Heaven’s light is our guide



**Rajshahi University of Engineering and Technology**

**Dept.: Electrical & Computer Engineering**

**Lab Report- 1**

**Course title : Data Structure & Algorithms Sessional**

**Course code : ECE- 2104**

**Date of exp. : 04-01-2025**

**Date of submission : 11-01-2025**

|  |  |
| --- | --- |
| **Submitted to:** | **Submitted by:** |
| **Md. Omaer Faruq Goni**  Assistant Professor  Electrical & Computer Engineering  RUET | **Asraful Islam**  ID : 2210016  Reg. No. : 1070  Session : 2022-2023  Dept. of ECE, RUET |

**Problem- 1 : Write a C++ program to implement a queue using an array with enqueue and dequeue operation and find the top element of the stack and check if the stack is empty or not**

**Code :**

#include<iostream>

using namespace std;

class Queue{

private:

int \*arr;

int front , rear, size;

public:

Queue(int s){

arr =new int [s];

size=s;

front=0;

rear=-1;

}

~Queue(){

delete []arr;

}

void enqueue(int x){

if(rear==size-1){

cout<<"queue overflow"<<endl;

return;

}

arr[++rear]=x;

}

void dequeue(){

if(front>rear){

cout<<"queue underflow"<<endl;

return ;

}

front++;

}

int getfront(){

if(front > rear){

cout<<"queue is empty"<<endl;

return -1;

}

return arr[front];

}

bool isempty(){

return front > rear;

}

};

int main(){

Queue q(10);

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.enqueue(40);

cout<<"Front element : "<<q.getfront()<<endl;

q.dequeue();

cout<<"after dequeue Front element : "<<q.getfront()<<endl;

q.dequeue();

cout<<"after dequeue Front element : "<<q.getfront()<<endl;

q.dequeue();

cout<<"after dequeue Front element : "<<q.getfront()<<endl;

q.dequeue();

cout<<"after dequeue Front element : "<<q.getfront()<<endl;

cout<<"queue is emty ???"<<endl;

if(q.isempty()){

cout<<"yes"<<endl;

}

else{

cout<<"no"<<endl;

}

}

**Output:**

|  |
| --- |
| Front element : 10  after dequeue Front element : 20  after dequeue Front element : 30  after dequeue Front element : 40  after dequeue Front element : queue is empty  -1  queue is emty ???  yes% |

**Problem- 2 : Write a C++ program to remove a given element from a queue.**

**Code :**

#include<iostream>

using namespace std;

class Queue{

private:

int \*arr;

int front , rear, size;

public:

Queue(int s){

arr =new int [s];

size=s;

front=0;

rear=-1;

}

~Queue(){

delete []arr;

}

void enqueue(int x){

if(rear==size-1){

cout<<"queue overflow"<<endl;

return;

}

arr[++rear]=x;

}

void dequeue(){

if(front>rear){

cout<<"queue underflow"<<endl;

return ;

}

front++;

}

int getfront(){

if(front > rear){

cout<<"queue is empty"<<endl;

return -1;

}

return arr[front];

}

bool isempty(){

return front > rear;

}

void remove(int e){

while(e!=arr[front]){

dequeue();

}

dequeue();

}

void display(){

cout<<"Queue :"<<endl;

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

};

int main(){

Queue q(15);

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.enqueue(40);

q.enqueue(50);

q.display();

q.remove(30);

cout<<"after removing element ";

q.display();

}

**Output:**

|  |
| --- |
| Queue :  10 20 30 40 50  after removing element Queue :  40 50 |

**Problem- 3 :Write a C++ program to sort the element of a queue**

**Code :**

#include<iostream>

using namespace std;

class Queue{

private:

int \*arr;

int front , rear, size;

public:

Queue(int s){

arr =new int [s];

size=s;

front=0;

rear=-1;

}

~Queue(){

delete []arr;

}

void enqueue(int x){

if(rear==size-1){

cout<<"queue overflow"<<endl;

return;

}

arr[++rear]=x;

}

void dequeue(){

if(front>rear){

cout<<"queue underflow"<<endl;

return ;

}

front++;

}

int getfront(){

if(front > rear){

cout<<"queue is empty"<<endl;

return -1;

}

return arr[front];

}

void sort(){

for(int i=0;i<size-1;i++){

for(int j=0;j<size-i-1;j++){

if(arr[j]>arr[j+1]){

int temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

}

void display(){

cout<<"Queue :"<<endl;

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

};

int main(){

Queue q(5);

q.enqueue(9);

q.enqueue(2);

q.enqueue(5);

q.enqueue(8);

q.enqueue(1);

q.display();

q.sort();

cout<<"after sort";

q.display();

}

**Output :**

|  |
| --- |
| Queue :  9 2 5 8 1  after sortQueue :  1 2 5 8 9 |