משימה 3# בקורס תוכנה 1

מטרת התרגיל שלפניכם לתרגל כתיבת שרותים סטטיים לא טריוויאליים.
ברוב המקרים ניתן לפתור את התרגיל בקלות ע"י הוספת **פונקציות עזר** (אף שהתרגיל לא דורש זאת במפורש). בסעיף 7 אתם מתבקשים לספק **מימוש יעיל**, בכל שאר המקרים יש להעדיף את קריאות הקוד וכלליותו על יעילות.

1. Consider the leftmost and rightmost appearances of some value in an array. We'll say that the "span" is the number of elements between the two inclusive. A single value has a span of 1. Returns the largest span found in the given array. (Efficiency is not a priority).

   maxSpan({1, 2, 1, 1, 3}) → 4
   maxSpan({1, 4, 2, 1, 4, 1, 4}) → 6
   maxSpan({1, 4, 2, 1, 4, 4, 4}) → 6

   ```
   public static int maxSpan(int[] nums) {

   }
   ```

2. Return an array that contains exactly the same numbers as the given array, but rearranged so that every 3 is immediately followed by a 4. Do not move the 3's, but every other number may move. The array contains the same number of 3's and 4's, every 3 has a number after it that is not a 3 or 4, and a 3 appears in the array before any 4.

   fix34({1, 3, 1, 4}) → {1, 3, 4, 1}
   fix34({1, 3, 1, 4, 4, 3, 1}) → {1, 3, 4, 1, 1, 3, 4}
   fix34({3, 2, 2, 4}) → {3, 4, 2, 2}

   ```
   public static int[] fix34(int[] nums) {

   }
   ```

3. Return an array that contains exactly the same numbers as the given array, but rearranged so that every 4 is immediately followed by a 5. Do not move the 4's, but every other number may move. The array contains the same number of 4's and 5's, and every 4 has a number after it that is not a 4. Note that 5's may appear anywhere in the original array.

   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}

   ```
   public static int[] fix45(int[] nums) {

   }
   ```

4. Given an array of ints, is it possible to choose a group of some of the ints, such that the group sums to the given target? This is a classic backtracking recursion problem. Once you understand the recursive backtracking strategy in this problem, you can use the same pattern for many problems to search a space of choices. **Hint**: Rather than looking at the whole array, our convention is to consider the part of the array starting at index **start** and continuing to the end of the array. The caller can specify the whole array simply by passing start as 0. No loops are needed -- the recursive calls progress down the array.

   groupSum(0, {2, 4, 8}, 10) → true
   groupSum(0, {2, 4, 8}, 14) → true
   groupSum(0, {2, 4, 8}, 9) → false

   ```
   public static boolean groupSum(int start, int[] nums, int target)
   {...}
   ```

5. Given a string, return true if the number of appearances of "is" anywhere in the string is equal to the number of appearances of "not" anywhere in the string (case sensitive).

   equalIsNot("This is not") → false
   equalIsNot("This is notnot") → true
   equalIsNot("noisxxnotyynotxisi") → true

   ```
   public static boolean equalIsNot(String str) {

   }
   ```

6. Given a string, return a string where every appearance of the lowercase word "is" has been replaced with "is not". The word "is" should not be immediately preceded or followed by a letter -- so for example the "is" in "this" does not count. (Note: Character.isLetter(char) tests if a char is a letter.)

   notReplace("is test") → "is not test"
   notReplace("is-is") → "is not-is not"
   notReplace("This is right") → "This is not right"

   ```
   public static String notReplace(String str) {

   }
   ```

7. Start with two arrays of strings, a and b, each in alphabetical order, possibly with duplicates. Return the count of the number of strings which appear in both arrays. Provide a "linear" solution that makes a single pass over both arrays, taking advantage of the fact that they are in alphabetical order.

   commonTwo({"a", "c", "x"}, {"b", "c", "d", "x"}) → 2
   commonTwo({"a", "c", "x"}, {"a", "b", "c", "x", "z"}) → 3
   commonTwo({"a", "b", "c"}, {"a", "b", "c"}) → 3

   ```
   public static int commonTwo(String[] a, String[] b) {

   }
   ```

<u>הוראות הגשה:</u>

1. קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
2. הגשת התרגיל תעשה ע"י המערכת VirtualTAU (http://virtual.tau.ac.il). הוראות שימוש במערכת ניתן למצוא ב-http://virtual2002.tau.ac.il/upload/misc/main1.html
3. לנוחותכם תמצאו באתר בקורס **קובץ שלד** של הפתרון ובו חתימות הפונקציות הדרושות. http://courses.cs.tau.ac.il/software1/0708b/pdf_assignments/Assignment3.java. ניתן (ומומלץ) להוסיף פונקציות עזר במקרה הצורך.
4. בשל מגבלות מערכת ה Virtual Tau, יש לעטוף את קובץ ה java בקובץ zip.

בהצלחה