תור

```
// 300.  כתוב פעולה המקבלת תור מסוג מספר שלם, ומחזירה כמה איברים בתור (לא
צריך לשמור על הערכים בתור)
public static int QueueLength(Queue<int> q)
{
    int length = 0;
    while (!q.IsEmpty())
    {
        q.Remove();
        length++;
    }
    return length;
}

// 301.  כתוב פעולה המקבלת תור מסוג מספר שלם, ומחזירה כמה איברים בתור.
צריך לשמור על הערכים בתור
public static int QueueLength2(Queue<int> q)
{
    Queue<int> newQ = new Queue<int>();
    int length = 0;
    while (!q.IsEmpty())
    {
        newQ.Insert(q.Remove());
        length++;
    }
    return length;
}


// 302. כתוב פעולה המקבלת תור מסוג מספר שלם ומחזירה את סכום האיברים
public static int SumOfQueue(Queue<int> q)
{
    int sum = 0;
    while (!q.IsEmpty())
    {
        sum = sum + q.Remove();
    }
    return sum;
}

// 303. כתוב פעולה המקבלת תור מסוג מספר שלם ומספר שלם NUM, ומחזירה האם NUM
נמצא בתור, צריך
// לשמור על הערכים בתור.
public static bool NumInTheQueue(Queue<int> q, int num)
{
    Queue<int> newQ = new Queue<int>();
    while (!q.IsEmpty())
    {
        if (q.Head() == num)
        {
            return true;
        }
        newQ.Insert(q.Remove());
    }
    return false;
}


// 304. כתוב פעולה המקבלת תור מסוג מספר שלם ומספר שלם NUM,  הפעולה תוציא
מהתור את NUM.( המספר
// NUM יכול להופיע יותר מפעם אחת)
```

```
public static void RemoveNumber(Queue<int> q, int num)
{
     Queue<int> newQ = new Queue<int>();
     while (!q.IsEmpty())
     {
         if (q.Head() != num)
         {
             newQ.Insert(q.Remove());
         }

         else
         {
             q.Remove();
         }
     }

     while (!newQ.IsEmpty())
     {
         q.Insert(newQ.Remove());
     }
}

// 305. כתוב פעולה המקבלת תור מסוג מספר שלם, ומוציאה את האיבר האחרון בתור
public static void RemoveLastNumber(Queue<int> q)
{
    Queue<int> newQ = new Queue<int>();
    int candidate; // מחזיק את האיבר שהוצא מהתור בכל פעם

    while (!q.IsEmpty())
    {
        candidate = q.Remove();
        if (!q.IsEmpty())
        {
            newQ.Insert(candidate);
        }
    }
    while(!newQ.IsEmpty())
    {
        q.Insert(newQ.Remove());
    }
}

// 306.  כתוב פעולה המקבלת תור מסוג מספר שלם, ומחזירה תור משוכפל. התור
המקורי ישאר כפי שהיה
public static Queue<int> CloneQueue(Queue<int> q)
{
    Queue<int> newQ = new Queue<int>();
    int size = 0;
    int current;
    while (!q.IsEmpty())
    {
        size++;
        newQ.Insert(q.Remove());
    }

    for (int i = 0; i < size; i++)
    {
        current = newQ.Remove();
        q.Insert(current);
        newQ.Insert(current);
    }
    return newQ;
}
```

```
// 307.   כתוב פעולה המקבלת תור מסוג מספר שלם, ומחזירה האם כל איבר הוא פי
2 מהאיבר הבא בתור אחריו
public static bool twice(Queue<int> q)
{
    int first;
    int second;
    while (!q.IsEmpty())
    {
        first = q.Remove();
        if (!q.IsEmpty())
        {
            second = q.Head();
            if (first * 2 != second)
            {
                return false;
            }
        }
    }
    return true;
}

// 308.   כתוב פעולה המקבלת תור מסוג מספר שלם, ומחזירה האם התור ממוין בסדר
עולה
public static bool according(Queue<int> q)
{
    Queue<int> newQ = new Queue<int>();
    int current;
    while (!q.IsEmpty())
    {
        current = q.Remove();
        if (!q.IsEmpty())
        {
            if (current > q.Head())
            {
                return false;
            }
            newQ.Insert(current);
        }
    }
    return true;
}

// 309. כתוב פעולה המקבלת תור ממויין מסוג מספר שלם ומספר שלם NUM,  הפעולה
תכניס את NUM למקומו לפי
// המיון
public static void InsetAccordingNumber(Queue<int> q, int num)
{
    Queue<int> newQ = new Queue<int>();

    while (!q.IsEmpty())
    {
        if(q.Head() > num)
        {
