Assignment No:=1.3

Title:=Implementation of Program based on Stack

Name:=Deshpande Piyush Bhikan

R.No=31

#include<iostream.h>

#include<conio.h>

class STACK\_31

{

private:

int \*A,s,top;

public:

STACK\_31(int);

void PUSH(int ele);

int POP();

void LIST\_ALL();

};

//---------------------------------

STACK\_31::STACK\_31(int par)

{

s=par;

top=0;

A=new int[s+1];

}

void STACK\_31::PUSH(int ele)

{

if(top==s)

{

cout<<"\n stack is full" ;

return;

}

top=top+1;

A[top]=ele;

}

int STACK\_31::POP()

{

if (top==0)

{

cout <<"\n stack is empty" ;

return NULL;

}

int ele=A[top];

top=top-1;

return ele;

}

void STACK\_31::LIST\_ALL()

{

if (top==0)

{

cout<<"\n stack is empty" ;

}

else

{

cout<<endl<<"\n element in stack" ;

for(int i=top;i>=1;i--)

{

cout<<A[i]<<" " ;

}

}

}

//----------------------------------------

void MENU()

{

int opt,ele,size;

cout<<"\n enter size of stack" ;

cin>>size;

STACK\_31 obj (size);

do

{

cout<<"\n 1.PUSH" ;

cout<<"\n 2.POP" ;

cout<<"\n 3.LIST" ;

cout<<"\n 4.EXIT" ;

cout<<"\n enter your option" ;

cin>>opt;

switch (opt)

{

case 1:

cout<<"\n enter ele to add" ;

cin>>ele;

obj.PUSH (ele);

break;

case 2:

int ele=obj.POP();

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

break;

case 3:

obj.LIST\_ALL();

break;

case 4:return;

default:cout<<"\n Invalid Option" ;

}

}while(1);

}

void main()

{

int ele;

clrscr();

//---------------

MENU();

//---------------

getch();

}

//Simple Queue

Assignment No:=1.4

Title:=Implementation of Program based on queue

Name:=Deshpande Piyush Bhikan

R.No=31

#include<iostream.h>

#include<conio.h>

class QUEUE\_31

{

private:

int \*A,size,front,rear;

public:

QUEUE\_31(int);

void insert(int);

int del();

void listall();

};

QUEUE\_31::QUEUE\_31(int par)

{

size=par;

A=new int[size+1];

front=rear=0;

}

void QUEUE\_31::insert(int ele)

{

if(rear==size)

{

cout<<"Queue is full"<<endl;

return;

}

if(front==0)

front++;

rear++;

A[rear]=ele;

}

int QUEUE\_31::del()

{

if(front==0)

cout<<"Queue is empty"<<endl;

int ele=A[front];

if(front==rear)

front=rear=0;

else

front++;

return ele;

}

void QUEUE\_31::listall()

{

if(front==0)

cout<<"Queue is empty"<<endl;

else

for(int i=front;i<=rear;i++)

cout<<A[i]<<"\t";

}

void MENU()

{

int ch,size,ele;

cout<<"Enter size of queue"<<endl;

cin>>size;

QUEUE\_31 obj(size);

do

{

cout<<"\n1.Insert\n2.Delete\n3.List All\n4.Exit"<<endl;

cin>>ch;

switch(ch)

{

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

cin>>ele;

obj.insert(ele);

break;

case 2:

cout<<obj.del()<<" is deleted"<<endl;

break;

case 3: obj.listall();

break;

case 4: return;

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

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}

//Circular Queue

Assignment No:=1.4

Title:=Implementation of Program based on Circular queue

Name:=Deshpande Piyush Bhikan

R.No=31

#include<iostream.h>

#include<conio.h>

class QUEUE\_31

{

private:

int \*A,size,front,rear;

public:

QUEUE\_31(int);

void insert(int);

int del();

void listall();

};

QUEUE\_31::QUEUE\_31(int par)

{

size=par;

front=rear=0;

A=new int[size+1];

}

void QUEUE\_31::insert(int ele)

{

if((front==1 && rear==size)||(rear+1==front))

{

cout<<"Queue is Full"<<endl;

return;

}

if(front==0)

front=1;

if(rear==size)

rear=0;

rear++;

A[rear]=ele;

}

int QUEUE\_31::del()

{

if(front==0)

{

cout<<"Queue is Empty"<<endl;

return NULL;

}

int ele=A[front];

if(front==rear)

front=rear=0;

else

{

if(front==size)

front=0;

front++;

}

return ele;

}

void QUEUE\_31::listall()

{

int i;

if(front==0)

{

cout<<"Queue is Empty"<<endl;

return;

}

if(front<=rear)

{

for(i=front;i<=rear;i++)

cout<<A[i]<<"\t";

}

else

{

for(i=front;i<=size;i++)

cout<<A[i]<<"\t";

for(i=1;i<=rear;i++)

cout<<A[i]<<"\t";

}

}

void MENU()

{

int ch,size,ele;

cout<<"Enter size of queue"<<endl;

cin>>size;

QUEUE\_31 obj(size);

do

{

cout<<"\n1.Insert\n2.Delete\n3.List All\n4.Exit"<<endl;

cin>>ch;

switch(ch)

{

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

cin>>ele;

obj.insert(ele);

obj.listall();

break;

case 2:

cout<<obj.del()<<" is deleted"<<endl;

obj.listall();

break;

/\* case 3: obj.listall();

break; \*/

case 4: return;

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

}

}while(1);

}

void main()

{

clrscr();

MENU();

getch();

}