Project 3: Recursion

Introduction

For this project you will write a class called, "Recursion", that has the following public, **non-static methods**.

Do not change the name of the class, the signature or return type of the methods, or include packages.

Do not print from this class or take user input into this class.

You should then write a class called, "Main", that has the main method. In this class you should test your work as you were supposed to in project 2.

Requirements

Using Visual Studio Code:

1. int count22NoOverlap(String str)
   Given a string, compute recursively (no loops) the number of "22" substrings in the string. The "22" **substrings should not overlap.**
   Example:
   count11("22abc22") → 2
   count11("abc22x22x22") → 3
   count11("222") → 1

2. `int count22Overlap(String str)`
   Given a string, compute recursively (no loops) the number of "22" substrings in the string. The "22" **substrings can overlap.**
   Example:
   count11("22abc22") → 2
   count11("abc22x22x22") → 3
   count11("222") → 2
   count11("abc222222") → 5

3. `int factorsOf10(int[] array, int value)`
   Given an array of ints, compute recursively the number of consecutive elements that increase or decrease by a factor of 10.
   Example:
   factorsOf10([1, 10, 20], 0) → 1
   factorsOf10([100, 10, 20, 200], 0) → 2
   factorsOf10([1000, 100, 10, 1, 10], 0) → 4
   factorsOf10([10, 20, 33, 340], 0) → 0

4. `boolean balancedParens(String string, ArrayStack stack)`
   Given a string of a math expression, check to see if the parentheses are balanced.

   Balanced parentheses means that for each opening parenthesis, there is a closing parenthesis. Rewrite your stack method from project 2 so that it takes `char` types instead of `int` types.

   Algorithm:

   ```
   With each recursive call:
       If the character is not a parenthesis, ignore it and call recursively on the remaining substring
       If the character is an open parenthesis, push it on the stack and call recursively on the remaining
       substring
       If the character is a close parenthesis, check that there is an open parentheses on the stack
           If so, pop it off and call recursively on the remaining substring
           If not, return false
       If the substring is zero length then
           If there are still open parentheses on the stack.
               return false
           otherwise return true
   ```

   Example:
   balancedParens("(a+b) * c", stack) → true
   balancedParens("c", stack) → true
   balancedParens("((a+b) * c)", stack) → true
   balancedParens("(a+b) * c)", stack) → false
   balancedParens("(a+b * c", stack) → false

5. `void reverseArray(Object[] array, int index1, int index2)`
   Given an array of objects, this method will reverse elements of the array using recursion. You must not create any new arrays, work only on the array that is passed into the method. This is known as an "in-place" operation.

What to submit

- Copy *`Main.java`* and *`Recursion.java`* to a folder named, "<your_username>_p3" where <your_username> is replaced with your Bronco username.
- Zip the folder. It should produce a file called, "<your_username>_p3.zip"
  - E.g., if your username is jdoe, then the folder should be named *`jdoe_p3`* and the resulting zip file named *`jdoe_p3.zip`*
- Submit the zip file before the deadline.

Note: check your zip file before you submit it. If it's empty you will receive a zero for the project.
Also, do your own work, it would be terrible if you were failed from the class and reported to the school in the last week of the semester.