            newQ.Insert(num);
            break;
        }
        else
        {
            newQ.Insert(q.Remove());
        }

        if (q.IsEmpty())
```

```
        {
            newQ.Insert(num);
        }
    }

    while (!q.IsEmpty())
    {
        newQ.Insert(q.Remove());
    }
    while (!newQ.IsEmpty())
    {
        q.Insert(newQ.Remove());
    }
}

// 309.  ב. לא יודעים אם התור ממוייין בסדר עולה או יורד ואז צריך למצוא את
הסדר( - אפשר להניח שיש לפחות 2
//  איברים בתור, בדי לקבוע אם סדר עולה או יורד
public static void InsetAccordingNumber2(Queue<int> q, int num)
{
    Queue<int> newQ = new Queue<int>();

    int first = q.Remove();
    if (first < q.Head())
    {
        // התור עולה
        if (num < first)
        {
            newQ.Insert(num);
            newQ.Insert(first);
            while (!q.IsEmpty())
            {
                newQ.Insert(q.Remove());
            }
        }
        else
        {
            newQ.Insert(first);
            while (!q.IsEmpty())
            {
                if (q.Head() < num)
                {
                    newQ.Insert(q.Remove());
                }
                else
                {
                    newQ.Insert(num);
                    break;
                }
            }

            while (!q.IsEmpty())
            {
                newQ.Insert(q.Remove());
            }
        }
    }


    else
    {
        // התור יורד
        if (num > first)
```

```
        {
            newQ.Insert(num);
            newQ.Insert(first);
            while (!q.IsEmpty())
            {
                newQ.Insert(q.Remove());
            }
        }
        else
        {
            newQ.Insert(first);
            while (!q.IsEmpty())
            {
                if (q.Head() > num)
                {
                    newQ.Insert(q.Remove());
                }
                else
                {
                    newQ.Insert(num);
                    break;
                }
            }

            while (!q.IsEmpty())
            {
                newQ.Insert(q.Remove());
            }
        }

    }

    while (!newQ.IsEmpty())
    {
        q.Insert(newQ.Remove());
    }
}

// 310. בתוב פעולה המקבלת תור מסוג מספר שלם ומחזירה:
// אם מספר האיברים בתור היא אי זוגית, אז מחזירה את האיבר האמצעי
//  אם מספר האיברים בתור היא זוגית, אז מחזירה את הממוצע של 2 האיברים
האמצעיים.
// 0. אם התור ריק, מחזירה
//יש לשמור על התור
public static double MiddleNumber(Queue<int> q)
{
    Queue<int> newQ = new Queue<int>();
    int size = 0;
    int current;
    while (!q.IsEmpty())
    {
        size++;
        newQ.Insert(q.Remove());
    }

    if (size == 0)
    {
        return 0;
    }

    if (size % 2 == 1)
    {
        for (int i = 0; i < size / 2; i++)
```

```
        {
            q.Insert(newQ.Remove());
        }
        return newQ.Head();
    }

    else
    {
        for (int i = 0; i < size / 2 - 1; i++)
        {
            q.Insert(newQ.Remove());
        }
        return (newQ.Remove() + newQ.Remove()) / 2.0;
    }
}


// 311.   בתוב פעולה המקבלת תור מסוג מספר שלם שבו כל איבר הוא ספרה (בין 0 ל 9 כולל) התור מייצג מספר שלם.באשר ספרת האחדות נמצאת בראש התור, העשרות אחרי הראש וכו'.
//יש להחזיר את המספר במספר שלם.יש לשמור על התור//
public static int NumberFromQueue(Queue<int> q)
{
    Queue<int> newQ = new Queue<int>();
    int multiplier = 1;
    int num = 0;
    while (!q.IsEmpty())
    {
        num += q.Head() * multiplier;
        multiplier *= 10;
        newQ.Insert(q.Remove());
    }
    while (!q.IsEmpty())
    {
        q.Insert(newQ.Remove());
    }

    return num;
}


// 312.  בתוב פעולה המקבלת תור מסוג מספר שלם שבו כל איבר הוא ספרה (בין 0 ל 9 כולל) התור מייצג מספר שלם.באשר ספרת האחדות נמצאת בסוף התור, העשרות לפני הסוף וכו'.
