

תוכנה 1

תרגיל מספר 4

הנחיות כלליות:

- קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
- הגשת התרגיל תעשה במערכת ה VirtualTAU בלבד (<http://virtual2002.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip) קובץ ה zip יכול:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
 - ב. קבצי ה java של התוכניות אותם התבקשתם לממש.
 - ג. קובץ טקסט עם העתק של כל קבצי ה java
 - ד. קובץ טקסט בשם answers עם התשובות לשאלות

חשוב: בשאלה 1 ובשאלה 3 יש לממש את כל המתודות הנדרשות במחלקה אחת אשר תקרא Assignment04.
הנכם מתבקשים להקפיד על חתימה של מתודות לפי המפורט בתרגיל, אחרת ייתכן והשאלה לא תיבדק.
יש להריץ ולבדוק את הקוד שכתבתם לפני ההגשה.

חלק א':

מטרת חלק זה לתרגל כתיבת שירותים סטטיים לא טריוויאליים בהינתן החוזה של השירות. בכל סעיף מוגדר שירות למימוש עם שני חוזים אפשריים. עליכם לממש את השירות פעמיים, כך שיתאים לחוזים. ברוב המקרים ניתן לפתור את התרגיל בקלות ע"י הוספת פונקציות עזר (אף שהתרגיל לא דורש זאת במפורש).

הערה: במידת הצורך, עבור כל זוג פונקציות תוכלו להשתמש באחת הפונקציות לצורך מימוש השנייה.

1. בהינתן מערך של מספרים שלמים כקלט, החזירו מערך המכיל את אותם מספרים, אך מסודר כך שלאחר כל מופע של 4 יש מופע של 5. מותר להזיז את כל אברי המערך למיקומים אחרים חוץ מהאברים שערכם 4. להלן מספר דוגמאות:

$\text{fix45}(\{5, 4, 9, 4, 9, 5\}) \rightarrow \{9, 4, 5, 4, 5, 9\}$

$\text{fix45}(\{1, 4, 1, 5\}) \rightarrow \{1, 4, 5, 1\}$

$\text{fix45}(\{1, 4, 1, 5, 5, 4, 1\}) \rightarrow \{1, 4, 5, 1, 1, 4, 5\}$

```
/*
 * @pre occurrences(4,arr) == occurrences(5, arr)
 * @pre arr[arr.length - 1] != 4
 * @pre forall 0 <= i < arr.length-1, arr[i] == 4 $implies arr[i+1]
 != 4
 * @post forall 0 <= i < arr.length-1, exists 0 <= j < arr.length-1
 s.t.$prev(arr[j]) == arr[i]
 * @post $ret != arr
 * @post "values in $ret are a permutation of values in arr"
 * @post forall 0 <= i < $ret.length-1, $ret[i] == 4 $implies
 $ret[i+1] == 5
 * @post forall 0 <= i < arr.length, arr[i] == 4 $implies $ret[i] == 4
 */
public static int[] fix45A (int[] arr) {
```

הערה: הסימון \$prev מסמן את הערך הקודם (לפני הפעלת המתודה) של משתנה כלשהו.

```
/*
 * @post ((occurrences(4,arr) == occurrences(5, arr)) AND
 * (arr[arr.length - 1] != 4) AND
 * (forall 0 <= i < arr.length-1, arr[i] == 4 $implies arr[i+1] !=
 4))
 * $implies
 * ((forall 0 <= i < arr.length-1, exists 0 <= j < arr.length-1 s.t.
 $prev(arr[j]) == arr[i]) AND
 * ($ret != arr) AND
 * ("values in $ret are permutation of values in arr") AND
 * (forall 0 <= i < $ret.length-1, $ret[i]==4 $implies
 $ret[i+1]==5) AND
 * (forall 0 <= i < arr.length, arr[i] == 4 $implies $ret[i] == 4))
 */
public static int[] fix45B (int[] arr) {
```

2. בהינתן מערך של מספרים שלמים (integers), האם ניתן לבחור תת קבוצה מתוך המערך כך שסכום תת הקבוצה שווה למספר מטרה מסוים? ניתן לפתור בעיה זו ע"י רקורסיה. טיפ: במקום להסתכל על כל המערך, נסתכל על המערך החל מאינדקס start ועד לסופו. הקורא לשירות זה יכול לציין שברצונו לפתור עבור כל המערך ע"י מתן ערך 0 ל-start. להלן כמה דוגמאות:

groupSum(0, {2, 4, 8}, 10) → true

groupSum(0, {2, 4, 8}, 14) → true

groupSum(0, {2, 4, 8}, 9) → false

```

/*
 * @pre 0 <= start <= nums.length - 1
 * @pre nums != null
 * @pre  $\forall 0 \leq i \leq \text{nums.length} - 1, \text{nums}[i] \in \mathbb{Z}$ 
 * @post  $(\exists S \subseteq \text{nums}[\text{start}] \dots \text{nums}[\text{nums.length} - 1], \sum_{s \in S} s == \text{target}) \Rightarrow \$ret == true$ 
 * @post  $(\neg \exists S \subseteq \text{nums}[\text{start}] \dots \text{nums}[\text{nums.length} - 1], \sum_{s \in S} s == \text{target}) \Rightarrow \$ret == false$ 
 */
public static boolean groupSumA(int start, double[] nums, int target) {
}

