פתרון מועד X

שאלה 1 (40 נקודות):

לוח פתקים (NotesBoard) מכיל פתקים שונים, כך שכל פתק בצבע מסוים וכתובה עליו תזכורת שנכתבה בתאריך מסוים. בלוח יש שורות ועמודות עליהן ניתן להדביק את הפתקים. ניתן להדביק פתק רק בתא שפנוי בלוח.

להלן תיאור של מחלקות הקיימות במערכת ומחלקות **שעליכם להוסיף או לעדכן**, בהמשך יוצג main לדוגמה ופלט תוצאת הרצתו.

1. נתונה בהמשך באופן **חלקי** המחלקה עבור תאריך (Date):
   1.1. תכונות המחלקה: יום, חודש, שנה.
   1.2. נתון קונסטרקטור המקבל את כל התכונות ומאתחל את האובייקט.
   1.3. נתונה הפעולה toString המחזירה מחרוזת עם נתוני האובייקט כפי שמופיע בפלט בהמשך.
   1.4. **(6 נק') כתבו** את הפעולה isBefore המקבלת כפרמטר תאריך אחר ומחזירה true אם התאריך באובייקט ישן יותר (כלומר, לפני) מהתאריך האחר שהתקבל, אחרת יוחזר false.

2. נתונה באופן **חלקי** מחלקה עבור פתק בלוח (StickyNote):
   2.1. תכונות המחלקה הן צבע הפתק, התאריך בו נכתב והטקסט שרשום עליו.
   2.2. נתון קונסטרקטור המחלקה המקבל את כל הנתונים ומאתחל את נתוני האובייקט.
   2.3. נתונות פעולות get לכל התכונות.
   2.4. נתונה הפעולה toString המחזירה מחרוזת עם נתוני האובייקט כפי שמופיע בפלט בהמשך.
   2.5. **(6 נק') כתבו** את הפעולה isOlder המקבלת כפרמטר פתק אחר ומחזירה true אם הפתק נכתב בתאריך ישן מאשר התאריך בו נכתב הפתק שהתקבל כפרמטר, אחרת יוחזר false.

3. נתונה באופן **חלקי** מחלקה עבור לוח הפתקים (NotesBord):
   3.1. תכונות המחלקה: מערך דו-מימדי ריבועי של פתקים.
   3.2. **(4 נק') כתבו** את קונסטרקטור המחלקה המקבל את מספר השורות והעמודות עבור פתקים בלוח ומקצה את מערך ההודעות, שיהיה כרגע ריק מהודעות.
   3.3. נתונה הפעולה toString המחזירה מחרוזת עם נתוני האובייקט כפי שמופיע בפלט בהמשך.
   3.4. נתונה הפעולה addStickyNote המקבלת פתק ומיקום בלוח, ובמידה והמקום פנוי מוסיפה את הפתק ללוח ומחזירה true, אחרת מחזירה false.

3.5. **(6 נק') כתבו** את הפעולה numOfStickyNotesFromColor המקבלת כפרמטר משתנה מטיפוס צבע (המוגדר כ- enum במחלקה StickyNote) ומחזירה את מספר הפתקים בלוח בעלי צבע זה.

3.6. **(10 נק') כתבו** את הפעולה getMostUsedColor המחזירה את הצבע שיש ממנו הכי הרבה פתקים בלוח. במידה ויש שיוויון בין מספר צבעים, לא משנה מי יוחזר.

3.7. **(8 נק') כתבו** את הפעולה findOldestStickNote המחזירה הפניה לפתק הכי ישן בלוח. במידה ואין פתקים בלוח יש להחזיר null.

