


***** נא לקרוא הנחיות לביצוע המבחן בתשומת לב בכדי להימנע מהורדת ניקוד מיותרת *****

- במבחן נעסוק במטריצות (מערך דו-ממדי) ריבועיות של מספרים טבעיים (שלמים חיוביים).
1. לאחר ההזדהות במחשב – תמצאו את התיקיה של חומרי המבחן. בתיקיה תמצאו:
 - עותק של שאלון הבחינה (אותו אתם קוראים כעת... 😊)
 - עותק של דפי העזר העומדים לרשותכם בעת המבחן (יש לכם גם עותק מודפס)
 - 2 קבצים טקסטואליים בשם: mat1.txt , mat2.txt
 ■ קבצים אלה הינם קבצים לדוגמה לשימוש בשאלה 1 (וכן שימושיים לצורך בדיקות של שאלות 2-6).
 2. בהתאם להנחיות הטכנאי המלווה את הכיתה, יש להגדיר תיקייה של workspace למבחן. שם התיקיה הוא: ID_XXXXXXXXXX (כאשר XXXXXXXXXXXX זו תעודת הזהות שלך) בדיסק המיועד לכך. שימו לב: במחשב יש 2 גרסאות של Eclipse, השתמשו בגרסה הרשומה לכם כאן משמאל.
 

Eclipse IDE for Enterprise Java and Web Developers - 2022-06
Desktop app
 3. יש להקים פרויקט בשם ID_XXXXXXXXXX (עם תעודת הזהות שלך)
 - לנוחיות עבודה שלך, מומלץ להעתיק את קבצי הטקסט לתיקיית הפרויקט לעבודה נוחה עם קבצים
 4. פתרון המבחן יכול 2 מחלקות בלבד (!) – כלומר 2 קבצי java בלבד:
 - מחלקה המכילה את הפתרונות לשאלות 1-5. לצורך בדיקות שלכם, מומלץ ורצוי לכתוב גם main ו/או JUnit לבדיקת הפונקציות בשאלות 1-5. זה יעזור לכם משמעותית בבדיקות שלכם!
 - מחלקה המכילה פתרון של שאלה 6
 5. ***** חשוב מאוד ***** שמות הפונקציות בפתרון נדרשות להיות בהתאם לחתימת הפונקציות הרשומה לכם בשאלות (לנוחיות שלכם – ניתן לבצע העתק/הדבק של שם הפונקציה מעותק שאלון המבחן הנמצא בתיקיית חומרי המבחן).
 6. הגשת הפתרון:
 - יש להגיש **רק** את שני קבצי ה java של הפתרון (**לא** של כל הפרויקט **ולא את** תיקיית ה src)
 - יש לבצע zip של שני קבצי ה java (בלבד!) לתוך קובץ בשם ID_XXXXXXXXX.zip
 - בתחילת כל קובץ הוסיפו הערה עם מספר ת"ז שלכם
 - בהתאם להנחיות הטכנאי במבחן – יש לבצע הגשה של קובץ הפתרון (קובץ ה zip)
 - **שימו לב:** זו אחריות שלכם בלבד לבצע הגשה נכונה של הקבצים. הטכנאי המלווה את המבחן אינו אחראי על תוכן הקובץ אותו אתם מגישים.
- *** אי הקפדה על הנחיות אלה תגרור הורדת ניקוד ואף עלולה למנוע אפשרות בדיקת המבחן *****

שאלה 1 (15 נקודות):

נתונה חתימת הפונקציה הבאה:

```
public static int[][] readFromFile(String fileName)
    throws FileNotFoundException
```

הפונקציה מקבלת מחרוזת המכילה **שם קובץ** וקוראת מהקובץ נתוני מטריצה. הפונקציה מחזירה מערך דו-ממדי ריבועי המכיל את המידע שנקרא מהקובץ.

מבנה קובץ הקלט (לדוגמה משמאל):

1	4			
2	1	2	3	4
3	3	1	3	3
4	3	1	3	3
5	1	2	3	4

השורה הראשונה בקובץ מכילה מספר יחיד X המיצג את ממדי המטריצה (אורך ורוחב).

