```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace QueueProj
{
    /*
     *           אוסף פעולות עזר לתור
     */
    internal class QueueUtil
    {
        //--- פעולה המקבלת תור של מס' שלמים ומספר נוסף ---
        //--- ומחזירה אמת אם המספר קיים בתור ושקר אחרת ---
        //---                 בסיום התור יישאר ללא שינוי ---
        public static bool Exist(Queue<int> que, int x)
        {
            bool found = false;
            Queue<int> qTemp = new Queue<int>();

            //---          חיפוש והעברה לתור העזר עד
            //---   למציאת האיבר או עד שהתור התרוקן ---
            while (!que.IsEmpty() && !found)
            {
                if (que.Head() == x)
                    found = true;
                qTemp.Insert(que.Remove());
            }

            //--- העברת שאר האיברים לתור העזר ---
            while (!que.IsEmpty())
                qTemp.Insert(que.Remove());

            //--- החזרת האיברים לתור ---
            while (!qTemp.IsEmpty())
                que.Insert(qTemp.Remove());

            return found;
        }

        //--- פעולה המחזירה את אורך התור של שלמים ---
        public static int Size(Queue<int> que)
        {
            Queue<int> qTemp = new Queue<int>();
            int count = 0;
            while (!que.IsEmpty())
            {
                count++;
                qTemp.Insert(que.Remove());
            }
            while (!qTemp.IsEmpty())
                que.Insert(qTemp.Remove());
            return count;
        }
```

```
56
57          //---      פעולה המקבלת תור ומספר ---
58          //--- ומחזירה את מספר מופעיו בתור ---
59          //---   בסיום נשאר התור ללא שינוי ---
60          public static int CountX(Queue<int> que, int x)
61          {
62              Queue<int> qTemp = new Queue<int>();
63              int count = 0;
64              while (!que.IsEmpty())
65              {
66                  if (que.Head() == x)
67                      count++;
68                  qTemp.Insert(que.Remove());
69              }
70              //--- החזרת האיברים לתור ---
71              while (!qTemp.IsEmpty())
72                  que.Insert(qTemp.Remove());
73              return count;
74          }
75
76          //--- פעולה המחזירה את סכום האיברים בתור ---
77          //---            שימוש בתור מעגלי        ---
78          public static int QueSum(Queue<int> que)
79          {
80              int sum = 0;
81              int demo = int.MinValue;
82              que.Insert(demo);
83              while (que.Head() != demo)
84              {
85                  sum += que.Head();
86                  que.Insert(que.Remove());
87              }
88              que.Remove();
89              return sum;
90          }
91
92          //--- פעולה המחזירה את המספר הגדול ביותר בתור ---
93          public static int QueMax(Queue<int> que)
94          {
95              int max = -20, x;
96              if (!que.IsEmpty())
97                  max = que.Head();
98              int demo = int.MinValue;
99              que.Insert(demo);
100             while (que.Head() != demo)
101             {
102                 x = que.Remove();
103                 if (x > max)
104                     max = x;
105                 que.Insert(x);
106             }
107             que.Remove();       //  הסרת חולית הדמה
108             return max;
109         }
110
```

```
111         //--- פעולה המחזירה את האיבר שנמצא בסוף התור מבלי להוציאו ---
112         public static int GetLast(Queue<int> que)
113         {
114             Queue<int> qTemp = new Queue<int>();
115             int x = -1;     //  חובה לאתחל את x
116             //---                העברת כל האיברים לתור העזר ---
117             //--- בסיום הלולאה x יכיל את האיבר האחרון בתור ---
118             while (!que.IsEmpty())
119             {
120                 x = que.Remove();
121                 qTemp.Insert(x);
122             }
123             //--- החזרת האיברים מתור העזר בחזרה לתור ---
124             while (!qTemp.IsEmpty())
125                 que.Insert(qTemp.Remove());
126             return x;
127         }
128 
129         //--- פעולה המקבלת תור של מספרים חיוביים ומספר ---
130         //---     אם המספר קיים בתור, יימחקו כל מופעיו ---
131         //---          שאר איברי התור יישארו ללא שינוי ---
132         public static void DeleteAllX(Queue<int> que, int x)
133         {
134             int demo = -1;      //  כל מספר שלילי יכול להיות איבר דמה
135             que.Insert(demo);
136             int y;
137             while (que.Head() != -1)
138             {
139                 y = que.Remove();
140                 if (y != x)
141                     que.Insert(y);
142             }
143             que.Remove();   //  הסרת איבר הדמה
144         }
145 
146         //--- פעולה המקבלת תור של מספרים חיוביים ומספר ---
147         //---         אם המספר קיים בתור, רק הוא יימחק ---
148         //---          שאר איברי התור יישארו ללא שינוי ---
149         public static void DeleteX(Queue<int> que, int x)
150         {
151             int demo = -1;  //  כל מספר שלילי יכול להיות איבר דמה
152             que.Insert(demo);
153             bool deleted = false;
154             int y;
155             while (que.Head() != -1)
156             {
157                 y = que.Remove();
158                 if (y == x && !deleted)
