Recursion Exercises

Implement each method below using recursion. Test your methods with JUnit. YOU MAY NOT USE ANY LOOPS FOR ANY OF THESE.

1. `public static int triangle(int rows)`
   We have triangle made of blocks. The topmost row has 1 block, the next row down has 2 blocks, the next row has 3 blocks, and so on. Compute recursively (no loops or multiplication) the total number of blocks in such a triangle with the given number of rows.
   triangle(0) → 0
   triangle(1) → 1
   triangle(2) → 3

2. `public static int countX(String str)`
   Given a string, compute recursively (no loops) the number of lowercase 'x' chars in the string.
   countX("xxhixx") → 4
   countX("xhixhix") → 3
   countX("hi") → 0

3. `public static int sumDigits(int n)`
   Given a non-negative int n, return the sum of its digits recursively (no loops). Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).
   sumDigits(126) → 9
   sumDigits(49) → 13
   sumDigits(12) → 3

4. `public static String pairStar(String str)`
   Given a string, compute recursively a new string where identical chars that are adjacent in the original string are separated from each other by a "*".
   pairStar("hello") → "hel*lo"
   pairStar("xxyy") → "x*xy*y"
   pairStar("aaaa") → "a*a*a*a"

Much harder problem:

`public static boolean splitArray(int[] nums)`
Given an array of ints, is it possible to divide the ints into two groups, so that the sums of the two groups are the same. Every int must be in one group or the other. Write a recursive helper method that takes whatever arguments you like, and make the initial call to your recursive helper from splitArray(). Return true if it is possible, false otherwise.
splitArray({2, 2}) → true
splitArray({2, 3}) → false
splitArray({5, 2, 3}) → true