לאחר מכן יהיה X שורות, כשבכל שורה יש X מספרים טבעיים מופרדים ברוחב

כתבו את גוף הפונקציה כך שתקרא את נתוני הקובץ ותחזיר מטריצה מלאה עם ערכים בהתאמה לנתוני הקובץ.

עבור הקובץ הקלט בדוגמה הנתונה, תתקבל המטריצה הבאה:

	0	1	2	3
0	1	2	3	4
1	3	1	3	3
2	3	1	3	3
3	1	2	3	4

אין צורך לבדוק את נכונות/תקינות הקלט!

שאלה 2 (15 נקודות):

נתונה חתימה הפונקציה הבאה:

```
public static void transformMatrix(int[][] matrix)
```

הפונקציה מקבלת מטריצה ריבועית, והופכת אותה: שורות לטורים (וטורים לשורות) – כמו באלגברה לינארית.

הנחיה חשובה ומהותית: יש לבצע את פעולת ההפיכה **ללא הקצאה של מטריצה חדשה**, כלומר, אין לייצר מערך חדש, אלא לבצע את משימת החלפת השורות/טורים במטריצה המתקבלת כפרמטר.

עבור המטריצה בקובץ הקלט בדוגמה של שאלה 1, נקבל את המטריצה הבאה (ראו דוגמה משמאל):

	0	1	2	3
0	1	3	3	1
1	2	1	1	2
2	3	3	3	3
3	4	3	3	4

רצוי ומומלץ לחשוב על מודולריות ולממש פונקציות עזר.

שאלה 3 (15 נקודות):

שתי שורות במטריצה נחשבות זהות אם הם מכילות את אותם מספרים, ובדיוק באותו סדר. לדוגמה: שורה 0 ושורה 3 בקובץ של המטריצה בקובץ הנתון (בהתאם לדוגמה בשאלה 1).

מטריצה תיקרא **שווה דו צדדית** אם:

- השורה הראשונה זהה לשורה האחרונה
- השורה השנייה זהה לשורה לפני אחרונה
- וכן הלאה: השורה ה- i זהה לשורה ה- $n-i$ (למטריצה בגודל n)
- רק אם **כל** זוגות השורות הללו זהות, זו מטריצה **שווה דו צדדית**

המטריצה בדוגמה של שאלה 1 – היא מטריצה **שווה דו צדדית**

המטריצה בדוגמה של שאלה 5 (בהמשך) – **אינה** מטריצה **שווה דו צדדית**

נתונה חתימה הפונקציה הבאה:

```
public static boolean doubleSide(int[][] matrix).
```

כתבו קוד שבודק האם מטריצה היא **שווה דו צדדית**.

רצוי ומומלץ לחשוב על מודולריות ולממש פונקציות עזר.

שאלה 4 (20 נקודות):

פתרו את שאלה 3 שוב – הפעם יש לממש את הפתרון, על כל חלקיו, כפתרון רקורסיבי.

שימו לב: יש לשים לב לשני ממדים של רקורסיה: במעברי השורות, ובמעברי העמודות. לכל אורך הפתרון בשאלה זו, **אין להשתמש** בלולאה מכל סוג שהוא!

רצוי ומומלץ לחשוב על מודולריות ולממש פונקציות עזר.

חתימת הפונקציה הרקורסיבית הינה:

```
public static boolean recDoubleSide(int[][] matrix)
```

שאלה 5 (20 נקודות):

נגדיר את המונח "שורת k" במטריצה באופן הבא:

- k מייצג עמודה בשורה (line) במערך
- x הינו הערך של האיבר במקום k שבשורה: $x = \text{matrix}[\text{line}][k]$

	0	1	2	3	4
0	7	2	3	5	6
1	4	5	1	5	5
2	4	5	6	5	5
3	4	5	1	9	5
4	1	2	3	5	5

"שורת k" הינה האיבר במקום k ו-x הערכים שאחריו.