159                     deleted = true;
160                 else
161                     que.Insert(y);
162             }
163             que.Remove();   //  הסרת איבר הדמה
164         }
165 
```

```
166         //--- היפוך איברי התור - פעולה רקורסיבית ---
167         public static void ReverseQue(Queue<int> que)
168         {
169             if (!que.IsEmpty())
170             {
171                 int x = que.Remove();
172                 ReverseQue(que);
173                 que.Insert(x);
174             }
175         }
176 
177         //--- פעולה המוצאת את האיבר המינימלי בתור ---
178         //---                 ומוחקת את כל מופעיו. ---
179         //---  בסיום שאר האיברים נשארים באותו סדר ---
180         //---                     הנחה: התור לא ריק ---
181         public static void RemoveMinValue(Queue<int> que)
182         {
183             Queue<int> qTemp = new Queue<int>();
184             int min = que.Head();
185             int x;
186 
187             //--- מציאת המינימום תוך העברת ---
188             //---  כל איברי התור לתור העזר ---
189             while (!que.IsEmpty())
190             {
191                 x = que.Remove();
192                 if (x < min)
193                     min = x;
194                 qTemp.Insert(x);
195             }
196 
197             //--- החזרת כל האיברים פרט ---
198             //--- למופעי min בחרה לתור ---
199             while (!qTemp.IsEmpty())
200             {
201                 x = qTemp.Remove();
202                 if (x != min)
203                     que.Insert(x);
204             }
205         }
206 
207 
208 
209 
210 
211 
212 
213 
214 
215 
216 
217 
218 
219 
220 
```

```
221         //--- פעולה המוצאת ומחזירה את האיבר המינימלי בתור ---
222         //--- אם יש שניים כאלה, יוחזר האחרון ---
223         //---  בסיום שאר האיברים נשארים באותו סדר ---
224         public static int GetMinValue(Queue<int> que)
225         {
226             Queue<int> qTemp = new Queue<int>();
227             int min = que.Head();
228             int x;
229 
230             //--- מציאת המינימום תוך העברת ---
231             //---  כל איברי התור לתור העזר ---
232             while (!que.IsEmpty())
233             {
234                 x = que.Remove();
235                 if (x < min)
236                     min = x;
237                 qTemp.Insert(x);
238             }
239 
240             //---     החזרת כל האיברים פרט ---
241             //--- הראשון בחזרה לתור min -ל ---
242             bool deleted = false;
243             while (!qTemp.IsEmpty())
244             {
245                 x = qTemp.Remove();
246                 if (x == min && !deleted)
247                     deleted = true;
248                 else
249                     que.Insert(x);
250             }
251             return min;
252         }
253 
254         //--- פעולה הממיינת תור בסדר עולה ---
255         public static void Sort(Queue<int> que)
256         {
257             Queue<int> q1 = new Queue<int>();
258             int x;
259             while (!que.IsEmpty())
260             {
261                 x = GetMinValue(que);
262                 q1.Insert(x);
263             }
264             while (!q1.IsEmpty())
265                 que.Insert(q1.Remove());
266         }
267 
268 
269 
270 
271 
272 
273 
274 
275 
```

```
276         /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
277          *      שימו לב!!!  שיכפול תור מותר אך ורק
278          *                  כשהשאלה דורשת זאת.
279          *              למשל - ספירת כפילויות בתור
280          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
281
282         //--- פעולה המשכפלת את התור ---
283         public static Queue<int> Clone(Queue<int> que)
284         {
285             Queue<int> qTemp = new Queue<int>();     //  תור עזר
286             Queue<int> qCopy = new Queue<int>();     //  התור המשוכפל
287             int x;
288             while (!que.IsEmpty())
289             {
290                 x = que.Remove();
291                 qTemp.Insert(x);
292                 qCopy.Insert(x);
293             }
294             while (!qTemp.IsEmpty())
295                 que.Insert(qTemp.Remove());
296             return qCopy;
297         }
298
299         /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
300          *      בניה של תור חדש
301          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
302         //--- בניית תור מספרים מאיברי מערך ---
303         public static Queue<int> GetQueue(int[] arr)
304         {
305             Queue<int> que = new Queue<int>();
306             for (int i = 0; i < arr.Length; i++)
307                 que.Insert(arr[i]);
308             return que;
309         }
310
311         public static Random rnd = new Random();
312
313         //---       size פעולה המחזירה תור של ---
314         //--- to -ל  from  מספרים אקראיים בין ---
315         public static Queue<int> GetRandomQue(int size,
316                                               int from, int to)
317         {
318             Queue<int> que = new Queue<int>();
319             int x;
320             for (int i = 0; i < size; i++)
321             {
322                 x = rnd.Next(from, to+1);
323                 que.Insert(x);
324             }
325             return que;
326         }
327     }
328 }
329
```