פתרון - מועד Y

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

במבחן נעסוק **במערך של מטריצות** ריבועיות (מערך תלת ממדי) של מספרים טבעיים (שלמים חיובים).

1. לאחר ההזדהות במחשב – תמצאו את התיקייה 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)
   - קובץ טקסטואלי בשם: arrayOfMatrices.txt
     - קובץ זה הוא קובץ דוגמה לשימוש בשאלה 1 (וכן שימושי לשימוש בבדיקות של שאלות 2-4)
   - קובץ טקסטואלי בשם: stringsList.txt
     - קובץ זה הינו קובץ עזר לשימושכם, ככלי עזר לבדיקות שאלה 5 ו- 6
2. בהתאם להנחיות הטכנאי המלווה את הכיתה, יש להגדיר תיקייה של workspace למבחן. שם התיקיה הוא: ID_XXXXXXXXX (כאשר XXXXXXXXX זו תעודת הזהות שלך) בדיסק המיועד לכך. שימו לב: במחשב יש 2 גרסאות של Eclipse, השתמשו בגרסה הרשומה לכם כאן משמאל.

3. יש להקים פרויקט בשם ID_XXXXXXXXX (עם תעודת הזהות שלך)
   - לנוחיות עבודה שלך, מומלץ להעתיק את קבצי הטקסט לתיקיית הפרויקט לעבודה נוחה עם קבצים
4. פתרון המבחן יכיל 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[][][] getArrayOfMatrices(String fileName)
                    throws FileNotFoundException
```

```
3
4

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

 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15 16

14 15 16 13
10 11 12  9
16 15 14 13
 2  3  4  1
```

הפונקציה מקבלת מחרוזת המכילה **שם קובץ** וקוראת מהקובץ נתוני מערך של מטריצות.

השורה הראשונה בקובץ מכילה מספר יחיד X המייצג את מספר המטריצות.

השורה השנייה בקובץ מכילה מספר יחיד Y המייצג את ממדי מטריצות ריבועיות (YxY).

לאחר מכן תהיינה X קבוצות של YxY ערכים שלמים.

כתבו את גוף הפונקציה כך שתקרא את נתוני הקובץ ותחזיר מערך של מטריצות המלאות עם הערכים בהתאמה לנתוני הקובץ.

עבור קובץ הקלט בדוגמה הנתונה תתקבלנה 3 מטריצות, שכשכל אחת מהן הינה מטריצה של 4x4.

אין צורך לבדוק את נכונות/תקינות הקלט!

שאלה 2 (20 נקודות):

נתונה חתימת הפונקציה הבאה:

```
public static boolean areThere2IdenticalMatrices(int[][][] aom)
```

```
4
4

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

 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15 16

14 15 16 13
10 11 12  9
16 15 14 13
 2  3  4  1

 1 2 3 4
 2 3 4 1
 3 4 1 2
 4 1 2 3
```

הפונקציה מקבלת מערך של מטריצות (כל המטריצות בעלות אותם ממדים). הפונקציה בודקת האם יש 2 מטריצות, בתוך מערך המטריצות, שיש זהות מלאה בערכים שלהן. במידה וכן, מוחזר true אחרת מוחזר false.

עבור דוגמת הקלט בשאלה 1 – הפונקציה תחזיר false.

עבור דוגמת הקלט משמאל – הפונקציה תחזיר true (מטריצה ראשונה ואחרונה זהות)

שאלה 3 (10 נקודות):

נתונה חתימת הפונקציה הבאה:

```
public static void sortByDiagonalSum(int[][][] aom)
```

הפונקציה מקבלת מערך של מטריצות (כל המטריצות ריבועיות בעלות אותם ממדים). הפונקציה ממיינת את המטריצות לפי סכום אלכסון ראשי (האלכסון ממקום [0][0] למטה/ימינה).

עבור הדוגמה המופיעה משמאל (נתוני 4 מטריצות לפני מיון):

```
14 15 16 13
10 11 12  9
16 15 14 13
 2  3  4  1

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

 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15 16

 1 2 3 4
 2 3 4 1
 3 4 1 2
 4 1 2 3
```

- סכום אלכסון מטריצה ראשונה – 40
- סכום אלכסון מטריצה שנייה – 8
- סכום אלכסון מטריצה שלישית – 34
- סכום אלכסון מטריצה רביעית – 8

לאחר המיון, יהיה הסדר שלהן, במערך המטריצות, בהתאמה:

