

## משימה #5 בקורס תוכנה 1

### הוראות הגשה:

1. קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
2. הגשת התרגיל תעשה ע"י המערכת VirtualTAU (<http://virtual2002.tau.ac.il/>).
3. הגשת התרגיל תתבצע ע"י יצירת קובץ zip שנושא 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 התכניות שהתבקשתם לכתוב.
  - ג. קובץ טקסט עם העתק של כל קבצי ה Java.
  - ד. קובץ טקסט עם פתרונות לשאלות בהן לא נדרשתם לכתוב קוד

### 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).

מקרא:

$\text{\$prev}(x)$  – הערך של  $x$  (משתנה, ביטוי) לפני הקריאה לפונקציה. יכול להופיע רק ב  $\text{postcondition}$   
 $\text{\$ret}$  – הערך המוחזר

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

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

```
fix45({5,4,9,4,9,5}) → {9,4,5,4,5,9}
fix45({1,4,1,5}) → {1,4,5,1}
fix45({1,4,1,5,5,4,1}) → {1,4,5,1,1,4,5}
```

```
/*
 * @pre arr != null
 * @pre occurrences(4,arr) == occurrences(5, arr)
 * @pre arr[arr.length - 1] != 4
 * @pre forall 0 <= i < arr.length-2, arr[i] == 4 ==> arr[i+1] != 4
 * @post forall 0 <= i < arr.length-1, $prev(arr[i]) == arr[i]
 * @post $ret != arr
 * @post permutation(arr, $ret)
 * @post forall 0 <= i < arr.length-1, arr[i] == 4 ==> $ret[i] == 4
 * @post forall 0 <= i < $ret.length-2, $ret[i] == 4 ==> $ret[i+1] == 5
 */
public static int[] fix45A (int[] arr) {
    ...
}
```

```

/*
 * @post (arr != null AND
 *       occurrences(4, arr) == occurrences(5, arr) AND
 *       arr[arr.length - 1] != 4 AND
 *       forall 0 <= i < arr.length-2, arr[i] == 4 ==> arr[i+1] != 4)
 *       ==>
 *       (forall 0 <= i < arr.length-1, $prev(arr[i]) == arr[i] AND
 *       $ret != arr AND
 *       permutation(arr, $ret) AND
 *       $ret.length == arr.length AND
 *       forall 0 <= i < arr.length-1, arr[i] == 4 ==> $ret[i] == 4 AND
 *       forall 0 <= i < $ret.length-2, $ret[i]==4 ==> $ret[i+1] == 5)
 */
public static int[] fix45B (int[] arr) {
    ...
}

```

$occurrences(i, a)$  היא פונקציה המחזירה את מספר המופעים של  $i$  במערך  $a$   
 $permutation(a1, a2)$  היא פונקציה בוליאנית המחזירה  $true$  אם אברי המערך  $a1$  הם פרמוטציה של אברי המערך  $a2$ .

2. בהינתן מערך של מספרים שלמים  $(integers)$ , האם ניתן לבחור תת קבוצה מתוך המערך כך שסכום תת הקבוצה שווה למספר מטרה מסוים?  
 רמז: ניתן לפתור בעיה זו ברקורסיה. במקום לפתור עבור כל המערך, נפתור עבור המערך החל מאינדקס  $start$  ועד לסופו. נציין שברצוננו לפתור עבור כל המערך ע"י נתינת 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  $\sum_{i=0}^{nums.length-1} |nums[i]| < Integer.MAX\_VALUE$ 
 * @pre  $\forall 0 \leq i \leq nums.length-1, nums[i] \in \mathbb{Z}$ 
 * @post  $(\exists S \subseteq nums, \sum_{s \in S} s == target) \Rightarrow \$ret == true$ 
 * @post  $(\forall S \subseteq nums, \sum_{s \in S} s \neq target) \Rightarrow \$ret == false$ 
 */
public static boolean groupSumA(int start, double[] nums, int target) {
    ...
}

```

```

/*
 * @pre 0 <= start <= nums.length - 1
 * @pre  $\forall S \subseteq [0..nums.length - 1], \sum_{s \in S} nums[s] < Integer.MAX\_VALUE$ 
 * @post  $(\exists S \subseteq [0..nums.length-1], \sum_{s \in S} nums[s] == target) \Rightarrow \$ret == true$ 
 * @post  $(\neg \exists S \subseteq [0..nums.length-1], \sum_{s \in S} nums[s] == target) \Rightarrow \$ret == false$ 
 */
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) {