//יש להחזיר את המספר במספר שלם.יש לשמור על התור//
public static int NumberFromQueue2(Queue<int> q)
{
    Queue<int> newQ = new Queue<int>();
    int size = 0;
    int current;
    int num = 0;
    while (!q.IsEmpty())
    {
        size++;
        newQ.Insert(q.Remove());
    }

    for (int i = 0; i < size; i++)
    {
        current = newQ.Remove();
        num = num + current * (int)Math.Pow(10, size - i - 1);
        q.Insert(current);
    }
```

```
    return num;
}


// 313.  כתוב פעולה המקבלת תור מסוג שלם, ומחזירה האם האיברים בחצי הראשון
של התור מסודרים בסדר עולה,
// והאיברים בחצי השני של המחסנית מסודרים בסדר יורד.מספר האיברים בתור הוא
דוגי. יש לשמור על התור
public static bool HalfAscendingHalfDescending(Queue<int> q)
{
    if (q.IsEmpty())
    {
        return false;
    }
    Queue<int> newQ = new Queue<int>();
    int len = 0;
    while (!q.IsEmpty())
    {
        len++;
        newQ.Insert(q.Remove());
    }
    bool flag = true;
    int prev1 = newQ.Remove();
    for (int i = 1; i < len / 2; i++)
    {
        if (prev1 > newQ.Head())
        {
            flag = false;
        }
        q.Insert(prev1);
        prev1 = newQ.Remove();
    }
    q.Insert(prev1);

    int prev2 = newQ.Remove();
    for (int i = 1; i < len / 2; i++)
    {
        if (prev2 < newQ.Head())
        {
            flag = false;
        }
        q.Insert(prev2);
        prev2 = newQ.Remove();
    }
    q.Insert(prev2);

    return flag;
}


// 314.  כתוב פעולה המקבלת תור מסוג מספר שלם, ומחזירה האם סכום החצי הראשון
של התור שווה לסכום החצי
//השני של התור.יש לשמור על התור.
//א.מספר האיברים בתור הוא דוגי
//ב.מספר האיברים בתור הוא אי דוגי, ואז האיבר האמצעי לא נכלל בסכום
public static bool EqualHalfSums(Queue<int> q)
{
    Queue<int> s = new Queue<int>();
    int sumAll = 0, len = 0;
    while (!q.IsEmpty())
    {
        len++;
        sumAll += q.Head();
```

```
        s.Insert(q.Remove());
    }

    int SumFirstHalf = 0;
    for (int i = 0; i < len / 2; i++)
    {
        SumFirstHalf += s.Head();
        q.Insert(s.Remove());
    }
    int middle = 0;
    if (len % 2 != 0)
    {
        middle = s.Head();
    }

    while (!s.IsEmpty())
    {
        q.Insert(s.Remove());
    }

    if (len % 2 == 0)
    {
        return sumAll == 2 * SumFirstHalf;
    }
    return sumAll - middle == 2 * SumFirstHalf;
}


// 315.  כתוב פעולה המקבלת תור מסוג מספר שלם, ומחזירה האם האיברים בחצי
הראשון, נמצאים גם בחצי השני.
// יש לשמור על התור
//א.האיברים מסודרים לפי אותו הסדר
//ב.האיברים לא מסודרים לפי אותו הסדר
// להשלים פה עם ניר


// 316.  כתוב פעולה המקבלת 2 תורים מסוג מספר שלם, ומחזירה תור חדש שבה כל
איבר הוא סכום של 2 איברים
// מ2 התורים.הסכום הוא לפי הסדר.בראש התור החדש יהיה הסכום של 2 האיברים
מראש 2 התורים.מספר
//האיברים יכול להיות שונה.יש לשמור על התורים.
public static Queue<int> SumBoth(Queue<int> q1, Queue<int> q2)
{
    Queue<int> q = new Queue<int>();
    Queue<int> q1Copy = new Queue<int>();
    Queue<int> q2Copy = new Queue<int>();
    int len1 = 0, len2 = 0;
    while (!q1.IsEmpty())
    {
        len1++;
        q1Copy.Insert(q1.Remove());
    }

    while (!q2.IsEmpty())
    {
        len2++;
        q2Copy.Insert(q2.Remove());
    }

    while (len1 > 0 && len2 > 0)
    {
        q.Insert(q2Copy.Head() + q1Copy.Head());
        q2.Insert(q2Copy.Remove());
```

```
        q1.Insert(q1Copy.Remove());
        len1--;
        len2--;
    }

    while (len1 > 0)
    {
        q1.Insert(q1Copy.Head());
        q.Insert(q1Copy.Remove());
        len1--;
    }

    while (len2 > 0)
    {
        q2.Insert(q2Copy.Head());
        q.Insert(q2Copy.Remove());
        len2--;
    }

    return q;
}


// 320.  כתוב פעולה המקבלת תור מסוג מערך של מספר שלם. הפעולה מחזירה את
המערך עם מספר האיברים הכי
//גדול.יש לשמור על התור.
