מטלת מנחה (ממ"ן) 13

הקורס: 20441 - מבוא למדעי המחשב ושפת Java

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

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

סמסטר: 2023ב **מועד אחרון להגשה:** 10.6.2023

השאלות במטלה זו לקוחות מבחינות גמר שונות או דומות לשאלות של בחינות גמר. אנו ממליצים מאד לענות עליהן ללא הרצה במחשב (כפי שמקובל בבחינת הגמר) ולאחר מכן להריץ.

את התשובות לכל השאלות עליכם לכתוב במחלקה אחת בשם Ex13 (בדיוק).

את התשובות לשאלות על הסיבוכיות כתבו (באנגלית בלבד) **כחלק מה- API של השאלה הרלוונטית.**

שאלה 1 - 25 נקודות

נתונים שני מערכים חד-ממדיים road1 ו-road2 מלאים במספרים שלמים **חיוביים** באורך N.
כל מערך מתאר נסיעה בכביש הבנוי מ-N מקטעים, כך שכל איבר i במערך שווה למספר הדקות הנדרשות לעבור מקטע i של הכביש. שני הכבישים מתחילים במקום A ומסיימים במקום B, ובכל מקטע אפשר לעבור מכביש אחד למשנהו, בלי תוספת זמן לנסיעה.

נהג **מתחיל מכביש מסוים (אחד מהשניים),** ~~כלומר עובר בו לפחות מקטע אחד~~, ומותר לו לעבור לכביש השני **רק פעם אחת לכל היותר.** כלומר, יתכן שהוא מתחיל בכביש 1 ונוסע עד סופו בלי לעבור בכלל לכביש 2, יתכן שהוא מתחיל בכביש 2 ונוסע עד סופו בלי לעבור בכלל לכביש 1 ויתכן שהוא מתחיל בכביש אחד (1 או 2) ועובר פעם אחת לכביש השני (2 או 1).

כתבו שיטה סטטית, המקבלת כפרמטרים את שני המערכים, ומחזירה את מספר הדקות במסלול המהיר ביותר שהנהג יכול לנסוע.

לדוגמא, אם נתונים המערכים הבאים:

	0	1	2	3	4	5	6	7
road1 =	5	4	5	8	12	9	9	3
road2 =	7	3	3	12	10	2	10	7

המסלול המהיר ביותר הוא זה הצבוע באפור. הנהג נוסע שישה מקטעים בכביש 2 ואז עובר לכביש 1 ונוסע בו את שני המקטעים האחרונים.

השיטה תחזיר את הערך 49 שהוא הזמן של המסלול המהיר ביותר.

למשל, אם הנהג היה נוסע כל הזמן בכביש 1, ולא עובר לכביש 2 בכלל, זמן הנסיעה היה 55.

אם הנהג היה נוסע כל הזמן בכביש 2, ולא עובר לכביש 1 בכלל, זמן הנסיעה היה 54. אם היה מתחיל בכביש 1 ועובר לכביש 2 אחרי 4 מקטעים, זמן הנסיעה היה 51, וכן הלאה.

חתימת השיטה היא:

```
public static int shortestRoad (int [] road1, int [] road2)
```

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.

ניתן להשתמש בשיטות עזר ככל הנדרש. בחישוב הסיבוכיות צריך לחשב גם את הזמן והמקום של שיטות העזר. אסור לשנות את המערכים!

כתבו (באנגלית בלבד) כחלק מה- API של השאלה מה סיבוכיות הזמן (Time complexity) וסיבוכיות המקום (Space complexity) של השיטה שכתבתם. הסבירו תשובתכם.

אל תשכחו לתעד את מה שכתבתם!

שאלה 2 - 25 נקודות

סדרה חשבונית (arithmetic sequence) היא סדרה של מספרים ,שבה ההפרש בין כל שני איברים עוקבים הוא קבוע.

לדוגמה: בסדרה 3, 5, 7, 9, 11, … : ההפרש בין כל שני איברים עוקבים הוא קבוע – 2.

נתון מערך חד-ממדי arr מלא במספרים שלמים המהווים סדרה חשבונית. אבל במערך נפלה טעות, ואחד המספרים של הסדרה החשבונית חסר.

כתבו שיטה סטטית, המקבלת מערך arr חד-ממדי המכיל סדרה חשבונית, ומחזיר מיהו האיבר החסר.

אפשר להניח שגודל המערך הוא לפחות 2.

לדוגמא, במערך arr הבא :

0	1	2	3	4	5	6	7	8
-2	1	4	7	10	13	16	22	25

חסר המספר 19. לכן השיטה תחזיר את הערך 19.

