Exam 3 Sample Problems
CSCI 1101B – Fall 2014

1. (pencil and paper) Write a recursive Java method called `floorLog2` that accepts as input an integer `n` and returns the log (base 2) of `n` rounded down to the nearest whole number. Note that the value your method is required to return is just the number of times `n` can be divided by 2 (integer division) before the result is 1. For example, I can divide 25 by 2 four times before I reach 1:

   $$\begin{aligned} 25/2 &= 12 \\ 12/2 &= 6 \\ 6/2 &= 3 \\ 3/2 &= 1 \end{aligned}$$

   The method should work for all integers greater than or equal to one, and you may assume that the method is never called with an invalid input. **Caution:** Watch out for your Base Case(s). In particular, be sure your method returns the correct value for 1 (i.e. 0).

2. (pencil and paper) Write a *recursive* Java method (you will get no credit for a non-recursive method, even if it works) called `containsPattern` that is sent two `String` objects (`text` and `pattern`), and returns whether the given pattern is contained in the given text (i.e., true or false). For example, given the `pattern "here"` and the `text "This is a heresy!"`, `containsPattern` should return `true`. Given the same `pattern` and the `text "This is a green herb."`, the method should return `false`.

   You may not use any built-in `String` methods except `equals`, `length`, and `substring` (e.g., you may not use the built-in `contains` method of the `String` class).

3. (pencil and paper) Write a method called `getRange` that is sent an array of integers (i.e., an `int[]` object) and returns the range of the values contained in the array – i.e., the difference between the maximum and the minimum values.

4. (pencil and paper) Write a method called `getMatrixAverage` that is sent a 2D array of integers (i.e., an `int[][]` object) and returns the average of all values contained in the array. You may assume that every row of the 2D array has the same number of columns.

5. (computer) Write a class `Carpet` that draws a *Sierpinski carpet*. Think of drawing a black square and then (mentally, not actually) dividing it into 9 equal squares, like a tic-tac-toe board. A Sierpinski carpet is created by removing the center square (or, equivalently, drawing a white square in the center), and then doing the same thing recursively for the other 8 squares: See Figure 1 for an example showing several stages in the recursive construction (but ignore the white lines in the top middle stage after the first central square has been "removed" and ignore the different shades of gray in all the stages).

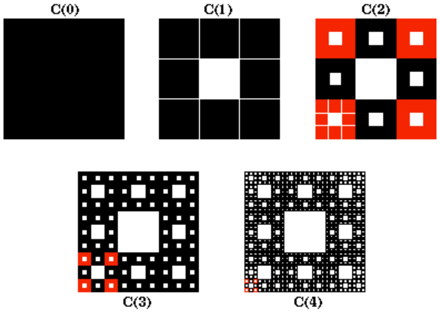

Figure 1: Figure 1: Creating a Sierpinski carpet.

Your class should include:

- a constructor that creates a Sierpinski carpet of a given size with its upper-left corner at a given location,
- a `contains` method that returns `true` if a `Location` object is contained in any of the *white* parts of the carpet; `false` otherwise, and
- a `move` method that moves the carpet by a specified `x` and `y` amount.

I have provided a project folder that includes an `Events` class that uses the `Carpet` class, and the skeleton of the `Carpet` class that also describes what you need to do. You will find this in the **Exams** section on Blackboard.

6. (computer) Write a method `isAnagram` that is sent two `String` objects and returns whether they are anagrams (i.e., the same set of letters possibly rearranged). For example, "anagram" and "granama" are anagrams. Note that the two sets of letters must be exactly the same to be an anagram, e.g., "banana" and "naban" are not anagrams, even though they both contain "b", "a", and "n".