The recursive methods you are asked to write in this preview are a subset of the full Program #10 expectations. The complete Program #10 handout will be available in several days. All parts, including these, will be due when the full Program #10 is due.

Overview: The construction of recursive solutions requires a different mind-set than does the construction of iterative (a.k.a. looping) solutions. For most people, the best way to develop the recursive mind-set is practice, and lots of it. In this preview, we have three recursive tasks for you. The complete assignment will add several more.

Assignment: Write a complete, well–documented Java program named Prog10 and a class named Recursion (stored in separate .java files). Recursion holds a collection of static recursive methods, one for each of the following problems detailed below. Prog10 will call the recursive methods in Recursion for the purpose of testing them. (Yes, you need to write your own tests and test cases for these recursive methods.)

The first three (remember, more will be coming in the full version of the assignment) are:

1. Greatest Common Divisor (GCD)  (Method header: public static int gcd (int x, int y))

   The GCD of two positive integers is the largest integer value that divides both evenly. For example, the GCD of 12 and 15 is 3 (that is, GCD(12,15) = 3), GCD(7,14) = 7, and GCD(52,65) = 13. The general case of a recursive algorithm for computing GCDs is easily stated:

   \[ \text{gcd}(x,y) = \text{gcd}(y,x \mod y) \]

   Eventually, the remainder will be zero, and the value of \( y \) that produced the zero remainder is the GCD. That’s the base case of the recursion.

2. Range Sum

   (Method header: public static double rangeSum (double [] array, int lower, int upper))

   Given an array of double and two indices within the array (lower and upper), return the sum of the elements of the array from index lower through index upper. For example, consider this array:

   \[
   \begin{array}{cccccccc}
   0 & 1 & 2 & 3 & 4 & 5 & 6 \\
   7 & -2 & 4 & 0 & 8 & -1 & 2 \\
   \end{array}
   \]

   Based on this content, \( \text{rangeSum}(1,4) = 10, \text{rangeSum}(5,5) = -1, \) and \( \text{rangeSum}(6,5) = 0. \)
3. Reversing Alternate Characters

(Method header: public static String backSkip (String message))

The backSkip() method uses recursion to construct a String object containing every-other character of the given string, but in reverse order. Specifically, the returned string has the original’s last character, followed by the 3rd-to-last character, etc.

For example, if the call to backSkip() is:

   String aphorism = "Anything you do in life will be insignificant, but" + " it is very important that you do it. Mahatma Gandhi";
   String result = Recursion.backSkip(aphorism);

The content of result should be:

   ida maa t duyth ntom rvs itb,ncfnin bli fln duygitn

Data: We are not supplying any test cases for this assignment. The necessary parameters for each method are given above. Include with Prog10 an appropriate main() method that adequately tests your methods to ensure that they work correctly for all reasonable input values. You’ll want to choose your tests to demonstrate that those methods function correctly in a wide variety of situations.

Note that it is perfectly acceptable for Prog10 to be written with a testing method for each of the recursive methods, and to have main() call them in turn. That is, you don’t need to write a massive main() method.

Output: For each of the recursive methods, the expected return types are given with the method headers, provided above. The output of the execution of Prog10.java will be determined by how you write its main() method; make sure that your output clearly shows the parameters used and the results produced by each invocation of each method.

Turn In: For this preview, no submission is required, because you will be adding methods to Recursion and tests to Prog10 when the complete assignment is provided. Submission details will be provided with it.

Hints, Reminders, and Other Requirements:

- Because the recursive methods will be static and in the file Recursion.java, to invoke them from Prog10.java you’ll have to prefix the method name with the class name (which is also the file name). For example, to call gcd(), you’ll have to type Recursion.gcd(...), just as you use Math.sqrt(...) to call the square root method.

- As mentioned in class, it’s difficult to “sanity-check” arguments to a recursive method. But, it’s easy (if perhaps a bit rude) to throw exceptions! For this assignment, let’s be rude: If one of your recursive methods is called with an argument that’s invalid for that method, throw an IllegalArgumentException.

- Remember to ask yourself, “What’s (slightly) simpler than . . .?”

- Some of these recursive problems are well–known. There are lots of web pages with information about them. Thus, we offer this friendly reminder: Programming assignments in this class are to reflect your work, not that of another person. The penalties for turning in someone else’s work as your own are detailed on the class syllabus.