מטלת מנחה (ממ"ן) 14

הקורס: 20441 - מבוא למדעי המחשב ושפת Java

חומר הלימוד למטלה: יחידות 9-10 **נושא המטלה:** יעילות ורקורסיה

מספר השאלות: 4 **משקל המטלה:** 5 נקודות

סמסטר: 2020א **מועד אחרון להגשה:** 1.2.2020

השאלות במטלה זו לקוחות מבחינות גמר שונות או דומות לשאלות של בחינות גמר. חלק מהשאלות הן לתרגול בלבד ולא להגשה. אנו ממליצים מאד לענות עליהן ללא הרצה במחשב (כפי שמקובל בבחינת הגמר).

את התשובות לכל השאלות עליכם לכתוב במחלקה אחת בשם Ex14.java (בדיוק).

את התשובות לשאלות על הסיבוכיות כתבו כחלק מה- API.

שאלה 1 – 25 נקודות

סעיף א (8%)

כתבו שיטה שחתימה היא

```
public static int subStrC(String s, char c)
```

המקבלת מחרוזת s ותו c ומחזירה כמה תת-מחרוזות יש ב- s שמתחילות ומסתיימות בתו c ויש **בתוכן** בדיוק פעם אחת את התו c.

לדוגמא, אם המחרוזת היא "s = "abcbcabcacabcc והתו 'c = 'c התוצאה היא 4. (כי התת-מחרוזות הן "cbcabc", "cabcac", "cacabc", "cabcc")

שימו לב שאינכם צריכים להחזיר את התת-מחרוזות, אלא רק את מספרן.

סעיף ב. (17%)

כתבו שיטה שחתימה היא

```
public static int subStrMaxC(String s, char c, int k)
```

המקבלת מחרוזת s, תו c ומספר שלם חיובי k, ומחזירה כמה תת-מחרוזות יש ב- s שמתחילות ומסתיימות בתו c ויש **בתוכן** מקסימום k פעמים את התו c.

שימו לב שאינכם צריכים להחזיר את התת-מחרוזות, אלא רק את מספרן.

דוגמאות:

- אם המחרוזת היא "s = "abcbc, התו 'c = 'c והמספר k=0 התוצאה היא 1. (התת-מחרוזת היא "cbc")
- אם המחרוזת היא "s = "abcbcabcacab, התו 'c = 'c והמספר k=2 התוצאה היא 6. (התת-מחרוזות הן "cbcabcac", "cabcac", "cbcabc", "cac", "cabc", "cbc")

- אם המחרוזת היא "s = "abcbcabcacab, התו c = 'c' והמספר k=3 התוצאה היא שוב 6. (שכן התת-מחרוזות הן "cbc", "cabc", "cac", "cbcabc", "cabcac", "cbcabcac", ואין אף תת-מחרוזת המתחילה בתו 'c' ומסתיימת בתו 'c' ויש בתוכה 3 תווי 'c')
- אם המחרוזת היא "s = "abc, התו c = 'c' והמספר k=2 התוצאה היא 0, שכן אין אף תת-מחרוזת שמתחילה ומסתיימת ב- 'c'.

השיטות שתכתבו צריכות להיות יעילות ככל הניתן, גם מבחינת סיבוכיות הזמן וגם מבחינת סיבוכיות המקום. תשובה שאינה יעילה מספיק כלומר, שתהיה בסיבוכיות גדולה יותר מזו הנדרשת לפתרון הבעיה תקבל מעט נקודות בלבד.

ניתן להשתמש בשיטות עזר ככל הנדרש. **בחישוב הסיבוכיות צריך לחשב גם את הזמן והמקום של שיטות העזר.**

אל תשכחו לתעד את השיטות שכתבתם.

כתבו מה סיבוכיות הזמן וסיבוכיות המקום של השיטות שכתבתם.

שימו לב, בפתרון הבעיות מותר להשתמש אך ורק בשיטות שלהלן (המוגדרות במחלקה String), ובפקודות השוואה. (אין חובה להשתמש בהן!)

- `public char charAt(int i)` - המחזירה את התו במקום ה- i במחרוזת (עליה היא מופעלת)
- `public int length()` - המחזירה את אורך המחרוזת עליה היא מופעלת.

שאלה 2 - 25 נקודות (להגשה)

נתון מערך חד-ממדי a המכיל ערכי 0 ו- 1 בלבד.

לדוגמא:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1

כתבו שיטה סטטית המקבלת כפרמטר מערך a כנ"ל, ומחליפה כל ערך במערך **שאינו אפס** במרחק ממנו לנקודת האפס הקרובה ביותר (מימין או משמאל). שימו לב שב"מרחק" הכוונה פשוט למספר הצעדים במערך שיש לבצע על מנת להגיע לאיבר אפס הקרוב ביותר.