להלן main לדוגמה:

```
public class Program {

	public static void main(String[] args) {
		NotesBoard myNotes = new NotesBoard(4, 5);

		myNotes.addStickyNote(new StickyNote(StickyNote.eColor.BlueSky,
				new Date(1, 5, 2022), "Don't forget call mom"), 2, 3);
		myNotes.addStickyNote(new StickyNote(StickyNote.eColor.LemonGreen,
				new Date(2, 5, 2022), "Buy rice"), 3, 0);
		myNotes.addStickyNote(new StickyNote(StickyNote.eColor.LemonGreen,
				new Date(15, 3, 2022), "Buy Milky"), 3, 3);
		myNotes.addStickyNote(new StickyNote(StickyNote.eColor.BlueSky,
				new Date(19, 1, 2022), "Cook Friday's big meal"), 1, 0);
		myNotes.addStickyNote(new StickyNote(StickyNote.eColor.LightYellow,
				new Date(19, 2, 2022), "Do homework in math"), 2, 0);
		myNotes.addStickyNote(new StickyNote(StickyNote.eColor.Pink,
				new Date(3, 3, 2022), "Call Gogo"), 1, 1);
		myNotes.addStickyNote(new StickyNote(StickyNote.eColor.BlueSky,
			new Date(31, 5, 2022), "Buy grandmother birhday present"), 1, 3);

		System.out.println(myNotes.toString());

		StickyNote.eColor mostUsedColor = myNotes.getMostUsedColor();
		int numOfAppearancesForMostUsedColor =
				myNotes.numOfStickyNotesFromColor(mostUsedColor);
		System.out.println("The most used color in the board is "
				+ mostUsedColor
				+ " (" + numOfAppearancesForMostUsedColor +" times)");

		StickyNote oldest = myNotes.findOldestStickNote();
		System.out.println("The oldest note is " + oldest.toString());

	}

}
```

להלן פלט עבור main זה:

```
The notes in my board:
       1)  [BlueSky written on the 19/1/2022] Cook Friday's big meal
       2)  [Pink written on the 3/3/2022] Call Gogo
       3)  [BlueSky written on the 31/5/2022] Buy grandmother birhday present
       4)  [LightYellow written on the 19/2/2022] Do homework in math
       5)  [BlueSky written on the 1/5/2022] Don't forget call mom
       6)  [LemonGreen written on the 2/5/2022] Buy rice
       7)  [LemonGreen written on the 15/3/2022] Buy Milky

The most used color in the board is BlueSky (3 times)
The oldest note is  [BlueSky written on the 19/1/2022] Cook Friday's big meal
```

השלימו את המחלקה Date עפ"י הנדרש בהגדרות למעלה:

```java
public class Date {
	private int day, month, year;

	public Date(int day, int month, int year) {
		this.day = day;
		this.month = month;
		this.year = year;
	}

	public String toString() {
		return day + "/" + month + "/" + year;
	}

	public boolean isBefore(Date other) {
		if (year < other.year)
			return true;
		else if (year > other.year)
			return false;
		else { // same year..
			if (month < other.month)
				return true;
			else if (month > other.month)
				return false;
			else { // same month..
				return day < other.day;
			}
		}
	}

}
```

השלימו את המחלקה StickyNote עפ"י הנדרש בהגדרות למעלה:

```
public class StickyNote {
	public enum eColor {Pink, BlueSky, LemonGreen, LightYellow};

	private eColor theColor;
	private Date whenWritten;
	private String theText;

	public StickyNote(eColor theColor, Date whenWritten, String theText) {
		this.theColor = theColor;
		this.whenWritten = whenWritten;
		this.theText = theText;
	}

	public eColor getTheColor() {
		return theColor;
	}

	public Date getWhenWritten() {
		return whenWritten;
	}

	public String getTheText() {
		return theText;
	}

	public String toString() {
		return " [" + theColor +  " written on the " + whenWritten.toString() +
"] " + theText;
	}

	public boolean isOlder(StickyNote other) {
		return getWhenWritten().isBefore(other.getWhenWritten());
	}

}
```

