CS170
Homework 5: Loops, Recursion, and Arrays

Due: Thursday Mar. 24th by 9pm to BB.

Getting Started:
Make a directory for hw5 in your cs170 directory. Make a file for this homework named HW5.java and open it in your text editor.

The Assignment:
Write the following methods. Each method should be named as specified and follow the specifications for parameters and return type. Also be sure to comment each method! There is a test file provided on the class website named TestHW5.java. While it is not sufficient for completely testing your code, it will provide you with a basic check on the structure of your code. Place it in the same directory as your HW5.java file and compile and run it.

1. Removing repeated characters. The idea of removing repeated characters from a String is straightforward: remove any characters which are repeated multiple times in succession. For example, "Hello" would become "HeIo" when the duplicate 'l' is removed. Likewise, "aabbccddd" would become "abcd". Only remove duplicates when they are immediately next to each other. For example, "aabba" would become "aba".
   a) Write a method named removeRepeatsFor which takes a String as an input parameter and returns a String based on the description above. This method MUST be implemented with a for loop to receive credit. Methods implemented in other ways will not receive credit. Hint: make a new String which is "built up" character by character from the input String.
   b) Write a method named removeRepeatsWhile which takes a String as an input parameter and returns a String based on the description above. This method MUST be implemented with a while loop to receive credit. Methods implemented in other ways will not receive credit. Hint: make a new String which is "built up" character by character from the input String.
   c) Write a method named removeRepeatsRecursive which takes a String as an input parameter and returns a String based on the description above. This method MUST be implemented recursively to receive credit. Methods implemented in other ways will not receive credit. Hint: Examine a single character and the character next to it. Decide whether or not to include the first character. Then hand the remaining (unexamined) portion of the String off to the recursive method call to solve.

2. Write a method named commonNums which takes two integer arrays as parameters. The method should return the number of elements the arrays have in common. You may assume that each array will not contain duplicate numbers (in other words, an array like {1, 3, 3} will not be given as a parameter). Note, however, that the arrays can be of different lengths and the arrays could be empty (ie contain no elements). This method SHOULD NOT modify either of the input arrays. Examples:
   commonNums({1,2,3}, {3,2,1}) returns 3 (each array contains a 1, a 2 and a 3)
   commonNums({1,2,3}, {4,3,2,1}) returns 3 (same reason as above)
   commonNums({}, {1,2,3}) returns 0 (no numbers in common)

3. Now, you will implement portions of the Luhn checksum algorithm. Most e-commerce
websites these days take credit cards. Users must enter their credit card number and the merchant verifies that the number is valid. Then Visa, Mastercard, or AmEx process the payment to the merchant and pass the bill along to the user. However, users often mistype their credit card number by one or two digits. These common errors are why credit cards are designed with a secret. Using just the credit card number, we can detect (most) mistakes and errors caused by user mistakes/mistypes. The credit card number contains an error control code called a “checksum”. Specifically, the credit card number is formatted to comply with a Luhn-10 checking algorithm. For this method, you will write a method to calculate the checksum for a given credit card number.

The Luhn-10 algorithm is a weighted algorithm. Each digit in the credit card number is multiplied by a weight. These weights are then summed, forming the checksum. The checksum is divided by 10. If the remainder is 0, the credit card number is valid. If the remainder is NOT 0, the user made an error and can be prompted to re-enter their credit card data. The weighting for the Luhn-10 algorithm is as follows:

Beginning with the first (ie leftmost) digit in the credit card, every other number is multiplied by 2. If the product results in a 2 digit number (eg 6 x 2 = 12) then the individual digits (eg 1 and 2) are added to the checksum.

The remaining digits of the credit card number are simply added to the checksum. That is, their weight is 1.

Several (small) examples are given below, but this algorithm will work with your Visa or Mastercard number. Try it!!

Write a method named \texttt{luhnChecksum} which takes an array of integers as an input parameter and returns the integer checksum computed by the above algorithm. This method SHOULD NOT modify the input parameter array.

Examples:
\texttt{luhnChecksum\{4,5,6,3,9,2\}} returns 30 (see below for full calculation)
\texttt{luhnChecksum\{4,9,9,1,6,5,7\}} returns 40 (see below for full calculation)

\textbf{Example Number 1: \{4,5,6,3,9,2\}}

digit: \hspace{1cm} 4 \hspace{0.3cm} 5 \hspace{0.3cm} 6 \hspace{0.3cm} 3 \hspace{0.3cm} 9 \hspace{0.3cm} 2
multiplied by: \hspace{0.2cm} 2 \hspace{0.1cm} 1 \hspace{0.1cm} 2 \hspace{0.1cm} 1 \hspace{0.1cm} 2 \hspace{0.1cm} 1
\hline
product: \hspace{0.5cm} 8 \hspace{0.1cm} 5 \hspace{0.1cm} 12 \hspace{0.1cm} 3 \hspace{0.1cm} 18 \hspace{0.1cm} 2
checksum: \hspace{0.2cm} 8 + 5 + 1+2 + 1+8 + 2 = 30

Conclusion: This is a valid number since 30 \% 10 == 0!

\textbf{Example Number 2: \{4,9,9,1,6,5,7\}}

digit: \hspace{1cm} 4 \hspace{0.1cm} 9 \hspace{0.1cm} 9 \hspace{0.1cm} 1 \hspace{0.1cm} 6 \hspace{0.1cm} 5 \hspace{0.1cm} 7
multiplied by: \hspace{0.2cm} 2 \hspace{0.1cm} 1 \hspace{0.1cm} 2 \hspace{0.1cm} 1 \hspace{0.1cm} 2 \hspace{0.1cm} 1
\hline
Product: \hspace{0.5cm} 8 \hspace{0.1cm} 9 \hspace{0.1cm} 18 \hspace{0.1cm} 1 \hspace{0.1cm} 12 \hspace{0.1cm} 5 \hspace{0.1cm} 14
checksum: \hspace{0.2cm} 8 + 9 +1+8 + 1 + 1+2 +5 + 1+4 = 40

Conclusion: This is a valid number since 40 \% 10 == 0!
**Submitting:**

Note: be sure to include your collaboration statement. It is also important to include comments in your code. Failure to do either of these things will result in a point deduction.

Submit your work to the HW5 assignment on BB.