```
Matrix at index 0 - the diagolal sum is: 8
   1       2       3       4
   2       3       4       1
   3       4       1       2
   4       1       2       3

Matrix at index 1 - the diagolal sum is: 8
   1       2       3       4
   2       3       4       1
   3       4       1       2
   4       1       2       3

Matrix at index 2 - the diagolal sum is: 34
   1       2       3       4
   5       6       7       8
   9      10      11      12
  13      14      15      16

Matrix at index 3 - the diagolal sum is: 40
  14      15      16      13
  10      11      12       9
  16      15      14      13
   2       3       4       1
```

בעת כתיבת הפתרון לשאלה זו חשבו היטב על מבנה מודולרי מתאים, שיעזור בתהליך. בפתרון שאלה זו יש מקום לפונקציית עזר נוספת (לפחות אחת, ואף שתיים).

את המיון עצמו, בסוס על אחד מאלגוריתמי המיון שלמדנו (ראו בדפי העזר של המבחן) – ועשו לו התאמה למיון שיסדר את המטריצות במערך המטריצות, על בסיס ערך סכום האלכסונים.

שאלה 4 (20 נקודות):

בשאלה זו שני חלקים. את שניהם יש לממש באופן רקורסיבי.

סעיף א':

ממשו את הפונקציה **הרקורסיבית**:

```
public static int recDiagonalSum(int[][] matrix)
```

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

הפונקציה מקבלת מטריצה ריבועית, ומחשבת בעזרת helper **רקורסיבי**, את סכום האלכסון הראשי של המטריצה.

לדוגמה: עבור המטריצה הריבועית משמאל, יוחזר הערך 34 (1+6+11+16)

סעיף ב':

ממשו את הפונקציה **הרקורסיבית**:

```
public static int recSearchForDiagonal(int[][][] aom, int serachValue)
```

הפונקציה מקבלת מערך של מטריצות, שנתון שהן ממוינות במערך המטריצות לפי סכום האלכסון הראשי (מהקטן לגדול) – כלומר, מערך המטריצות הינו לאחר המיון שבוצע בשאלה 3.

בנוסף, הפונקציה מקבלת ערך נוסף – "**ערך החיפוש**".

הפונקציה בודקת האם קיימת מטריצה, במערך המטריצות, אשר סכום האלכסון הראשי שלה שווה ל-"ערך החיפוש" שהתקבל. במידה וכן, יוחזר האינדקס של המטריצה במערך המטריצות (אם יש יותר ממטריצה אחת עם סכום אלכסון השווה לערך – יש להחזיר אינדקס של אחת מהמטריצות). במידה ואין מטריצה אשר סכום האלכסון שלה שווה ל- "ערך החיפוש" יוחזר הערך (1-) כסימן לכך שלא קיימת מטריצה עם סכום זה.

לדוגמה, על בסיס דוגמת המטריצות לאחר מיון בשאלה 3:

- עבור הערך 34 – יוחזר הערך 2
- עבור הערך 7 – יוחזר הערך 1-

הנחיות חשובות לפתרון סעיף זה:

1. הפתרון **חייב** להיות פתרון רקורסיבי. במידת הצורך ניתן להשתמש ב helper רקורסיבי
2. יש לשלב בפתרון סעיף זה שימוש בפונקציה שכתבתם בסעיף א'

שאלה 5 (20 נקודות):

ממשו את הפונקציה הבאה:

```
public static void lexicalSorting(String[] stringsForSort)
```

הפונקציה מקבלת מערך חד ממדי של מחרוזות.

הפונקציה תמיין את המחרוזות במערך לפי סדר מילוני (לקסיקוגרפי), מהקטן לגדול, ללא חשיבות ל-Upper/Lower Case.

יש לממש את פתרון המיון בעזרת מיון בועות.

דוגמה:

- בהינתן מערך המחרוזות הבא (משמאל לימין):

