,pos);

obj.VIEW();

break;

case 4:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_FIRST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW();

}

break;

case 5:

if(!obj.IS\_EMPTY())

{

ele = obj.DEL\_LAST();

cout<<endl<<"Delted element = "<<ele;

obj.VIEW();

}

break;

case 6:

if(!obj.IS\_EMPTY())

{

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

cin>>pos;

ele = obj.DEL\_POS(pos);

cout<<endl<<"Delted element = "<<ele;

obj.VIEW();

}

break;

case 7:

obj.VIEW();

break;

case 8:

return;

default:

cout<<endl<<"invalid input";

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}