לדוגמא, עבור המערך a שלעיל, המערך a לאחר ההחלפה (כך שיכיל את המרחקים) יהיה:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	3	2	1	0	1	2	3	2	1	0	1	2

ניתן להניח שהמערך מכיל לפחות 0 אחד. כלומר, הוא אינו מכיל רק 1-ים.

חתימת השיטה היא:

```
public static void zeroDistance (int [] a)
```

שימו לב:

השיטה שתכתבו צריכה להיות יעילה ככל הניתן, גם מבחינת סיבוכיות הזמן וגם מבחינת סיבוכיות המקום. תשובה שאינה יעילה מספיק כלומר, שתהיה בסיבוכיות גדולה יותר מזו הנדרשת לפתרון הבעיה תקבל מעט נקודות בלבד.

כתבו מה סיבוכיות הזמן וסיבוכיות המקום של השיטה שכתבתם.

אל תשכחו לתעד את מה שכתבתם!

שאלה 3- 25 נקודות

נתונה מחרוזת תוים s. במהלך שליחתה ברשת האינטרנט קרתה תקלה וחלק מהתווים במחרוזת הוכפלו והופיעו ברצף מספר לא ידוע של פעמים. כלומר במחרוזת t שהתקבלה הופיעו כל התווים של s, לפי הסדר במחרוזת המקורית, אבל לפעמים תו מסוים לא הופיע פעם אחת כמו במחרוזת s אלא ברצף מספר לא ידוע של פעמים.

נגדיר שמחרוזת תוים t **עברה-טרנספורמציה** ממחרוזת התווים s , אם כל תו של s נמצא ב- t (לפי הסדר ב- s) **לפחות** פעם אחת.

שימו לב שאם תו מופיע במחרוזת s כמה פעמים (נניח k), אז במחרוזת t שעברה טרנספורמציה הוא יופיע לפחות k פעמים.

לדוגמא:

אם המחרוזת "abbcd" = s

אז כל המחרוזות הבאות **עברו-טרנספורמציה** ממחרוזת זו:

"abbcd" , "aaaabbcd" , "abbcddddd" , "aabbccdd" , "abbbccd"

וכל המחרוזות הבאות **לא עברו-טרנספורמציה** מהמחרוזת s:

"a" , "abcd" , "aaccbbdd"

כתבו שיטה **סטטית** בוליאנית **רקורסיבית** שחתימתה היא:

```
public static boolean isTrans (String s, String t)
```

המקבלת כפרמטרים שתי מחרוזות תווים s ו- t.

השיטה צריכה להחזיר true אם t עברה-טרנספורמציה מהמחרוזת s, ו- false אחרת.

השיטה צריכה להיות רקורסיבית ללא שימוש בלולאות כלל. כך גם כל שיטות העזר שתכתבו (אם תכתבו) לא יכולות להכיל לולאות.

אפשר להשתמש בהעמסת-יתר (overloading).

אין צורך לדאוג ליעילות השיטה, אבל כמובן שצריך לשים לב לא לעשות קריאות רקורסיביות מיותרות!

אל תשכחו לתעד את מה שכתבתם!

שאלה 4- 25 נקודות

נתון מערך דו-ממדי של מספרים שלמים אי-שליליים קטנים מ- 100.

נגדיר **מסלול חוקי** במערך כסדרה של תאים במערך, המתחילה בתא הראשון (שורה 0 ועמודה 0) ומתקדמת במערך לפי ספרת האחדות והעשרות של המספר שבתא, עד לתא האחרון במערך (שורה אחרונה ועמודה אחרונה).

לדוגמא, אם בתא [3][2] נמצא המספר 15, אז מהתא הזה אפשר להתקדם:

- או +1 בשורות ו- +5 בעמודות, כלומר לתא [3][8]
- או +5 בשורות ו- +1 בעמודות, כלומר לתא [7][4]

שימו לב שאי אפשר להתקדם מעבר לגבולות המערך.

כתבו שיטה **רקורסיבית** countPaths אשר מקבלת מערך דו-ממדי כנ"ל , ומחזירה את מספר המסלולים החוקיים האפשריים במערך.

חתימת השיטה היא: public static int countPaths (int [][] mat)

לדוגמא,

אם נריץ את השיטה על המערך להלן:

	0	1	2	3
0	**12**	22	23	54
1	43	35	**21**	20
2	34	21	43	21
3	25	30	0	**20**
4	0	22	10	10
5	20	13	3	**45**

השיטה תחזיר את המספר 3, שכן יש שלושה מסלולים חוקיים (הראשון מסומן במערך):

```
1. [0][0]-> [1][2]-> [3][3]-> [5][3]
2. [0][0]-> [2][1]-> [3][3]-> [5][3]
3. [0][0]-> [2][1]-> [4][2]-> [4][3]-> [5][3]
```

השיטה שתכתבו צריכה להיות רקורסיבית, ללא שימוש בלולאות בכלל.