השלימו את המחלקה NotesBoard עפ"י הנדרש בהגדרות למעלה:

```
public class NotesBoard {
	private StickyNote allNotes[][];

	public NotesBoard(int lines, int notesInLine) {
		allNotes = new StickyNote[lines][notesInLine];
	}

	// add StickyNote
	public boolean addStickyNote(StickyNote theNote, int line, int indexInLine) {
		if (allNotes[line][indexInLine] == null) {
			allNotes[line][indexInLine] = theNote;
			return true;
		}
		else
			return false;
	}

	public int numOfStickyNotesFromColor(StickyNote.eColor theColor) {
		int count = 0;

		for (int i=0 ; i < allNotes.length ; i++) {
			for (int j=0 ; j < allNotes[i].length ; j++) {
				if (allNotes[i][j] != null && allNotes[i][j].getTheColor()
== theColor)
					count++;
			}
		}
		return count;
	}

	public StickyNote.eColor getMostUsedColor() {
		// count how many times each color appears
		StickyNote.eColor allColors[] = StickyNote.eColor.values();
		int[] counters = new int[allColors.length];
		for (int i=0 ; i < allNotes.length ; i++) {
			for (int j=0 ; j < allNotes[i].length ; j++) {
				if (allNotes[i][j] != null)
					counters[allNotes[i][j].getTheColor().ordinal()]++;
			}
		}

		// find the biggest value
		int maxIndex = 0;
		for (int i=1 ; i < counters.length ; i++) {
			if (counters[maxIndex] < counters[i])
				maxIndex = i;
		}

		return allColors[maxIndex];
	}

	public String toString() {
		StringBuffer sb = new StringBuffer("The notes in my board:\n");
		int noteNumber = 1;
		for (int i=0 ; i < allNotes.length ; i++) {
			for (int j=0 ; j < allNotes[i].length ; j++) {
				if (allNotes[i][j] != null) {
```

```
                                sb.append("\t" + (noteNumber++) + ") " +
allNotes[i][j] + "\n");
                        }
                    }
            }

            return sb.toString();
        }

        public StickyNote findOldestStickNote() {
            StickyNote oldestStickyNote = null;

            for (int i=0 ; i < allNotes.length ; i++) {
                for (int j=0 ; j < allNotes[i].length ; j++) {
                    if (allNotes[i][j] != null){
                        if (oldestStickyNote == null)
                            oldestStickyNote = allNotes[i][j];
                        else {
                            if (allNotes[i][j].isOlder(oldestStickyNote))
                                oldestStickyNote = allNotes[i][j];
                        }
                    }
                }
            }

            return oldestStickyNote;
        }

}
```

שאלה 2 (30 נקודות):

סעיף א' (20 נק'):

נתונה מתודת *main* הבאה:

```
public static void main(String[] args) {
    int[] arr1 = { 1,3,5,7,12,14,15 };
    int[] arr2 = { 2,4,6,8,9,10,11,12,16 };
    int[] mergedArr = reverseMerge2SortedArr (arr1,arr2);
    System.out.println( Arrays.toString(mergedArr) );
}
```

בנוסף נתונה המתודה *reverseMerge2SortedArr* הבאה:

```
public static int[] reverseMerge2SortedArr(int[] arr1, int[] arr2) {
    int[] res = new int[arr1.length + arr2.length];
    reverseMerge2SortedArr(arr1, arr2, 0, 0, res);
    return res;
}
```

עליכם לממש את המתודה **הרקורסיבית** *reverseMerge2SortedArr* המקבלת 5 פרמטרים:

- *arr1* – מערך חד ממדי של שלמים, ממוין בסדר עולה
- *arr2* – מערך חד ממדי של שלמים, ממוין בסדר עולה
- *start1* – מספר שלם המייצג אינדקס (מיקום) במערך arr1 (החל ממנו נשתמש במערך לצורך המיון)
- *start2* – מספר שלם המייצג אינדקס (מיקום) במערך arr2 (החל ממנו נשתמש במערך לצורך המיון)
- *merged* – מערך חד ממדי, מערך התוצאה, שיכיל את שני ערכי המערכים ממוינים בסדר מהגדול לקטן

המתודה **מכניסה ערכים** למערך merged מתוך הערכים שבשני המערכים, כאשר הם ממוינים בסדר **יורד** (כלומר, מהגדול לקטן).

במימוש המתודה:

- אין להשתמש בלולאה (מכל סוג שהוא) !!!
- אין להוסיף פרמטרים למתודה
- אין להוסיף מתודת עזר נוספת (כלומר, אין להוסיף מתודת helper)
- גודל מערכי הקלט – לא בהכרח זהה (כמובן, שייתכן מצב ששניהם באותו גודל). הפתרון צריך להיות מתאים לכל גודל מערך.
- **הפתרון חייב להיות במתודה יחידה ורקורסיבית!**

דוגמת פלט להרצת מתודת ה main הנתונה בתחילת השאלה:

```
[16, 15, 14, 12, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

פתרון:

```
private static void reverseMerge2SortedArr(int[] arr1, int[] arr2
                                , int start1, int start2, int[] merged) {
 // none of them have values
 if ( start1 == arr1.length  &&  start2 == arr2.length )
       return;
 // only arr2 has more values
 if ( start1 == arr1.length ) {
       merged[merged.length-1-start1-start2] = arr2[start2];
       reverseMerge2SortedArr(arr1, arr2, start1, start2+1, merged);
       return;
 }
 // only arr1 has more values
 if ( start2 == arr2.length ) {
       merged[merged.length-1-start1-start2] = arr1[start1];
       reverseMerge2SortedArr(arr1, arr2, start1+1, start2, merged);
       return;
 }
 // both of them have values - check what is smaller
 if ( arr1[start1] < arr2[start2] ) {
       // arr1 value is smaller then arr2 value
       merged[merged.length-1-start1-start2] = arr1[start1];
       reverseMerge2SortedArr(arr1, arr2, start1+1, start2, merged);
       return;
 }
 else if ( arr1[start1] > arr2[start2] ) {
       // arr2 value is smaller then arr1 value
       merged[merged.length-1-start1-start2] = arr2[start2];
       reverseMerge2SortedArr(arr1, arr2, start1, start2+1, merged);
       return;
 }
 else {      // both values are equals
       merged[merged.length-1-start1-start2] = arr1[start1];
       merged[merged.length-1-start1-start2-1] = arr2[start2];
       reverseMerge2SortedArr(arr1, arr2, start1+1, start2+1, merged);
       return;
 }
}
```

<u>סעיף ב (10 נק'):</u>

נתון הקוד הבא:

```
public static boolean what(int n1, int n2, int n3) {
      if (n1 < 10) {
            if (n3 == 1)
                  return n1 == n2;
            else if (n3 == 0) {
                  return n1 != n2;
            }
            else // count > 1
                  return false;
```

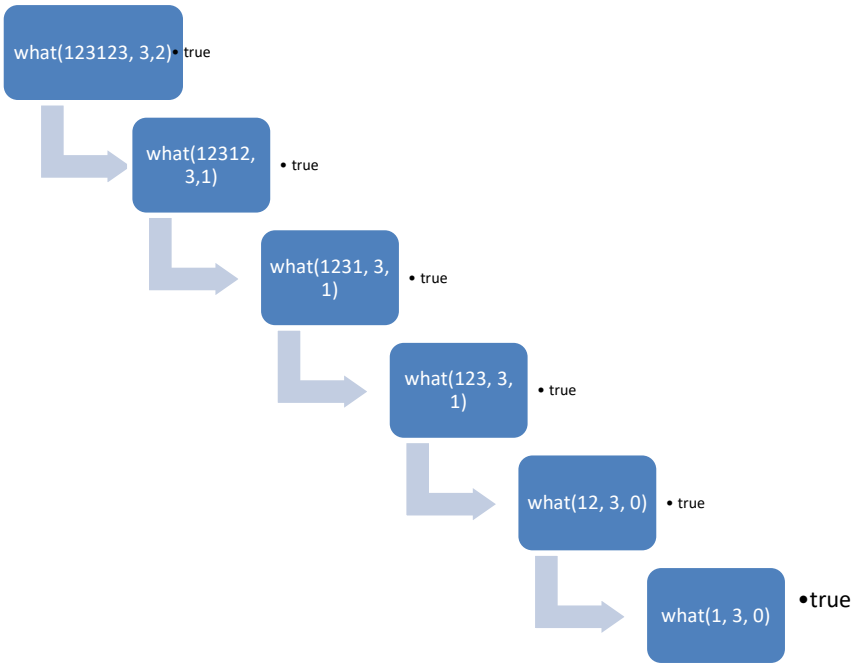
```
        }