}

```

4. ניתנים כקלט שני מערכים של מחרוזות,  $a$  ו- $b$ , ללא כפילויות. החזירו את מספר המחרוזות המופיעות בשני המערכים. הפתרון צריך להיות יעיל ככל האפשר. במידה והמערכים ממוינים הפתרון צריך להיות לינארי – יש לעבור פעם אחת בלבד על המערכים. להלן כמה דוגמאות:

`sharedStr({"Call", "me", "Ishmael"}, {"Call", "me", "Jonha"}) → 2`

`sharedStr ({"a", "c", "x"}, {"z", "b", "c", "x", "a"}) → 3`

`sharedStr ({"a", "b", "c"}, {"a", "b", "c"}) → 3`

```

/*
 * @pre a != null AND b != null
 * @pre forall i, j; (0 <= i <= a.length-1 AND 0 <= j <= a.length-1) ==>
 *   ((i+1 == j) ==> (a[i] <= a[j]))
 * @pre forall i, j; (0 <= i <= b.length-1 AND 0 <= j <= b.length-1) ==>
 *   ((i+1 == j) ==> (b[i] <= b[j]))
 * @pre not exists i, j; i != j ==> a[i].equals(a[j])
 * @pre not exists i, j; i != j ==> b[i].equals(b[j])
 * @post |S| where S ≡ {s | ∃ i, j; s.equals(a[i]) ∧ s.equals(b[j])}
 */
public static int sharedStrA(String[] a, String[] b) {

}

```

```

/*
 * @pre not exists i, j; i != j ==> a[i].equals(a[j])
 * @pre not exists i, j; i != j ==> b[i].equals(b[j])
 * @post |S| where S ≡ {s | ∃ i, j; s.equals(a[i]) ∧ s.equals(b[j])}
 */
public static int sharedStrB(String[] a, String[] b) {

}

```

## 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.

1. עצרת

```
public static int factorial(int n) {
    int fact = 1;
    for (int i = 1; i <= n; ++i) {
        fact *= i;
    }
    return fact;
}
```

2. מיון

```
public static void sort(int[] arr) {
    int first, location, temp;

    for (first = 1; first < arr.length; first++) {
        if (arr[first] < arr[first - 1]) {
            temp = arr[first];
            location = first;

            do {
                arr[location] = arr[location-1];
                location--;
            }
            while (location > 0 && arr[location-1] > temp);

            arr[location] = temp;
        }
    }
}
```

## חלק ג':

בחלק זה נתרגל כתיבת מחלקות בהינתן תיאור של השירותים. עליכם לממש מחלקה בשם DisjointSets המייצגת קבוצה  $S$  של קבוצות זרות (disjoint)  $S = \{S_1, S_2, S_3, \dots, S_n\}$ . כאשר לכל  $i$   $S_i$  הינה קבוצת של מספרים שלמים אי-שליליים ומתקיים  $S_i \cap S_j = \emptyset$  עבור  $i \neq j$ . המחלקה מספקת את השירותים הציבוריים הבאים:

```
public class DisjointSets {

    /**
     * Add a singleton set (a set with a single element).
     * The specified value should not be an element of an already
     * existing subset.
     */
    public void addSingletonSet(int x) {
    }

    /**
     * Check if x and y are members of the same subset
     *
     * @return true if x and y are elements of the same subset;
     *         false otherwise.
     */
    public boolean inSameSet(int x, int y) {
    }

    /**
     * Join the subsets of x and y. Requires that x and y be members
     * of two different sets. The method replaces the two subsets
     * with their union.
     */
    public void joinSets(int x, int y) {
    }

