**DoublyEndedQueue.cpp**

**Code:**

#include<iostream>

using namespace std;

#define QSize 5

class queue{

int arr[QSize];

int qfront;

int qrear;

int size = 0;

public:

queue(){

qfront = -1;

qrear = -1;

}

void insertRear(int val){

if(isFull()== true){

cout << "Queue is Full";

}

else if(isEmpty()==true){

qrear = 0;

qfront = 0;

arr[qrear] = val;

size++;

}

else{

qrear = (qrear + 1)%QSize;

arr[qrear] = val;

size++;

}

}

void insertFront(int val){

if(isFull() == true){

cout << "Queue is Full";

}

else if(isEmpty()==true){

qrear = 0;

qfront = 0;

arr[qrear] = val;

size++;

}

else{

qfront = (qfront-1+QSize) % QSize;

arr[qfront] = val;

size++;

}

}

void deleteElementfront()

{

int q\_element;

q\_element=arr[qfront];

if(isEmpty()==true)

{

cout<<"Queue is empty, can not delete" <<endl;

}

else if(qfront==qrear)

{

q\_element=arr[qfront];

qrear=-1;

qfront=-1;

cout<<"\n Delete Element is: "<<q\_element<<endl;

size--;

}

else

{

q\_element=arr[qfront];

qfront=(qfront+1)%QSize;

cout<<"\n Deleted Element is: "<<q\_element<<endl;

size--;

}

}

void deleteElementrear()

{

int q\_element;

q\_element=arr[qfront];

if(isEmpty()==true)

{

cout<<"\n Queue is empty, can not delete"<<endl;

}

else if(qfront==qrear)

{

q\_element=arr[qrear];

qrear=-1;

qfront=-1;

cout<<"\n Delete Element is: "<<q\_element<<endl;

size--;

}

else

{

q\_element=arr[qrear];

qrear=(qrear-1+QSize)%QSize;

cout<<"\n Deleted Element is: "<<q\_element<<endl;

size--;

}

}

bool isEmpty()

{

if(qrear==-1)

return true;

else

return false;

}

bool isFull()

{

if((qrear+1)%QSize == qfront)

return true;

else

return false;

}

int size1()

{

return size;

}

void displayElement()

{

if(qrear==-1)

{

cout<<"No element to display"<<endl;

return;

}

cout<<"Element in the queue are: ";

for(int i=qfront;i!=qrear;i=(i+1)%QSize)

{

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

}

cout<<arr[qrear]<<" ";

cout<<endl;

}

};

int main()

{

queue myqueue;

int val;

int choice;

while(1)

{

cout<<"1. Insert at Front: \n";

cout<<"2. Insert at Rear: \n";

cout<<"3. Delete at Front: \n";

cout<<"4. Delete at Rear: \n";

cout<<"5. Display: \n";

cout<<"6. Size: \n";

cout<<"7. Exit \n";

cout<<"Enter Choice ";

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter the value:";

cin>>val;

myqueue.insertFront(val);

break;

case 2:

cout<<"Enter the Value: ";

cin>>val;

myqueue.insertRear(val);

break;

case 3:

myqueue.deleteElementfront();

break;

case 4:

myqueue.deleteElementrear();

break;

case 5:

myqueue.displayElement();

break;

case 6:

cout << "Size of the Queue:" << myqueue.size1()<<endl;

break;

case 7:

exit(0);

default:

cout<<"Wrong Choice!!! ";

