שאלה 5 (22 נק')

נאמר שמערך a הינו מערך **ממויין-בדילוגי-** m אם כל איבר $a[i]$ במערך קטן או שווה לכל האיברים $a[i+l\cdot m]$, באשר $l\in\mathbb{N}$. לדוגמה, המערך

7	2	1	5	7	6	3	6	8	9	5	8

הינו מערך ממוין-בדילוגי-4 (סדרת האיברים ברקע הלבן ממוינת, כנ"ל סדרת האיברים ברקע השחור וכו').
בשאלה זו נניח כי אורכי המערכים וערכי m הינם חזקות שלמות של 2.

סעיף א'

ממשו את הפונקציה `mreduce()` המקבלת את הפרמטרים הבאים:

`a`	מערך ממויין-בדילוגי- m .
`aux`	מערך-עזר באותו אורך כמו `a`. ניתן להשתמש בו כרצונכם ואין מגבלה על הערכים שבו בתום ביצוע הפונקציה.
`n`	אורך המערכים `a` ו-`aux`; ניתן להניח שזוהי חזקה של 2.
`m`	גודל ה'דילוג' של הממויינות במערך `a`; ניתן להניח שזוהי חזקה של 2.

והופכת את `a` ממערך ממויין-בדילוגי- m למערך ממויין-בדילוגי- $m/2$.

דוגמה:

המערך לעיל עשוי להפוך לאחר ריצת הפונקציה למערך הבא (זוהי אחת האפשרויות):

1	2	3	5	5	6	7	6	7	8	8	9

אשר בו כל רצף של איברים בדילוגים של 2 מאחד לשני הינו רצף ממויין.

דרישות סיבוכיות: $O(n)$ זמן.

פתרון אשר אינו עונה על דרישות הסיבוכיות הנ"ל לא יזכה בניקוד.

סעיף ב'

כתבו פונקציה `msort()` המשתמשת בפונקציה `mreduce()` מסעיף א' לצורך מיון מערך (אתם רשאים להשתמש בה גם אם לא עניתם על סעיף א').

על הפונקציה לקבל את הארגומנטים הבאים:

`a`	מערך ממויין-בדילוגי- m .
`aux`	מערך-עזר באותו אורך כמו `a`. ניתן להשתמש בו כרצונכן ואין מגבלה על הערכים שבו בתום ביצוע הפונקציה.
`n`	אורך המערכים `a` ו-`aux`; ניתן להניח שזוהי חזקה של 2.
`m`	גודל ה'דילוג' של הממויינות במערך `a`; ניתן להניח שזוהי חזקה של 2.

דרישות סיבוכיות: $O(n\cdot\log(m))$ זמן.

פתרון אשר אינו עונה על דרישות הסיבוכיות הנ"ל לא יזכה בניקוד.

סעיף א'

```
int mreduce(int arr[], int aux[], int n, int m)
{
  int i,j,k_1,k_2,k_aux;

  for(i=0;i<m/2;i+=1) {

    /* merge the elements with index i modulu m and index
     * i+m/2 modulo m from arr[] into a single sorted
     * sequence in aux[]
     */
    k_1 = i;
    k_2 = i + m/2;
    k_aux = 0;
    while((k_1 < n) && (k_2 < n)) {
      if (arr[k_1] <= arr[k_2]) {
        aux[k_aux++] = arr[k_1];
        k_1 += m;
      }
      else  {
        aux[k_aux++] = arr[k_2];
        k_2 += m;
      }
    }
    while(k_1 < n) {
      aux[k_aux++] = arr[k_1];
      k_1 += m;
    }
    while(k_2 < n) {
      aux[k_aux++] = arr[k_2];
      k_2 += m;
    }

    /* copy the merged sequence back into the arr[] cells
     * whose index modulo m/2 is i
     */
    for(j=0; j<(n/m)*2; j++)
      arr[m/2*j+i] = aux[j];
  }
}
```

סעיף ב'

```
void msort(int arr[], int aux[], int n, int m)
{
  while (m > 1) {
    mreduce(arr,aux,n,m);
    m /= 2;
  }
  /* now arr[] is 1-sorted, i.e. properly sorted */
}
```