לדוגמה, במטריצה משמאל:

- בשורה 0, עבור $k = 1$:
○ "שורת k" הינה הערכים 2, 3, 5
- בשורה 3, עבור $k = 0$:
○ "שורת k" הינה הערכים 4, 5, 1, 9, 5
- בשורה 2, עבור $k = 2$:
○ אין "שורת k" מאחר והערך במקום [2][2] הוא 6, ואין 6 ערכים אחרי מקום זה (חריגה מגבול השורה במערך)

נגדיר את המונח: **k-Same**: שתי שורות במטריצה נחשבות **k-Same** אם בשתי שורות אלה, עבור k זהה:

- יש "שורת k" ב-2 שורות אלה
 - כל ערכי "שורת k" ב-2 שורות אלה זהים
- לדוגמה, במטריצה המתוארת בשאלה:
- עבור שורות 0 ו-4, ועבור $k = 1$: מתקיים **k-Same** מאחר ובשתי שורות אלה "שורת k" מכילה בדיוק אותם ערכים: 2, 3, 5
 - עבור שורות 1 ו-3, ועבור $k = 2$: לא מתקיים **k-Same**: בשורה 1 "שורת k" מכילה את הערכים 5, 1. בשורה 3 "שורת k" מכילה את הערכים 9, 1

חשוב להגדיש: כללים אלה נכונים לכל מטריצה מלבנית/ריבועית

נתונה חתימת הפונקציה:

```
public static boolean sameKLine(int[][] matrix,
                                int line1, int line2, int k)
```

כתבו קוד הבודק האם שתי שורות נתונות, ו-k נתון – הינן **k-Same**

הערה חשובה: לא ניתן להניח שנתוני השורות, k, וכן k הערכים ב "שורת k" נמצאים בגבולות המערך

שאלה 6 (15 נקודות):

במחלקה נפרדת (class נפרד) כתבו קוד הבודק את הפונקציה בשאלה 5 (sameKLine). תעדו כל בדיקה, עם הערה – מה נבדק בסעיף זה. בססו את הבדיקות על המטריצה המוגדרת בקובץ הקלט mat1.txt (קובץ המצורף לכם).

לתשומת ליבכם: חישבו טוב על כל המקרים להם אתם צריכים לתכנן בדיקה – בהתאם לפתרון שכתבתם בשאלה 5. חישבו על המקרים איתם מתמודדת הפונקציה בשאלה זו, ובהתאם תכננו בדיקות להוכחת תקינות הפונקציה על כל האפשרויות הקיימות. חישבו טוב על מקרי קצה, מצבי ביניים, מצבים תקינים ומצבים שגויים. יש לרשום לפחות 3 מקרים שונים.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.Arrays;

public class exam_moed_X {

    public static void main(String[] args) throws FileNotFoundException {
        // question 1 test
        int[][] matrix = readFromFile("mat1.txt");
        System.out.println("Question no' 1 - The input matrix is:");
        printMatrix(matrix);

        // question 2 test
        int[][] matrixS = readFromFile("mat1.txt");
        System.out.println("\nQuestion no' 2 - The transformed matrix is:");
        transformMatrix(matrixS);
        printMatrix(matrixS);

        // question 3 test
        System.out.println("\nQuestion no' 3 - is original matrix is double side? "
            + doubleSide(matrix));
        System.out.println("Question no' 3 - is the transformed matrix is double side? "
            + doubleSide(matrixS));

        // question 4 test
        System.out.println("\nQuestion no' 4 - is original matrix is double side? "
            + recDoubleSide(matrix));
        System.out.println("Question no' 4 - is the transformed matrix is double side? "
            + recDoubleSide(matrixS));

        // question 5 test
        int[][] mat = { {7,2,3,5,6}
            ,{4,5,1,5,5}
            ,{4,5,6,5,5}
            ,{4,5,1,9,5}
            ,{1,2,3,5,5} };
        System.out.println("\nQuestion no' 5 - is line0/line4, k=2 - k-Same? "
            + sameKLine(mat, 0,4,1));
        System.out.println("Question no' 5 - is line1/line3, k=3 - k-Same? "
            + sameKLine(mat, 1,3,2));
    }
}
```

// Question 1

```
public static int[][] readFromFile(String fileName) throws FileNotFoundException {
    File f = new File(fileName);
    int[][] matrix;
    int size;