אפשר להשתמש בהעמסת-יתר (overloading), אבל לא בשיטות עזר אחרות.

אפשר להניח שהמערך מלא במספרים שלמים אי-שליליים קטנים מ- 100, אין צורך לבדוק זאת.

אסור לשנות את המערך mat במהלך השיטה.

אין צורך לדאוג ליעילות השיטה, אבל כמובן שצריך לשים לב לא לעשות קריאות רקורסיביות מיותרות!

שימו לב:

בשאלות 3 ו- 4 אין צורך לדאוג ליעילות השיטה שתכתבו!

בכל השאלות - אל תשכחו לתעד את מה שכתבתם!

שימו לב ששמנו טסטר באתר הקורס .חובה שטסטר ירוץ ללא שגיאות קומפילציה עם המחלקה שלכם .אם יש שיטה שלא כתבתם, כתבו חתימה והחזירו ערך סתמי כדי שהטסטרים ירוצו עם המחלקות ללא שגיאות קומפילציה. אם הטסטר לא ירוץ ללא שגיאות קומפילציה הציון במטלה יהיה אפס.

הגשה

1. הגשת הממ״ן נעשית בצורה אלקטרונית בלבד, דרך מערכת שליחת המטלות.
2. הקפידו ששמות השיטות יהיו **בדיוק** כפי שמוגדר בממ״ן.
3. עליכם לתעד את כל השיטות שאתם כותבים בתיעוד API ובתיעוד פנימי המסביר מה עשיתם בשיטה. בתיעוד זה כתבו גם מה הסיבוכיות של השיטות (בשאלות 1 ו-2).
4. את התשובות לכל השאלות עליכם לכתוב במחלקה אחת בשם Ex14.java (**בדיוק**). ארזו אותו בתוך קובץ zip. אין לשלוח קבצים נוספים.

בהצלחה

שאלה לא להגשה

לפניכם שני קטעי הקוד (שאינם קשורים זה לזה):

```
int a =3;
while (a <= n)
   a = a*a;
```

```
public void foo (int n, int m)
{
   int i = m;
   while (i > 100)
         i = i/3;
   for (int k=i ; k>=0; k--)
   {
         for (int j=1; j<n; j*=2)
               System.out.print(k + "\t" + j);
         System.out.println();
   }
}
```

מה סיבוכיות זמן הריצה של קטעי הקוד האלו?

להזכירכם – חוקי הלוגריתמים:

$$\log_a m \times n = \log_a m + \log_a n$$

$$\log_a {}^{m}/_{n} = \log_a m - \log_a n$$

$$\log_a n^m = m \times \log_a n$$

שאלה לא להגשה

לפניכם קטע הקוד הבא:

```
public static int foo (int a, int b)
{
   if (a>3)
       return 2 + foo (b-1, a+1);
   if (b<=4)
       return 1 + foo (a-1, b+1);
   return 0;
}
```

לכל אחת מהקריאות הבאות לשיטה foo, ענו אם היא תעצור, ואם כן, מה היא תחזיר.

א. foo (3, 4)

ב. foo (4, 5)

שאלה לא להגשה

התבוננו בשיטות הבאות:

```
public static void f(int [][] a,
                     int a1, int b1, int a2, int b2)
{
     int temp = a[a1][b1] ;
     a[a1][b1] = a[a2][b2] ;
     a[a2][b2] = temp ;
     if (b1 < a[0].length-1)
         f(a, a1, b1+1, a2, b2-1) ;
     else if (a1+1 < a2-1)
         f(a, a1+1, 0, a2-1, a[0].length-1) ;
}

public static void printArray(int[][] a)
{
    for (int i= 0; i< a.length; i++)
    {
        for (int j= 0; j< a[i].length; j++)
         System.out.print (a[i][j] + "\t");
       System.out.println();
     }
 }
```

נניח שנתונה השיטה main הבאה:

```
public static void main (String [] args)
    {
        int[][] arr = {{1, 2, 3, 4}, {5, 6, 7, 8}} ;
        f(arr, 0, 0, arr.length-1, arr[0].length-1) ;
        printArray (arr);
    }
```

1. מה הפלט שתפיק השיטה main?
2. כמה קריאות רקורסיביות מתבצעות בזימון

```
f(arr, 0, 0, arr.length-1, arr[0].length-1) ;
```