        if (n1%10 == n2) {
                return what(n1/10, n2, n3-1);
        }
        else {
                return what(n1/10, n2, n3);
        }
}
```

i. ציירו עץ מעקב עבור הפונקציה what עובר הקריאה הבאה: ***what*(123123, 3, 2)**. כתבו מה הפונקציה תחזיר עבור זימון הפונקציה עם ערכים אלה.

ii. כתבו במשפט אחד מה הפונקציה *what* עושה (שימו לב, אין להסביר מהן הפקודות המבוצעות או לתת דוגמת הרצה – כלומר אין להסביר את "איך", אלא להתמקד ב- "מה").

פתרון:



הפונקציה מקבלת מספר ספרה ומספר נוסף. הפונקציה מחזירה true אם הספרה המתקבלת מופיעה במספר בדיוק כמספר הפעמים של המספר הנוסף, אחרת יוחזר false.

שאלה 3 (30 נקודות):

מטריצת "מסגרות" הינה מטריצה אשר:

- בכל "מסגרת" – כל הערכים במסגרת שווים זה לזה
- הערכים ב- "מסגרות" השונות – לא בהכרח שווים זה לזה

להלן 4 דוגמאות של "מטריצת מסגרות" בגדלים שונים:

דוגמה 1

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

דוגמה 2

	0	1	2	3
0	9	9	9	9
1	9	4	4	9
2	9	4	4	9
3	9	4	4	9
4	9	4	4	9
5	9	9	9	9

דוגמה 3

	0	1	2	3	4	5
0	7	7	7	7	7	7
1	7	4	4	4	4	7
2	7	4	1	1	4	7
3	7	4	1	1	4	7
4	7	4	4	4	4	7
5	7	7	7	7	7	7

דוגמה 4

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

סעיף א' (20 נק'):

כתבו את הפונקציה הבוליאנית הבאה:

```
public static boolean isFrame(int[][] matrix, int frameNum)
```

הפונקציה מקבלת 2 פרמטרים:

- matrix: מערך דו-ממדי של שלמים **(שימו לב שהמטריצה אינה בהכרח ריבועית)**
- frameNum – מספר שלם המייצג "עומק" של המסגרת:
  - 1 – המסגרת החיצונית
  - 2 – המסגרת השנייה (אחד פנימה מהחיצונית)
  - 3 - .... וכך הלאה

הפונקציה בודקת תקינות של מסגרת אחת בלבד. הפונקציה תבדוק שכל ערכי מסגרת מבוקשת (לפי ערכו של frameNumber) שווים לזה לזה. במידה וכל ערכי המסגרת הרלוונטית זהים – יוחזר true, אחרת false.

דוגמה: עבור המערך "דוגמה 1", והערך 2, תיבדק המסגרת השנייה, שכל ערכיה זהים (במקרה זה – הערך של המסגרת הוא 2). עבור המערך "דוגמה 3", והערך 3, תיבדק המסגרת השלישית, שכל ערכיה זהים (במקרה זה – הערך של המסגרת הוא 1)

סעיף ב' (10 נק'):

כתבו את הפונקציה הבוליאנית הבאה:

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

הפונקציה מקבלת פרמטר אחד – מערך דו-ממדי של שלמים.

הפונקציה תבדוק, בעזרת הפונקציה מסעיף א', שהמטריצה הינה "מטריצת מסגרות", כלומר, שכל המסגרות במטריצה, מסגרת אחר מסגרת, הן מסגרות תקינות בהתאם להגדרות בשאלה.

הפונקציה תחזיר True במידה וכל המסגרות תקינות, אחרת יוחזר הערך False.

פתרון:

```
public static boolean isFrame(int[][] matrix, int frameNum)
{
    for (int row = frameNum-1;
                    row <= matrix.length-frameNum; row++) {
        if (matrix[row][frameNum-1] !=
                        matrix[frameNum-1][frameNum-1])
            return false;
        if (matrix[row][matrix[row].length-frameNum]
                        != matrix[frameNum-1][frameNum-1])
            return false;
    }
    for (int col = frameNum-1;
                    col <= matrix[0].length-frameNum; col++){
        if (matrix[frameNum-1][col] !=
                            matrix[frameNum-1][frameNum-1])
            return false;
        if (matrix[matrix.length-frameNum][col] !=
                            matrix[frameNum-1][frameNum-1])
            return false;
    }
    return true;
}

public static boolean isFramedMatrix(int[][] matrix) {
    for (int frameNum = 1; frameNum <=
(Math.min(matrix.length,matrix[0].length)+1)/2; frameNum++)
        if ( !isFrame(matrix,frameNum) )
            return false;
    return true;
}
```