```
[AbCdEfG!@#, xYz123&*, PqR$%^, mNoPqRsT, lKjIhGf, ZyxWvU$#,
                    9876!@#, sTuVwXyZ, dCbA123&*, !@#lmnOP]
```

- לאחר המיון, הסדר במערך יהיה כדלהלן (משמאל לימין):

```
[!@#lmnOP, 9876!@#, AbCdEfG!@#, dCbA123&*, lKjIhGf,
          mNoPqRsT, PqR$%^, sTuVwXyZ, xYz123&*, ZyxWvU$#]
```

שאלה 6 (15 נקודות):

במחלקה נפרדת (class נפרד) כתבו קוד הבודק את הפונקציה בשאלה 5 (`lexicalSorting`). תעדו כל בדיקה, עם הערה – מה נבדק בסעיף זה.

לנוחיותכם, ניתן לבסס את הבדיקות על המחרוזות המופיעות בקובץ המוגדרת בקובץ stringsList (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 האפשרות הקיימות. חישבו טוב על מקרי קצה, מצבי ביניים, מצבים תקינים ומצבים שגויים. יש לרשום לפחות 3 מקרים שונים.

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

public class exam_moed_Y {

   public static void main(String[] args) throws FileNotFoundException {
      // Question no' 1
      int[][][] aom = getArrayOfMatrices("arrayOfMatrices.txt");
      System.out.printf("Question no' 1 - There are %d matrices\n", aom.length);
      for (int matNum = 0; matNum < aom.length; matNum++) {
        System.out.printf("\nMatrix no' %d:\n", matNum+1);
        printMatrix(aom[matNum]);
      }

      // Question no' 2
      System.out.println("\nQuestion no' 2 - are there 2 identical matrices? "
                                        + areThere2IdenticalMatrices(aom));

      // Question no' 3
      sortByDiagonalSum(aom);
      System.out.printf("\nQuestion no' 3 - the matrices after sort by diagonal sum\n");
      for (int matNum = 0; matNum < aom.length; matNum++) {
        System.out.printf("\nMatrix at index %d - the diagolal sum is: %d\n",
                                    matNum, matrixDiagonalSum(aom[matNum]));
        printMatrix(aom[matNum]);
      }

      // Question no' 4
      System.out.printf("\nQuestion no' 4: matrix number [%d] has diagonal some of
34 (-1 means not found).\n"
                                        ,recSearchForDiagonal(aom, 34));
      System.out.printf("Question no' 4: matrix number [%d] has diagonal some of 7
(-1 means not found).\n"
                                        ,recSearchForDiagonal(aom, 7));

      // Question no' 5
      File inputStrings = new File("stringsList.txt");
      Scanner inStr = new Scanner(inputStrings);
      String[] stringsForSort = new String[inStr.nextInt()];
      for (int i = 0; i < stringsForSort.length; i++)
        stringsForSort[i] = inStr.next();
      inStr.close();
      System.out.println("\nQuestion no'5: the input stirngs: " +
                                        Arrays.toString(stringsForSort));
      lexicalSorting(stringsForSort);
      System.out.println("Question no'5: the srings after sort: " +
                                        Arrays.toString(stringsForSort));
   }
```

```java
   // Question no' 1
public static int[][][] getArrayOfMatrices(String fileName) throws FileNotFoundException {
      File f = new File(fileName);
      Scanner s = new Scanner(f);
      int[][][] aom;
      int matrixSize;

      aom = new int[s.nextInt()][][];
      matrixSize = s.nextInt();

      for (int matNum = 0; matNum < aom.length; matNum++) {
        aom[matNum] = new int[matrixSize][matrixSize];
        for (int row = 0; row < matrixSize; row++)
              for (int col = 0; col < matrixSize; col++)
                    aom[matNum][row][col] = s.nextInt();
      }
      s.close();
      return aom;
   }

   // Question no' 2
   public static boolean areThere2IdenticalMatrices(int[][][] aom) {
      boolean have2IdenticalMatrices = false;
      for (int mat1 = 0; !have2IdenticalMatrices && mat1 < aom.length-1; mat1++)
        for (int mat2 = mat1+1; !have2IdenticalMatrices && mat2 < aom.length; mat2++)
              have2IdenticalMatrices = haveSameValues(aom[mat1],aom[mat2]);
      return have2IdenticalMatrices;
   }
   public static boolean haveSameValues(int[][] mat1, int[][] mat2) {
