Lab #7 – Recursion

Total Points: 50

Introduction

In this lab you will be developing recursive definitions for several problems and then implementing those recursive definitions as Java programs.

Getting Started

Create your GitHub repository and import the code into Eclipse as described on the "How to…" page, available from the course Moodle page. Expand the `recursion` package.

The Assignment

Develop and implement and test a recursive solution to each of the problems described below. The `recursion` package contains a stub for each of the problems. You will complete the appropriate method(s) within the stub for each problem.

1. Pyramid Area

For this problem you will write a method that computes the number of boxes in a pyramid of boxes based on the number of boxes in the base. The boxes in each layer of the pyramid are stacked directly on top of one another (i.e. not offset between two boxes below). For example, below are shown two pyramids, one with base 5 and one with base 6.

```
    []              [][]
  [][][]          [][][][]
[][][][][]    [][][][][][]
```

Implement and test the `computePyramidBoxes` method in the `PyramidBoxCount` class. When testing the `computePyramidBoxes` method you should be sure to have at least one test method for each base case and at least one test method for each recursive case in your program.

2. Exponentiation

In the homework you were asked to give a recursive definition for computing x^n (where both x and n are non-negative integer values). The most straight forward recursive definition for x^n is:

$$x^n = 1 \text{ if } n == 0$$
$$x^n = x * x^{n-1} \text{ if } n > 0$$

However, the value of x^n can be computed far more efficiently using the fact that if n is even then:

$$x^n = x^{n/2} * x^{n/2}$$

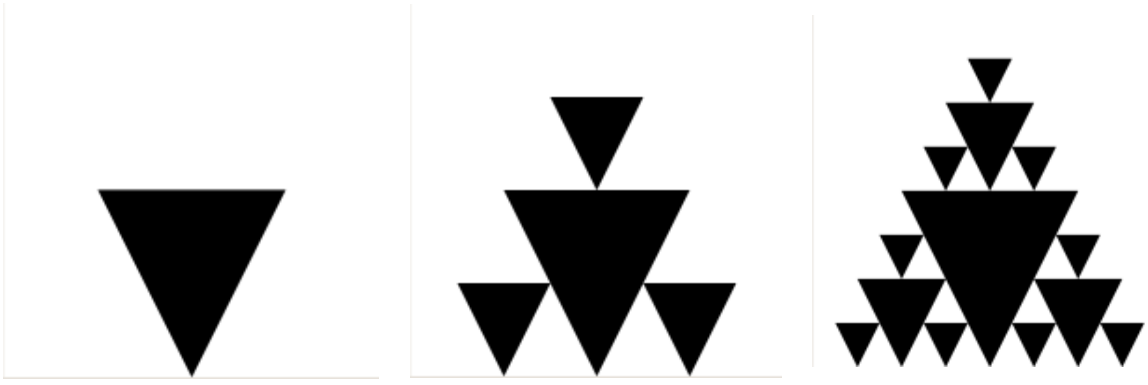
Implement and test the `exp` method in the `Exponentiation` class. When testing the `exp` method you should be sure to have at least one test method for each base case and at least one test method for each recursive case in your program.

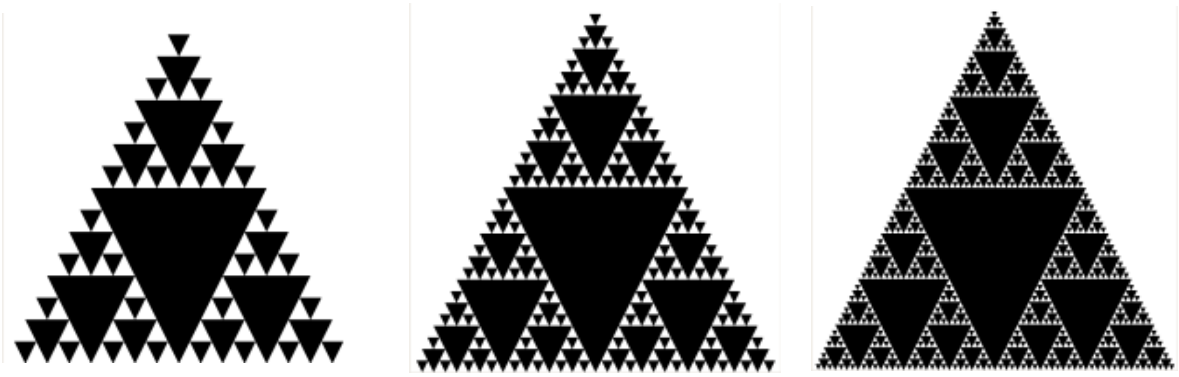
3. Array Sum

In the homework you were asked to give a recursive definition for computing the sum of all of the elements of an array. The `arraySum` method in the `ArraySum` class asks you to implement this definition in Java. For full credit, your solution to this problem may not make any copies of the array. You may find it useful to use a recursive transformation on this problem to eliminate the need to copy the array. When testing the `arraySum` method you should be sure to have at least one test method for each base case and at least one test method for each recursive case in your program.

4. SierpinskiTriangle

The Sierpinski triangle is another example of a fractal pattern. It was described by Polish mathematician Waclaw Sierpinski in 1915, but has appeared in Italian art since the 13th century. Though the Sierpinski triangle looks complex, it can be generated with a short recursive program. The Sierpinski triangles of order 1, 2, 3, 4, 5 and 6 (respectively) are:





The recursive structure of the Sierpinski Triangle can be seen by looking at the order 3 triangle. In the order 3 triangle there are 3 smaller order 2 Sierpinski Triangles arranged around an order 1 Sierpinski Triangle. Similarly, if we look at the order 4 Sierpinski Triangle, it has 3 smaller order 3 Sierpinski Triangles arranged around an order 1 Sierpinski Triangle. Also note that the triangles added for order 2 are half the size of the triangle of order 1. Similarly the triangles added for order 3 are half the size of the triangles added for order 2, and so on.

Implement the `drawSierpinski` method in the `SierpinskiPanel` class. To see your triangle run the `main` method `SierpinskiGUI` class, which will display a `JFrame` containing a `SierpinskiPanel` that draws an order 6 Sierpinski Triangle. You do not need to create a JUnit test for the `SierpinskiPanel` class.

NOTE: This problem is based on one developed by Prof. Tim Wahls.

Submitting your solution

As usual, push your code to GitHub regularly for backup purposes and push your final version to submit the assignment.