**Assignment No:- 1.6**

**Assignment Name:- Implementation of program based on Stack using Linked List.**

**Name:- YOGESH PRALHAD PATIL.**

**Roll No:- 104.**

#include<iostream.h>

#include<conio.h>

class NODE

{

public:

NODE \*next;

int data;

};

class STACK

{

private:

NODE \*top;

public:

STACK();

void PUSH(int);

int POP();

void VIEW\_ALL();

};

STACK::STACK()

{

top=NULL;

}

void STACK::PUSH(int ele)

{

NODE \*NEW=new NODE();

NEW->data=ele;

NEW->next=NULL;

NEW->next=top;

top=NEW;

}

int STACK::POP()

{

int ele;

if(top==NULL)

{

cout<<"List is Empty : "<<endl;

return(NULL);

}

ele=top->data;

NODE \*TEMP=top;

top=top->next;

delete TEMP;

return ele;

}

void STACK::VIEW\_ALL()

{

int data;

if(top==NULL)

{

cout<<"List is Empty"<<endl;

}

else

{

NODE \*ptr;

ptr=top;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void MENU()

{

int opt,ele;

STACK obj;

do

{

cout<<"\n ----MENU----";

cout<<"\n1.PUSH";

cout<<"\n2.POP";

cout<<"\n3.LIST\_ALL";

cout<<"\n4.EXIT";

cout<<"\n Enter your Option : "<<endl;

cin>>opt;

switch(opt)

{

case 1:

cout<<"Enter an Element : ";

cin>>ele;

obj.PUSH(ele);

break;

case 2:

ele=obj.POP();

if(ele!=NULL)

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

break;

case 3:

cout<<"Elemente in List are : ";

obj.VIEW\_ALL();

break;

case 4:

return;

default:

cout<<"Enter Valid Option"<<endl;

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Assignment No:- 1.6**

**Assignment Name:- Implementation of program based on Queue using Linked List.**

**Name:- YOGESH PRALHAD PATIL.**

**Roll No:- 104.**

#include<iostream.h>

#include<conio.h>

class NODE

{

public:

NODE \*next;

int data;

};

class QUEUE

{

private:

NODE \*front,\*rear;

public:

QUEUE();

void ADD(int);

int DEL();

void VIEW\_ALL();

};

QUEUE::QUEUE()

{

front=NULL;

}

void QUEUE::ADD(int ele)

{

NODE \*NEW=new NODE();

NEW->data=ele;

NEW->next=NULL;

if(front==NULL)

{

front=NEW;

rear=NEW;

}

else

{

rear->next=NEW;

rear=NEW;

}

}

int QUEUE::DEL()

{

int ele;

if(front==NULL)

{

cout<<"QUEUE is Empty : "<<endl;

return(NULL);

}

ele=front->data;

NODE \*TEMP=front;

front=front->next;

delete TEMP;

return ele;

}

void QUEUE::VIEW\_ALL()

{

int data;

if(front==NULL)

{

cout<<"QUEUE is Empty"<<endl;

}

else

{ NODE \*ptr;

ptr=front;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void MENU()

{

int opt,ele;

QUEUE obj;

do

{

cout<<"\n-----MENU-----";

cout<<"\n1.INSERT";

cout<<"\n2.DELETE";

cout<<"\n3.LIST\_ALL";

cout<<"\n4.EXIT";

cout<<"\n Choose your Option : ";

cin>>opt;

switch(opt)

{

case 1:

cout<<"Enter an Element : ";

cin>>ele;

obj.ADD(ele);

break;

case 2:

ele=obj.DEL();

if(ele!=NULL)

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

break;

case 3:

cout<<"Elemente in List are : ";

obj.VIEW\_ALL();

break;

case 4:

return;

default:

cout<<"Enter Valid Option"<<endl;

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Assignment No:- 1.6**

**Assignment Name:- Implementation of program based on Linear Linked List.**

**Name:- YOGESH PRALHAD PATIL.**

**Roll No:- 104.**

#include<iostream.h>

#include<conio.h>

class NODE

{

public:

NODE \*next;

int data;

};

class LIST

{

private:

NODE \*start,\*ptr1,\*ptr2;

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 VIEW\_ALL();

};

LIST::LIST()

{

start=NULL;

}

void LIST::ADD\_FIRST(int ele)

{

NODE \*NEW=new NODE(); //create NODE

NEW->data=ele; //assign data

NEW->next=NULL;

NEW->next=start; //set link

start=NEW;

}

int LIST::DEL\_FIRST()

{

int ele;

if(start==NULL)

{

cout<<"List is Empty : "<<endl;

return(NULL);

}

ele=start->data;

NODE \*TEMP=start;

start=start->next;

delete TEMP;

return ele;

}

void LIST::ADD\_END(int ele)

{

NODE \*NEW=new NODE;

NEW->data=ele;

NEW->next=NULL;

if(start==NULL)

start=NEW;

else

{

ptr1=start;

while(ptr1->next!=NULL)

{

ptr1=ptr1->next;

}

ptr1->next=NEW;

}

}

int LIST::DEL\_END()

{

if(start==NULL)

