

```
1 using System;
2
3 namespace coursework
4 {
5     class PriorityQueue
6     {
7         int size;
8         int front, rear, maxSize;
9         int[,] queue;
10        public PriorityQueue(int maxSize)
11        {
12            size = 0;
13            front = 0;
14            rear = -1;
15            this.maxSize = maxSize;
16            queue = new int[maxSize, 2];
17        }
18        /// <summary>
19        /// add item to queue
20        /// </summary>
21        /// <param name="item"></param>
22        /// <param name="priority"></param>
23        public void enqueue(int item, int priority)
24        {
25            //check if item is in queue, if yes update the distance rather than adding item
26            if (!isFull())
27            {
28                size = size + 1;
29                int i = rear;
30                while (i >= front)
31                {
32                    if (priority < queue[i, 1])
33                    {
34                        queue[i + 1, 0] = queue[i, 0];
35                        queue[i + 1, 1] = queue[i, 1];
36                    }
37                    else { break; }
38                    i--;
39                }
40                queue[i + 1, 0] = item;
41                queue[i + 1, 1] = priority;
42                rear++;
43            }
44            else
45            {
46                Console.WriteLine("Queue is full. {0} could not be added.", item);
47            }
48        }
49        /// <summary>
50        /// reorders the queue as item priorities are altered
51        /// </summary>
52        /// <param name="item"></param>
53        /// <param name="priority"></param>
54        public void UpdateQueue(int item, int priority)
```

```
55     {
56         for (int i = front; i <= rear; i++)
57         {
58             if (queue[i, 0] == item)
59             {
60                 queue[i, 1] = priority;
61                 BubbleSort();
62             }
63         }
64     }
65     /// <summary>
66     /// return first item from the queue
67     /// </summary>
68     /// <returns></returns>
69     public int dequeue()
70     {
71         size = size - 1;
72         int item = queue[front, 0];
73         if (front != maxSize - 1)
74         {
75             front = front + 1;
76         }
77         else
78         {
79             front = 0;
80         }
81         return item;
82     }
83
84     public bool isFull()
85     {
86         if (size == maxSize || rear == maxSize - 1)
87         {
88             Console.WriteLine("Queue is full\n");
89             return true;
90         }
91         else
92         {
93             return false;
94         }
95     }
96     public bool isEmpty()
97     {
98         if (size == 0)
99         {
100             return true;
101         }
102         else
103         {
104             return false;
105         }
106     }
107     /// <summary>
108     /// bubble sort algorithm
109     /// </summary>
110     public void BubbleSort()
```

```
111     {
112         int[] temp = new int[2];
113         for (int j = front; j < rear; j++)
114         {
115             for (int i = front; i < rear; i++)
116             {
117                 if (queue[i, 1] > queue[i + 1, 1])
118                 {
119                     for (int x = 0; x < 2; x++)
120                     {
121                         temp[x] = queue[i, x];
122                         queue[i, x] = queue[i + 1, x];
123                         queue[i + 1, x] = temp[x];
124                     }
125                 }
126             }
127         }
128     }
129 }
130 }
131
```