    Scanner s = new Scanner(f);
    size = s.nextInt();

    matrix = new int[size][size];
    for (int row = 0; row < matrix.length; row++)
        for (int col = 0; col < matrix[row].length; col++)
            matrix[row][col] = s.nextInt();

    s.close();
    return matrix;
}
```

// Question 2:

```
public static void transformMatrix(int[][] matrix) {
    for (int row = 0; row < matrix.length-1; row++)
        for (int col = row+1; col < matrix[row].length; col++)
            swapRowCol(matrix, row, col);
}

public static void swapRowCol(int[][] matrix, int row, int col) {
    int temp = matrix[row][col];
    matrix[row][col] = matrix[col][row];
    matrix[col][row] = temp;
}
```

// Question 3:

```
public static boolean doubleSide(int[][] matrix) {
    boolean isDoubleSide = true;
    for (int row = 0; isDoubleSide && row < matrix.length/2; row++)
        isDoubleSide = areIdenticalRows(matrix, row, matrix.length-1-row);
    return isDoubleSide;
}

public static boolean areIdenticalRows(int[][] matrix, int row1, int row2) {
    boolean identicalRows = true;
    for (int col = 0; identicalRows && col < matrix[row1].length; col++)
        identicalRows = (matrix[row1][col] == matrix[row2][col]);
    return identicalRows;
}
```

// Question 4:

```
public static boolean recDoubleSide(int[][] matrix) {
    return recDoubleSideHelper(matrix, 0);
}
```

```
public static boolean recDoubleSideHelper(int[][] matrix, int row) {
    if (matrix.length/2 >= row)
        return true;
    if (!recIdenticalRows(matrix, row, matrix.length-1-row))
        return false;
    return recDoubleSideHelper(matrix, row+1);
}
```

```
public static boolean recIdenticalRows(int[][] matrix, int row1, int row2) {
    return recIdenticalRowsHelper(matrix, row1, row2, matrix.length-1);
}
public static boolean recIdenticalRowsHelper(int[][] matrix, int row1, int row2
                                             , int col) {
    if (col == 0)
        return matrix[row1][col] == matrix[row2][col];
    if (matrix[row1][col] != matrix[row2][col])
        return false;
    return recIdenticalRowsHelper(matrix, row1, row2, col-1);
}
```

// Question 5:

```
public static boolean sameKLine(int[][] matrix, int line1, int line2, int k) {
    if (line1 < 0 || line2 < 0 || line1 >= matrix.length || line2 >= matrix.length)
        return false;
    if (k < 0 || k >= matrix.length)
        return false;
    if (matrix[line1][k] != matrix[line2][k])
        return false;
    if (matrix[line1][k]+k >= matrix.length || matrix[line2][k]+k >= matrix.length)
        return false;
    for (int col = k+1; col <= matrix[line1][k]+k; col++)
        if (matrix[line1][col] != matrix[line2][col])
            return false;
    return true;
}
}
```



```
// Question 5
import static org.junit.jupiter.api.Assertions.*;
import java.io.FileNotFoundException;
import org.junit.jupiter.api.Test;

class exam_moed_X_quest5Test {

    @Test
    void test() throws FileNotFoundException {
        int[][] mat = exam_moed_X.readFile("mat1.txt");
        int[][] mat1 = new int[0][0];
        int[][] mat2 = new int[1][1];

        assertTrue(exam_moed_X.sameKLine(mat, 1, 2, 1));
        assertTrue(exam_moed_X.sameKLine(mat, 0, 3, 1));
        assertTrue(exam_moed_X.sameKLine(mat, 0, 3, 0));
        assertTrue(exam_moed_X.sameKLine(mat, 2, 1, 0));
        assertFalse(exam_moed_X.sameKLine(mat, 0, 3, 2));
        assertFalse(exam_moed_X.sameKLine(mat, 0, 1, 1));
        assertFalse(exam_moed_X.sameKLine(mat, 0, 1, 7));
        assertFalse(exam_moed_X.sameKLine(mat, 10, 1, 7));
        assertFalse(exam_moed_X.sameKLine(mat, 0, 11, 7));

        assertFalse(exam_moed_X.sameKLine(mat1, 0, 0, 0));

        assertTrue(exam_moed_X.sameKLine(mat2, 0, 0, 0));
    }
}
```