DoublyEndedQueue.cpp
Code:

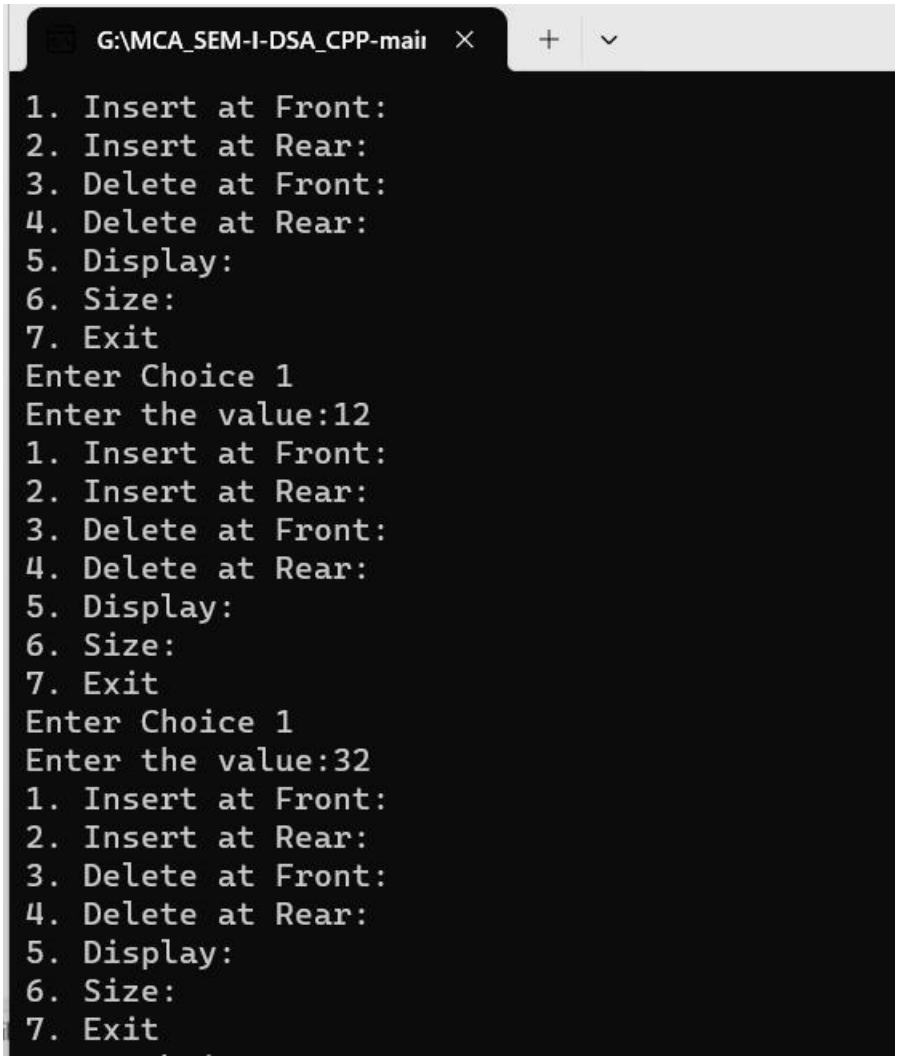
```
#include<iostream>
using namespace std;
#define QSize 5
class queue{
	int arr[QSize];
	int qfront;
	int qrear;
	int size = 0;

	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:



```
Enter Choice 1
Enter the value:33
1. Insert at Front:
2. Insert at Rear:
3. Delete at Front:
4. Delete at Rear:
5. Display:
6. Size:
7. Exit
Enter Choice 1
Enter the value:45
1. Insert at Front:
2. Insert at Rear:
3. Delete at Front:
4. Delete at Rear:
5. Display:
6. Size:
7. Exit
Enter Choice 2
Enter the Value: 11
1. Insert at Front:
2. Insert at Rear:
3. Delete at Front:
4. Delete at Rear:
5. Display:
6. Size:
7. Exit
```

```
Enter Choice 3

 Deleted Element is: 45
1. Insert at Front:
2. Insert at Rear:
3. Delete at Front:
4. Delete at Rear:
5. Display:
6. Size:
7. Exit
Enter Choice 4

 Deleted Element is: 11
1. Insert at Front:
2. Insert at Rear:
3. Delete at Front:
4. Delete at Rear:
5. Display:
6. Size:
7. Exit
Enter Choice 5
Element in the queue are: 33 32 12
1. Insert at Front:
2. Insert at Rear:
3. Delete at Front:
4. Delete at Rear:
5. Display:
6. Size:
7. Exit
Enter Choice 6
```

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:

```
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice:1

Enter value to be pushed:12
Enter priority of the value to be punched:3
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice:1

Enter value to be pushed:23
Enter priority of the value to be punched:2
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice:1

Enter value to be pushed:43
Enter priority of the value to be punched:4
1.Insert
2.Delete
3.Display
4.Quit
```

```
Enter your choice:3
Element in the queue are:

Value: 43  Priority:4 ;

Value: 12  Priority:3 ;

Value: 23  Priority:2 ;
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice:2

 Delete Elemnet is:43
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice:4
```