{

cout<<"List is Empty ";

return NULL;

}

ptr1=start;

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

{

ptr1=start;

ptr2=NULL;

int 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)

{

int ele;NODE \*TEMP;

if(start==NULL)

{

cout<<"List is Empty"<<endl;

return NULL;

}

if(pos==1)

{

ele=start->data;

TEMP=start;

start=start->next;

}

else

{

ptr1=start;

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::VIEW\_ALL()

{

int data;

if(start==NULL)

{

cout<<"List is Empty"<<endl;

}

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<<"\n ----MENU----";

cout<<"\n1.ADD\_FIRST";

cout<<"\n2.DELETE\_FIRST";

cout<<"\n3.ADD\_END";

cout<<"\n4.DELETE\_END";

cout<<"\n5.ADD\_POSITION";

cout<<"\n6.DELETE\_POSITION";

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

cout<<"\n8.EXIT";

cout<<"\n Enter your Option"<<endl;

cin>>opt;

switch(opt)

{

case 1:

cout<<"Enter an Element : ";

cin>>ele;

obj.ADD\_FIRST(ele);

break;

case 2:

ele=obj.DEL\_FIRST();

if(ele!=NULL)

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

break;

case 3:

cout<<"Enter Element to add : ";

cin>>ele;

obj.ADD\_END(ele);

break;

case 4:

ele=obj.DEL\_END();

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

break;

case 5:

cout<<"Enter Element to add at Pos : ";

cin>>ele;

cout<<"Enter POSITION to add : ";

cin>>pos;

obj.ADD\_POS(ele,pos);

break;

case 6:

cout<<"Enter POSITION to Delete : ";

cin>>pos;

ele=obj.DEL\_POS(pos);

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

break;

case 7:

cout<<"Elemente in List are : "<<endl;

obj.VIEW\_ALL();

break;

case 8:

return;

default:

cout<<"Enter Valid Option"<<endl;

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

**Assignment No:- 1.6**

**Assignment Name:- Implementation of program based on Doubly Linked List.**

**Name:- YOGESH PRALHAD PATIL.**

**Roll No:- 104.**

#include<iostream.h>

#include<conio.h>

class NODE

{

public:

NODE \*next,\*prev;

int data;

};

class DQUEUE

{

private:

NODE \*start;

public:

DQUEUE();

void ADD\_FIRST(int);

int DEL\_FIRST();

void ADD\_END(int);

int DEL\_END();

void ADD\_POS(int,int);

int DEL\_POS(int);

void LIST\_ALL();

};

DQUEUE::DQUEUE()

{

start=NULL;

}

void DQUEUE::ADD\_FIRST(int ele)

{

NODE \*NEW=new NODE;

NEW->data=ele;

NEW->next=NULL;

NEW->prev=NULL;

NEW->next=start;

if(start!=NULL)

start->prev=NEW;

start=NEW;

}

int DQUEUE::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 DQUEUE::ADD\_END(int ele)

{

NODE \*ptr;

NODE \*NEW=new NODE;

NEW->data=ele;

NEW->next=NULL;

NEW->prev=NULL;

if(start==NULL)

start=NEW;

else

{

ptr=start;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next=NEW;

NEW->prev=ptr;

}

}

int DQUEUE::DEL\_END()

{

NODE \*ptr1,\*ptr2;

if(start==NULL)

{

cout<<"List is Empty"<<endl;

return NULL;

}

ptr1=start;

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 DQUEUE::ADD\_POS(int ele,int pos)

{

NODE \*ptr1,\*ptr2;

NODE \*NEW=new NODE;

NEW->data=ele;

NEW->next=NULL;

NEW->prev=NULL;

if(start==NULL)

start=NEW;

else

{

if(pos==1)

{

NEW->next=start;

start->prev=NEW;

start=NEW;

}

else

{

ptr1=start;

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;

NEW->prev=ptr2;

ptr2->next=NEW;

}

}

}

int DQUEUE::DEL\_POS(int pos)

{

NODE \*ptr1,\*ptr2;

int ele;

NODE \*TEMP;

if(start==NULL)

{

cout<<"List is Empty"<<endl;

return NULL;

}

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

{

ptr1=start;

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(ptr1->next!=NULL)

ptr1->next->prev=ptr2;

}

}

delete TEMP;

return ele;

}

void DQUEUE::LIST\_ALL()

{

NODE \*ptr;

if(start==NULL)

{

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

}

else

{

ptr=start;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void MENU()

{

int ch,ele,pos;

DQUEUE obj;

do

{

cout<<"\n----MENU----";

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.LIST\_ALL";

cout<<"\n8.EXIT";

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

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enter Element to add first : ";

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 to add End : ";

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 to add: ";

cin>>ele;

cout<<"Enter position : ";

cin>>pos;

obj.ADD\_POS(ele,pos);

break;

case 6:

cout<<"Enter position to delete : ";

cin>>pos;

ele=obj.DEL\_POS(pos);

if(ele!=NULL)

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

break;

case 7:

cout<<" Elements in List are : ";

obj.LIST\_ALL();

break;

case 8:

return;

default:

cout<<"Invalid Option"<<endl;

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}