      boolean sameValues = true;
      for (int row = 0; sameValues  &&  row < mat1.length; row++)
        for (int col = 0; sameValues  &&  col < mat1[row].length; col++)
              sameValues = (mat1[row][col] == mat2[row][col]);
      return sameValues;
   }

   // Question no' 3
   public static void sortByDiagonalSum(int[][][] aom) {
      int tempSum, tempMat[][];
      int[] diagonalSums = getDiagonalSums(aom);
      for (int matNum = 1; matNum < aom.length; matNum++) {
        for (int i = matNum; i > 0 && diagonalSums[i-1] > diagonalSums[i]; i--) {
              tempSum = diagonalSums[i];
              tempMat = aom[i];
              diagonalSums[i] = diagonalSums[i-1];
              aom[i] = aom[i-1];
              diagonalSums[i-1] = tempSum;
              aom[i-1] = tempMat;
        }
      }
   }
   public static int[] getDiagonalSums(int[][][] aom) {
      int[] diagonalSums = new int[aom.length];
      for (int matNum = 0; matNum < aom.length; matNum++)
        diagonalSums[matNum] = matrixDiagonalSum(aom[matNum]);
      return diagonalSums;
   }
```

```
   public static int matrixDiagonalSum(int[][] mat) {
      int sum = 0;
      for (int i = 0; i < mat.length; i++)
        sum+= mat[i][i];
      return sum;
   }

   // Question no' 4
   public static int recDiagonalSum(int[][] matrix) {
      return recDiagonalSumHelper(matrix, matrix.length-1);
   }
   private static int recDiagonalSumHelper(int[][] matrix, int i) {
      if (i == 0)
        return matrix[0][0];
      return recDiagonalSumHelper(matrix,i-1) + matrix[i][i];
   }
   public static int recSearchForDiagonal(int[][][] aom, int serachValue) {
      return recSearchForDiagonalHelper(aom, serachValue, 0, aom.length-1);
   }
   public static int recSearchForDiagonalHelper(int[][][] aom, int value, int low, int high) {
      if (low > high)
        return -1;
      int middle = (low+high)/2;
      int diagonalSum = recDiagonalSum(aom[middle]);
      if (value == diagonalSum)
              return middle;
      if (value < diagonalSum)
        return recSearchForDiagonalHelper(aom, value, low, middle-1);
      else
        return recSearchForDiagonalHelper(aom, value, middle+1, high);
   }

   // Question no' 5
   public static void lexicalSorting(String[] stringsForSort) {
      boolean hasChanged = true;
      String tempStr;

      for (int i = stringsForSort.length-1; hasChanged  &&  i > 0; i--) {
        hasChanged = false;
        for (int j = 0; j < i; j++)
              if (stringsForSort[j].compareToIgnoreCase(stringsForSort[j+1]) > 0) {
                     tempStr = stringsForSort[j];
                     stringsForSort[j] = stringsForSort[j+1];
                     stringsForSort[j+1] = tempStr;
                     hasChanged = true;
              }
      }
   }
   // help utilities
   public static void printMatrix(int[][] matrix) {
      for (int row = 0; row < matrix.length; row++) {
        for (int col = 0; col < matrix[row].length; col++)
              System.out.printf("%4d\t",matrix[row][col]);
        System.out.println();
      }
   }
}
```

```java
  // Question no' 6
import static org.junit.jupiter.api.Assertions.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import org.junit.jupiter.api.Test;

class exam_moded_Y_quest6Test {

	@Test
	void test() throws FileNotFoundException {
		File f = new File("stringsList.txt");
		Scanner s = new Scanner(f);
		String[] strList = new String[s.nextInt()];
		for (int i = 0; i < strList.length; i++)
			strList[i] = s.next();
		s.close();
		exam_moed_Y.lexicalSorting(strList);

		// check the excepted value of the first string
		assertTrue(strList[0].equals("!@#lmnOP"));

		// check the excepted value of the last string
		assertTrue(strList[strList.length-1].equals("ZyxWvU$#"));

		// check an example value of a string at the middle
		assertTrue(strList[4].equals("lKjIhGf"));
	}
}
```