Java – Recursion with Euclid

**Purpose**

This lab was designed to teach you how to use recursion to solve real world problems.

**Description**

Design a class that will hold four static recursive methods. You should define your own class with a main method to test the recursive methods that you write in Euclid.java.

1. The **gcd** method takes two integer arguments and returns an integer. For the sake of clarity, we will refer to the two inputs as **a** and **b**. This method should check the base case (if b divides evenly into a) and if it meets this exit condition, return b. If not, return the result of the recursive call, where parameter 1 is b and parameter 2 is a % b. This function will execute a famous algorithm devised by the mathematician Euclid, in which you can determine the greatest common divisor between two integer values. For instance, let’s look at an example using the numbers 100 and 80.

gcd(100, 80)

100 is not divided evenly by 80, so we proceed to the recursive step

gcd(80, 100 % 80) == gcd(80, 20)

80 is divided evenly by 20, and so 20 is the greatest common divisor between 100 and 80.

1. The **geometric** method takes one integer argument and returns a double. This method will be used to compute a geometric series, and in particular, it will compute the series demonstrated below.

sum = 1 + 1/2 + 1/4 + 1/8 + 1/16 + 1/32 + 1/64 + …

Your method should compute this series starting with the smallest term and computing sums until you have reached the number 1 (), which is the base case. Test your method with a variety of inputs. This is a convergent series, which means that as you increase the number of terms, your result should get closer to a specific value. The integer argument for the method will represent the number of terms in the series. For example, an input of 4 will produce the series, 1 + 1/2 + 1/4 + 1/8.

1. The **basel** method takes one integer argument and returns a double. This method will be used to compute the Basel series, which was famously solved and proven by Euler in the 1800s to sum exactly to . This series is as follows:

sum = 1/ + 1/ + 1/ + 1/ + 1/ + …

1. The **scrambler** method takes one String argument and returns a String. This method will take an input String and scramble it so that it is unreadable to the user. The pattern it applies will also work to descramble a String that has been previously scrambled with this method.

The pattern used to scramble the String at each step is as follows:

1) If the String is length 0, return an empty String

2) If the String is length 1, return the String itself

3) If the String is length 2, return the String reversed

4) Otherwise, return a String constructed of the last character, a recursive call with characters from the second up to (but not including) the midpoint, a second recursive call with characters from the midpoint to the next to last character, and then the first character. An example is given below of the process.

Ex) scrambler(“computers”) =

“s” + scrambler(“omp”) + scramble(“uter”) + “c”

“s” + (“p” + scrambler(“m”) + scrambler(“”) + “o”) + (“r” + scrambler(“t”) + scrambler(“e”) + “u”) + “c”

“s” + (“p” + (“m”) + (“”) + “o”) + (“r” + (“t”) + (“e”) + “u”) + “c”

“spmorteuc”

**Program Shell**

Create a java class named Euclid.java