חתימת השיטה היא:

```
public static int missingValue (int [] arr)
```

אפשר להניח שמערך arr אכן מכיל סדרה חשבונית **ובדיוק** מספר אחד חסר בסדרה. אין צורך לבדוק זאת.

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.

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

אסור לשנות את המערך!

כתבו (באנגלית בלבד) כחלק מה- API של השאלה מה סיבוכיות הזמן (Time complexity) וסיבוכיות המקום (Space complexity) של השיטה שכתבתם. הסבירו תשובתכם.

אל תשכחו לתעד (באנגלית) את מה שכתבתם!

שאלה 3 - 25 נקודות

פָּלִינְדְרוֹם (Palindrome) הוא מילה, מספר, משפט או כל רצף סמלים אחר שקריאתו מימין לשמאל ומשמאל לימין היא זהה.

לדוגמא: מחרוזות התווים "aba" , "1221" הן פלינדרום.

במערך חד-ממדי המכיל מספרים שלמים, נגדיר **רצף פלינדרומי** כסדרה של תאים רצופים במערך המהווים פלינדרום.

לדוגמא:

במערך arr הבא:

0	1	2	3	4	5	6	7	8
1	3	2	3	10	10	3	2	4

ישנם ארבעה רצפים פלינדרומיים:

1. בין האינדקסים 4 ל- 5 {10, 10} אורכו 2
2. בין האינדקסים 1 ל- 3 {3, 2, 3} אורכו 3
3. בין האינדקסים 3 ל- 6 {3, 10, 10, 3} אורכו 4
4. בין האינדקסים 2 ל- 7 {2, 3, 10, 10, 3, 2} אורכו 6

ועוד תשעה רצפים פלינדרומיים, כל אחד מהם באורך 1, שהם כל אחד מהתאים.

כתבו שיטה **סטטית רקורסיבית**, המקבלת כפרמטר מערך חד-ממדי arr המכיל מספרים שלמים ומחזירה את אורכו של הרצף הפלינדרומי הגדול ביותר.

לדוגמא, במערך arr שלעיל, הרצף הפלינדרומי הגדול ביותר הוא בין האינדקסים 2 ל- 7, ואורכו הוא 6. לכן, השיטה צריכה להחזיר את הערך 6.

על המערך arr = {1, 2, 3, 4} השיטה תחזיר 1.

חתימת השיטה היא:

```
public static int longestPalindrome (int[] arr)
```

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!

אסור לשנות את המערך!

אסור להשתמש במשתנים סטטיים (גלובליים)!

שאלה 4 - 25 נקודות

נתון מערך חד-ממדי arr המלא במספרים שלמים חיוביים, ומספר שלם ~~חיובי~~ num.

כתבו שיטה סטטית בוליאנית רקורסיבית, המקבלת כפרמטרים את המערך arr ואת המספר num. השיטה צריכה להחזיר true אם קיימת תת-קבוצה של איברי המערך, שסכום האיברים שלה שווה ל- num, שמקיימת את התנאים הבאים:

1. כוללת איברי המערך ללא חזרות. כלומר, אסור לקחת איבר מהמערך יותר מפעם אחת.
2. **אינה** כוללת שלושה איברים סמוכים במערך. כלומר, אסור להכניס לתת-קבוצה גם את האיבר באינדקס i, גם את האיבר באינדקס i+1 וגם את האיבר באינדקס i+2, **לכל i.**

אם אין תת-קבוצה במערך שמקיימת את התנאים האלו, השיטה תחזיר את הערך false.

לדוגמא,

אם המערך הוא arr = {5, 4, 2, 1, 3}

אז:

- אם num = 0 , השיטה תחזיר true (תת-קבוצה ריקה).
- אם num = 8, השיטה תחזיר true (המספרים 4 + 1 + 3 = 8 או המספרים 5 + 3 = 8).
- אם num = 9, השיטה תחזיר true (המספרים 5 + 4 = 9 אמנם הם רצופים במערך, אבל זו לא שלשה, אלא רק זוג מספרים רצופים).
- אם num = 2, השיטה תחזיר true (המספר 2).
- אם num = 11, השיטה תחזיר false (5+4+2 = 11, אבל הם שלשה רצופה, וגם 5+2+1+3 יש שלשה רצופה).
- אם num = 17 , השיטה תחזיר false (אין במערך מספרים שסכומם 17).

חתימת השיטה היא:

```
public static boolean isSum (int[] arr, int num)
```

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!

אסור להשתמש במשתנים סטטיים (גלובליים)!

אסור לשנות את המערך!

שימו לב:

בכל השאלות - אל תשכחו לתעד (באנגלית בלבד) 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 ללא אפשרות ערעור.

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 **(באנגלית בלבד)** 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 Ex13 (**בדיוק**). ארזו את הקובץ 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) ;
```