# Task 2

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node \*next;

};

class Queue{

public:

Node \*front,\*rear;

Queue()

{

front=NULL;

rear=NULL;

}

void add(int p);

void del();

void display();

};

void Queue::add(int p){

Node \*temp=new Node;

if(temp==NULL){

cout<<"Overflow"<<endl;

return;

}

temp->data=p;

temp->next=NULL;

if(front==NULL){

front=rear=temp;

}

else{

rear->next=temp;

rear=temp;

}

}

void Queue::display(){

if(front==NULL){

cout<<"Underflow."<<endl;

return;

}

else{

cout<<"elements"<<endl;

Node \*temp=front;

while(temp){

cout<<temp->data<<" ";

temp=temp->next;

}

cout<<endl;}

}

void Queue :: del()

{

if (front==NULL){

cout<<"underflow"<<endl;

return;

}

if(front==rear)

front=rear=NULL;

else

front=front->next;

}

int main(){

Queue Q;

Q.display();

Q.add(10);

Q.add(24);

Q.add(28);

Q.add(32);

Q.add(30);

Q.display();

Q.del();

cout<<"after deleteion:"<<endl;

Q.display();

return 0;

}

# Lab task 1

#include<iostream>

using namespace std;

class que{

private:

int arr[5],front,rear;

public:

que(){

front=-1;

rear=-1;

}

Boolisfull()

{

if(front==-1&&rear==5-1)

{

return true;

}

else

return false;

}

// boolisempty()

// {

// if(front==-1)

// {

// return true;

// }

// else

// return false;

// }

int enque (int value)

{

if(Boolisfull())

{

cout<<"que is fullfil"<<endl;

}

else {

if(front==-1)

{

// front=0;

arr[rear]=value;

rear++;

cout<<value<<"is added"<<endl;

}

}

}

int deque()

{

int value;

if(Boolisfull())

{

cout<<"stack is already empty"<<endl;

}

else{

value=arr[front];}

if(front>=rear)

{

front=-1;

rear=-1;

}

else

{

cout<<"deleted element from que"<<endl;

value--;

return (value);

}

}

void display()

{

// if(boolisempty())

// {

// cout<<"que is empty"<<endl;

// }

{

cout<<"que elemnts are:"<<endl;

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

{

cout<<arr[i]<<endl;

}

}

}

};

int main()

{

que q;

cout<<"que created:"<<endl;

q.enque(4);

q.enque(5);

q.enque(6);

q.enque(7);

q.enque(8);

q.display();

q.deque();

q.display();

}