```

```

/*
 * @pre 0 <= start <= nums.length - 1
 * @pre nums != null
 * @post  $(\exists S \subseteq [\text{start} \dots \text{nums.length} - 1], \sum_{s \in S} \text{nums}[s] == \text{target}) \Rightarrow \$ret == true$ 
 * @post  $(\neg \exists S \subseteq [\text{start} \dots \text{nums.length} - 1], \sum_{s \in S} \text{nums}[s] == \text{target}) \Rightarrow \$ret == false$ 
 */
public static boolean groupSumB(int start, int[] nums, int target) {
}

```

3. בהינתן מחרוזת, החזירו true אם מספר המופעים של תת המחרוזת "is" במקום כלשהוא במחרוזת

שווה למספר ההופעות של תת המחרוזת "not" במחרוזת (case sensitive). להלן כמה דוגמאות:

```
equalIsNot("This is not") → false
```

```
equalIsNot("This is notnot") → true
```

```
equalIsNot("noisxxnotyynotxisi") → true
```

```
/*
 * @pre str != null
 * @post $ret == (occurrences("is") == occurrences("not"))
 */
public static boolean equalIsNotA(String str) {

}
```

```
/*
 * @pre (str != null) AND (occurrences("is") == occurrences("not"))
 * @post $ret == (occurrences("is") == occurrences("not"))
 */
public static boolean equalIsNotB(String str) {

}
```

חלק ב':

חלק זה נועד לתרגל כתיבת חוזים עבור שירותים קיימים. בכל סעיף נתונה פונקציה, עליכם כתוב את החוזה (תנאי קדם ואחר) עבור הפונקציה הנתונה. הניחו כי כל הפונקציות אינן בודקות את הקלט.

1. שורש

```
public static double sqrt(double d) {  
    ...  
}
```

2. ערך מוחלט

```
public static double abs(double d) {  
    ...  
}
```

3. עצרת

```
public static int factorial(int n) {  
    ...  
}
```

4. מיון

```
public static void sort(int[] arr) {  
    ...  
}
```

חלק ג':

כתבו את המתודה `isEquation`, המקבלת מערך של מספרים ומחזירה `true` אם ניתן לחלק את המערך לשני תתי מערכים רציפים כך שערכו של ביטוי חשבוני הנוצר מהפעולות האריתמטיות **חיבור**, **חיסור** ו**כפל** בין אברי תת-המערך הראשון, יהיה שווה לערכו של ביטוי אריתמטי דומה הנוצר מאיברי תת-המערך השני.

לא ניתן לשנות את סדר האיברים במערך, אך ניתן להשתמש בסוגריים. כלומר, בדוגמאות למטה, אם מסירים את סימני הפעולות והסוגריים מ"הוכחה", מתקבלים המספרים לפי סדרם במערך המקורי. שימו לב: אין צורך למצוא ולהדפיס את הביטוי החשבוני שמתאים לערכי המספרים, או להוכיח שהוא קיים, אלא רק לומר אם קיים ביטוי כזה או לא.

דוגמאות:

```
isEquation({1,2,3,6}) -> true  
// proof: 1+2+3=6  
isEquation({1,2,3,9}) -> true  
// proof: (1+2)*3=9  
isEquation({1,2,20}) -> false
```

```

isEquation({1,2,-1,4,1}) -> true
// proof: 1 = (2*-1)+4-1
isEquation({4,6,1,2,5,2,2,6}) -> true
// proof: 4 = (6+1+2+5+2)-(2*6)
isEquation({1,5,8,3}) -> true
// proof (1*5) = (8-3)

```

```

public static boolean isEquation(int[] numbers) {
}

```

הדרכה: נסו לפתור את השאלה בשלבים, כך שתתחילו מלפתור עבור המקרה הכי בסיסי (שבו סימן השוויון נמצא במקום קבוע, ושימוש בפעולה אריתמטית בודדת) ולאחר מכן תתקדמו משם. ניתן לפתור שאלה זאת בעזרת שימוש ברקורסיה.

דרך אפשרית לפתרון:

1. ראשית, יש לקבוע את מיקומו של סימן השוויון
2. לאחר מכן יש לנסות ולבנות את כל הביטויים האפשריים על כל תת מערך (בצורה רקורסיבית).
3. אם מתקבלים ביטויים שערכם שווה, ניתן לעצור.
4. אחרת יש לקבוע מיקום שונה לסימן השוויון ולחזור על סעיפים 2-3.
5. אם בכל זאת לא הצלחנו, נחזיר false.