    /**
     * Check if x is a member of some subset
     */
    public boolean inASet(int x) {
    }
}
```

דרך אפשרית למימוש המחלקה והדרך שבה הינכם נדרשים לממש היא ע"י ייצוג כל תת קבוצה  $S_i$  כעץ – כל איבר בתת הקבוצה יצביע לאב שלו בעץ כאשר השורש יצביע לעצמו. נשתמש במערך parent כדי להחזיק את תת הקבוצות. הערך בתא ה- $i$  במערך יהיה האב של  $i$  או  $-1$  אם הערך  $i$  לא קיים באף תת קבוצה. יש לדאוג שהמערך עצמו יהיה גדול מספיק כדי להכיל את כל הערכים הנוכחיים, לפיכך אם נוסיף ערך גדול יותר מהמקסימלי עד כה נצטרך להחליף את המערך הנוכחי במערך חדש גדול יותר. כמובן תוך שמירה על כל הנתונים מהמערך הישן.

כל אובייקט של המחלקה DisjointSets מייצג קבוצה של תת-קבוצות זרות, של שלמים אי-שליליים  $\{S_1, \dots, S_n\}$ .

נוכל להגדיר את המיפוי (פונקצית הפשטה) מ-parent, שדה של האובייקט, לסט של סטים  $AF(this) = \{S_1, S_2, \dots, S_n\}$  כך ש:

$$\exists i x \in S_i \Leftrightarrow \text{root}(x) \neq -1$$

$$x \neq y \Rightarrow x, y \in S_i \Leftrightarrow \text{root}(x) = \text{root}(y) \wedge \text{root}(x) \neq -1$$

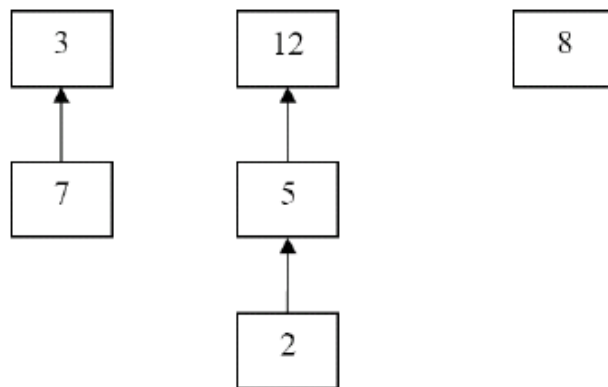
כאשר פונקציית העזר  $\text{root}(x)$  מוגדרת כ:

$$\text{root}(x) = \begin{cases} -1 & x > \text{parent.length or parent}[x] = -1 \\ x & \text{parent}[x] = x \\ \text{root}(\text{parent}[x]) & \text{otherwise} \end{cases}$$

למשל, לאובייקט של DisjointSets כאשר מערך ה parent הוא:

-1	-1	5	3	-1	12	-1	3	8	-1	-1	-1	12
0	1	2	3	4	5	6	7	8	9	10	11	12

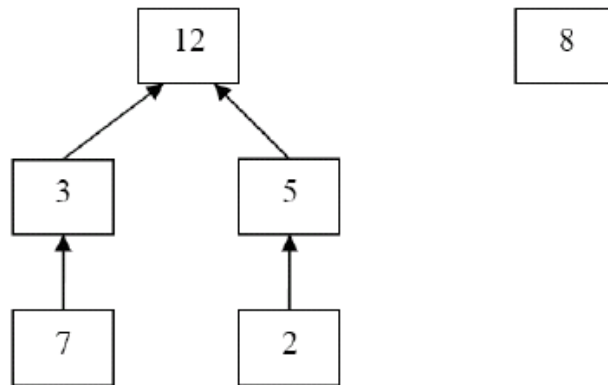
העצים יהיו:



ו-  $S = \{ \{3,7\} \{12,5,2\} \{8\} \}$ . אם נקרא ל- $\text{inSameSet}(3,5)$  נקבל false. אם נפעיל  $\text{joinSets}(7,5)$  נקבל לדוגמה את:

-1	-1	5	12	-1	12	-1	3	8	-1	-1	-1	12
0	1	2	3	4	5	6	7	8	9	10	11	12

והעצים יהיו :



ו- $S = \{ \{3,7,12,5,2\}, \{8\} \}$  אם נפעיל כעת `addSingletonSet(4)` נקבל ש-  
 $S = \{ \{3,7,12,5,2\}, \{8\}, \{4\} \}$   
הערה : באופן שקול ניתן היה לאחד את הקבוצות כך ש 12 יצביע ל-3.

### המשימה :

הורידו את קובץ `disjoint.zip` מאתר הקורס ויבאו את הפרויקט לאקליפס. השלימו את קוד המחלקה `DisjointSets`. שימו לב שניתן להגדיר שירותי עזר אולם אין להגדיר שירותים פומביים נוספים. בנוסף, הגדירו את תנאי הקדם והאחר לכל אחד מהשירותים וכן את משתמר המחלקה באם קיים. יש להגדיר גם את החוזה הפומבי וגם את משתמרי הייצוג היכן שדרוש.



## חלק ד':

נתונה המחלקה Rational המייצגת מספר רציונאלי (ללא מימוש). יש לציין את המצב המופשט (abstract state) של הטיפוס. כמו כן, עבור כל אחד מהשירותים יש לציין את משמעותו בעזרת המצב המופשט.

**אין צורך לממש את השירותים!**

```
public class Rational {  
    public int getNumerator() {...}  
    public int getDenominator() {...}  
    public Rational add(Rational other) {...}  
    public Rational subtract(Rational other) {...}  
    public Rational multiply(Rational other) {...}  
    public Rational divide(Rational other) {...}  
    public boolean equals(Rational other) {...}  
    public String toString() {...}  
}
```