public static int[] BiggestArr(Queue<int[]> q)
{
    if (q.IsEmpty())
    {
        return new int[0];
    }
    int maxLength = q.Head().Length;
    int[] arr = q.Head();
    Queue<int[]> qCopy = new Queue<int[]>();
    while (!q.IsEmpty())
    {
        if (maxLength < q.Head().Length)
        {
            arr = q.Head();
            maxLength = arr.Length;
        }
        qCopy.Insert(q.Remove());
    }

    while (!qCopy.IsEmpty())
    {
        q.Insert(qCopy.Remove());
    }

    return arr;
}

// 321.  כתוב פעולה המקבלת תור מסוג מערך של מספר שלם. הפעולה מחזירה את
המספר הכי גדול שמופיע בכל
//המערכים.יש לשמור על התור.
public static int BiggestNum(Queue<int[]> q)
{
    Queue<int[]> qC = new Queue<int[]>();
    int max = q.Head()[0]; // הנחה שהמערך הראשון אינו ריק
    while (!q.IsEmpty())
    {
        for (int i = 0; i < q.Head().Length; i++)
```

```
        {
            if (q.Head()[i] > max)
            {
                max = q.Head()[i];
            }
        }
        qC.Insert(q.Remove());
    }

    while (qC.IsEmpty())
    {
        q.Insert(qC.Remove());
    }

    return max;
}


// 322.  בתוב פעולה המקבלת תור מסוג תור של מספר שלמים. יש להחדיר את סכום
בל המספרים בבל התורים. יש
// לשמור על שלמות בל התורים.
public static int SUMALL(Queue<Queue<int>> q)
{
    Queue<Queue<int>> qC = new Queue<Queue<int>>();
    int sum = 0;
    while (!q.IsEmpty())
    {
        Queue<int> qINqC = new Queue<int>();
        while (!q.Head().IsEmpty())
        {
            sum += q.Head().Head();
            qINqC.Insert(q.Head().Remove());
        }

        while (!qINqC.IsEmpty())
        {
            q.Head().Insert(qINqC.Remove());
        }

        qC.Insert(q.Remove());
    }

    while (!qC.IsEmpty())
    {
        q.Insert(qC.Remove());
    }

    return sum;
}


// 323. בתוב פעולה המקבלת מערך מסוג תור מסוג מספר שלם. הפעולה תוציא ותחדיר
)return( את המספר הבי
//גדול שנמצא בראש התורים.יתבנו מחסניות ריקות.
public static int RemoveMax(Queue<int>[] arr)
{
    if (arr.Length == 0)
    {
        return 0;
    }
    Queue<int> ContainsMax = arr[0];
    int lenMax = 0;
```

```
    int max = arr[0].Head();

    for (int i = 0; i < arr.Length; i++)
    {
        int len = 0;
        bool FoundMax = false;
        Queue<int> qC = new Queue<int>();
        while (!arr[i].IsEmpty())
        {
            len++;
            if (arr[i].Head() > max)
            {
                FoundMax = true;
                ContainsMax = arr[i];
                max = arr[i].Head();
            }
            qC.Insert(arr[i].Remove());
        }
        if (FoundMax)
        {
            lenMax = len;
        }

        while (!qC.IsEmpty())
        {
            arr[i].Insert(qC.Remove());
        }
    }

    if (lenMax == 0)
    {
        return arr[0].Remove(); // Is the max if nothing has changed
    }
    bool HasRemoved = false;
    for (int i = 0; i < lenMax - 1; i++)
    {
        if (!HasRemoved && ContainsMax.Head() == max)
        {
            ContainsMax.Remove();
            HasRemoved = true;
        }

        ContainsMax.Insert(ContainsMax.Remove());
    }

    if (!HasRemoved)
    {
        ContainsMax.Remove();
    }


    return max;
}
```