שימו לב: אין צורך לספק את הביטויים המספקים את השוויון (כלומר את ה"הוכחה") אלא רק אם ניתן או לא ניתן להגיע לפתרון.

חלק ד':

סודוקו היא חידה בה יש למקם ספרות על לוח משובץ שגודלו 9x9, המורכב מאזורים בגודל 3x3. מטרת המשחק - למקם את הספרות 1 עד 9 על גבי לוח המשחק כך שבאותו טור, באותה שורה ובאותו אזור לא תופיע אותה ספרה יותר מפעם אחת. חלק מהמשבצות בלוח כבר מכילות ספרות. היפרסודוקו (Hypersoduko) הינו סודוקו המכיל בנוסף עוד 4 אזורים צבועים (במקום קבוע כפי שמופיע בדוגמא למטה) שגם בהם מותר לכל ספרה להופיע פעם אחת בלבד.

חידת היפרסודוקו לדוגמא:

8	4	5	2				7	
9		7	4				1	
							4	5
1	2				7			3
	7				9		5	
	9			2	1			
		4		9				
		2	7	8				1
	1		3			8		6

ופתרונה:

8	4	5	2	1	3	6	7	9
9	3	7	4	6	5	2	1	8
2	6	1	9	7	8	3	4	5
1	2	8	5	4	7	9	6	3
4	7	6	8	3	9	1	5	2
5	9	3	6	2	1	7	8	4
6	8	4	1	9	2	5	3	7
3	5	2	7	8	6	4	9	1
7	1	9	3	5	4	8	2	6

החבילה hypersudoku אשר ניתנת להורדה מאתר הקורס מכילה תוכנית לחישוב פתרון חידות היפרסודוקו. החבילה מורכבת משתי מחלקות:

- GUI – אחראית על הממשק הגרפי למשתמש. מכילה את פונקציה ה-main של האפליקציה (כלומר, הרצת האפליקציה נעשית ע"י הרצת GUI.hypersudoku). המימוש של מחלקה זו נתון בשלמותו.

- Solution – אחראית על חישוב הפתרון לחידת היפרסודוקו. מחלקה זו מכילה את המתודה:

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

מתודה זו מקבלת מטריצה בגודל 9x9 של חידת היפרסודוקו כאשר כל כניסה ריקה מכילה את

הערך 1-. המתודה מחזירה true אם קיים פתרון לחידה ו- false אחרת. במידה וקיים פתרון

לחידה, המתודה תמלא את הכניסות הריקות בספרות בהתאם לפתרון. **אחרת, matrix לא**

תשונה!!! המתודה calcSolution נקראת ע"י המחלקה GUI. מימושה של calcSolution אינו נתון.

עליכם להשלים את מימוש המתודה calcSolution במחלקה Solution. אתם יכולים להיעזר בקטע

הפסאודו קוד הבא לפתרון נאיבי של חידת סודוקו. אלגוריתם זה מתעלם מבעיות יעילות ולכן ייתכן

כי הרצת תוכנית המבוססת עליו תיקח זמן רב:

Algorithm solveSudoku (Matrix m)

1. Verify that each digit appears at most once in every row, column, zone and hyper-zone.
If not - return false (no solution)
2. If there are no empty cells in m - return true.
3. Let $m[i][j]$ be an empty cell in m.
4. For $k=1$ to 9:
 - 4.1. $m[i][j]=k$
 - 4.2. if solveSudoku(m)==true - return true (recursive call)
5. $m[i][j] = -1$
6. return false

במידת הצורך ניתן להוסיף למחלקה Solution **מתודות עזר** חדשות.

הדרכה טכנית

המחלקה GUI משתמשת בספרייה חיצונית בשם SWT שכוללת גם קוד ג'אווה וגם קוד תלוי- מערכת הפעלה בשפת C. בגלל שהספריות הללו אינן מותקנות ביחד עם ג'אווה, צריך להתקין אותן כדי להשתמש ב-SWT. לצורך ההתקנה בצעו את הפעולות הבאות:

a. הורדת קובץ zip של ספריית SWT המתאים למערכת ההפעלה בה אתם עובדים מהאתר

<http://eclipse.org/swt/>

b. מתוך קובץ ה-zip העתיקו את קובץ swt.jar לתוך הפרויקט שלכם.

c. הוסיפו את ספריית swt.jar לתוך ה-build path של הפרויקט שלכם בעזרת לחיצה

ימנית על הפרויקט ולחיצה על properties ולאחר מכן Java Build Path וכאשר אתם

נמצאים בטאב Libraries בחרו Add Jars ובחרו את קובץ ה-swt.jar שהעתקתם לתוך

הפרויקט.