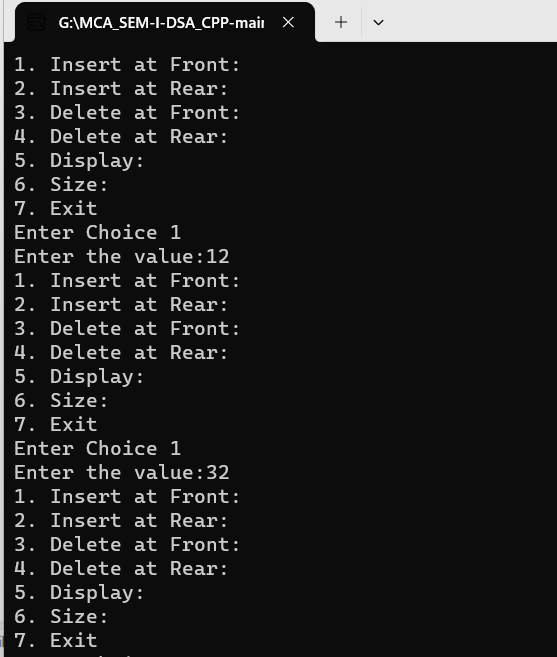
}

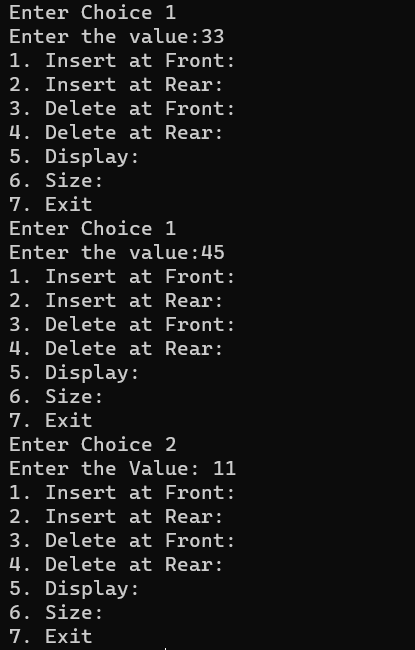
}

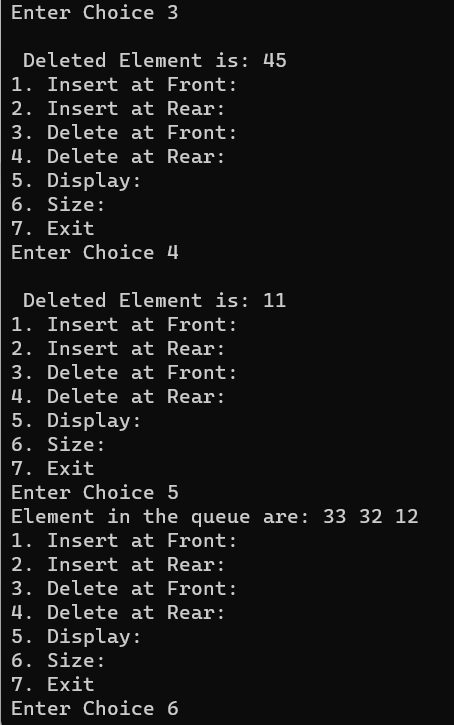
return 0;

}

**Output:**







**PriorityQueue.cpp**

**Code:**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#define QUEUESIZE 5

using namespace std;

class node

{

public:

int value;

int priority;

node()

{

value=0;

priority=0;

}

};

class queue

{

private:

node arr[QUEUESIZE];

int q\_front;

int q\_rear;

public:

queue();

void insertElement(int,int);

int deleteElement();

bool is\_empty();

bool is\_full();

int size();

void displayElement();

};

queue::queue()

{

q\_front=-1;

q\_rear=-1;

}

void queue::insertElement(int val,int pr)

{

if(is\_empty()==true)

{

q\_rear=0;

q\_front=0;

arr[q\_rear].value=val;

arr[q\_rear].priority=pr;

}

else

{

int walker=q\_rear;

while((arr[walker].priority)<pr)

{

arr[(walker+1)%QUEUESIZE].value=arr[walker].value;

arr[(walker+1)%QUEUESIZE].priority=arr[walker].priority;

walker=(walker-1+QUEUESIZE)%QUEUESIZE;

if((walker+1)%QUEUESIZE==q\_front){

break;

}

}

walker=(walker+1)%QUEUESIZE;

arr[walker].value=val;

arr[walker].priority=pr;

q\_rear=(q\_rear+1)%QUEUESIZE;

}

}

int queue::deleteElement()

{

int q\_element;

q\_element=arr[q\_front].value;

if(q\_front==q\_rear)

{

q\_rear=-1;

q\_front=-1;

}

else

{

q\_front=(q\_front+1)%QUEUESIZE;

}

return q\_element;

}

bool queue::is\_empty()

{

if(q\_rear==-1)

return true;

else

return false;

}

bool queue::is\_full()

{

if((q\_rear+1)%QUEUESIZE==q\_front)

return true;

else

return false;

}

int queue::size()

{

return QUEUESIZE-(q\_rear-q\_front+1);

}

void queue::displayElement()

{

if(q\_rear==-1)

{

cout<<"No element to display"<<endl;

return;

}

cout<<"Element in the queue are:\n";

for(int i=q\_front;i!=q\_rear;i=(i+1)%QUEUESIZE)

{

cout<<"\nValue: "<<arr[i].value<<" Priority:"<<arr[i].priority<<" ;" << endl;

}

cout<<"\nValue: "<<arr[q\_rear].value<<" Priority:"<<arr[q\_rear].priority<<" ;" <<endl;

}

int main(){

queue myqueue;

int val,pr;

int choice;

while(1)

{

cout<<"1.Insert\n";

cout<<"2.Delete\n";

cout<<"3.Display\n";

cout<<"4.Quit\n";

cout<<"Enter your choice:";

cin>>choice;

switch(choice)

{

case 1:

if(myqueue.is\_full()==false){

cout<<"\nEnter value to be pushed:";

cin>>val;

cout<<"Enter priority of the value to be punched:";

cin>>pr;

myqueue.insertElement(val,pr);

}

else

cout<<"Queue is full,can't insert"<<endl;

break;

case 2:

if(myqueue.is\_empty()==false)

{

val=myqueue.deleteElement();

cout<<"\n Delete Elemnet is:"<<val<<endl;

}

else

cout<<"\n Queue is empty,can't delete"<<endl;

break;

case 3:

myqueue.displayElement();

break;

case 4:

exit(1);

default:

cout<<"Wrong choice\n";

}

}

return 0;

}

**Output:**

