COMP 1020 Winter 2016 Assignment 5

Due Date: Friday, Apr 8, before midnight

1 MATERIAL COVERED

- Recursion

2 NOTES AND INSTRUCTIONS—PLEASE FOLLOW *EXACTLY*

- Follow the programming standards posted on the course website to avoid losing marks.
- You must complete the "Blanket Honesty Declaration" checklist on the course website, before you can submit any assignment.
- To submit the assignment you will upload the required files, as specified for each question, to the Dropbox for Assignment 5 on the course website.
- Do **not** manipulate your output in any way.
- Complete instructions for handing in your assignment, including an easy way to create the output files, can be found in the "Handin instructions" document in the "General Information" folder on the website.
- To be eligible to earn full marks, your Java programs must compile and run upon download, without requiring any modifications.
- **Do not** hand in zipped or otherwise compressed files. Hand in **only and exactly** those files specified in this assignment, in the specified formats. Failure to do so will result in lost marks.

3 QUESTION 1: SIERPINSKI TRIANGLES

3.1 GENERAL DESCRIPTION

Write an `A5Q1` class containing a `main` method calling a recursive method to draw Sierpinski triangles (http://en.wikipedia.org/wiki/Sierpinski_triangle). Sierpinski triangles are equilateral triangles that can be drawn recursively. You will use the `StdDraw` class to draw the triangles.

3.2 THE DRAWTRIANGLES METHODS

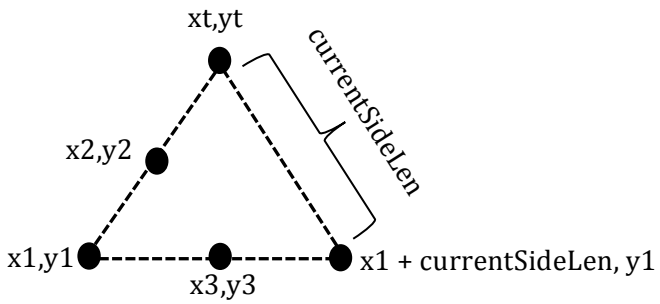
Define two `drawTriangles` methods, one non-recursive, and one recursive method. The non-recursive version simply calls the recursive version. The recursive version will draw a "level n" Sierpinski triangle. A "level 1" triangle is just an ordinary equilateral triangle. A "level n" triangle is made up of three smaller "level n-1" Sierpinski triangles, as defined below.

The recursive method has five parameters. The non-recursive method has four parameters (the Color parameter is not used). The parameters are described below:

1. The `int` value `n`, giving the "level" of the desired triangle, which should be 1 or more.
2. Two parameters of type `double` (`x1` and `y1`) which give the coordinates of the lower left corner of the triangle.
3. A parameter of type `double`, `currentSideLen`, specifying the length of the current triangle's sides.
4. One parameter of type `Color` (you need to import `java.awt.Color`) that is the fill color of a triangle.

For n=1 (the base case), a single equilateral triangle should be drawn with the given lower left corner, the given side length and color. The `StdDraw` class does not have a `filledTriangle` method, but it does have a `filledPolygon` method. Find some information about how that method is used (either online or by looking in `StdDraw.java`). For n>1, you must draw three smaller "level n-1" Sierpinski triangles, each with a different color (`StdDraw.BLUE`, `StdDraw.GREEN` and `StdDraw.BLACK` were used in the sample image below, but you may choose others). Their side lengths should be `currentSideLen/2` and their lower left coordinates should be as shown below:

1. The same `x1` and `y1`
2. The midpoint of the line between (`x1`,`y1`) and the top of triangle (`xt`,`yt`)
3. The midpoint between (`x1`,`y1`) and lower right corner of the triangle

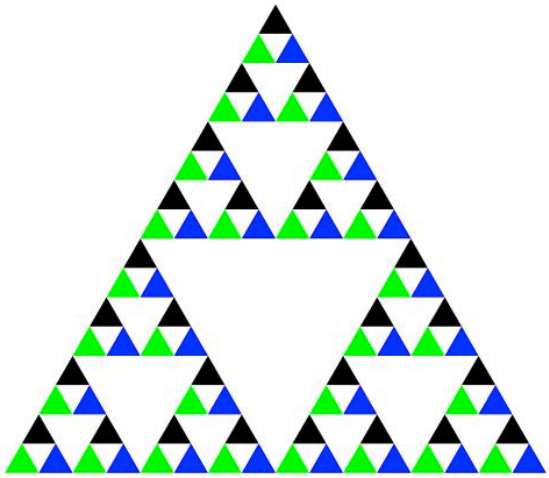


New coordinates can be calculated as follows:

- `xt` = `x1` + `.5` * `currentSideLen`
- `yt` = `y1` + `sqrt(3)` * `currentSideLen/2`
- `x2` = `x1` + `.25` * `currentSideLen`
- `x3` = `xt` = `x1` + `.5` * `currentSideLen`
- `y3` = `y1`
- `y2` = `y1` + `sqrt(3)` * `currentSideLen/4`

3.3 SAMPLE OUTPUT

A sample output with the initial call `drawTriangles(5,0.0,0.0,1.0)` will look like the following figure.



3.4 HAND-IN

Submit your `A5Q1.java` file (but **not** the StdDraw class) which will include a `main` method with a call to your recursive `drawTriangles` method that will draw the sample level 5 triangle shown above. Use the `save` command in the `StdDraw` window to save this image as `A5Q1-output.jpg` and hand in this file as well.

4 QUESTION 2: SHORTEST PATH IN MAZE

For this question, you will modify the provided `A5Q2Template.java` file to find the *shortest* path in a Maze (the input file to test the maze will also be provided). This is similar to the maze program that you saw in class (or soon will see). Rename the class to `A5Q2`.

First, you need to add the required code to explore *all* the paths in the maze and select the one with smallest length as the result. The `solve` method explores all possible paths, but does not 'remember' the shortest path seen. You will add that by updating a global variable called `bestSolution`.

You will also add a random element to the recursive search used in `solve`. The solve method always searches for the next move in the same order (as defined in the global variable `directions`). Instead of always starting at index 0 in `directions`, choose a different starting index randomly.

You should keep the `solve` method signature intact and *only* modify the body of the `solve` method. The locations where you should add or modify code are marked with comments.

4.1 SAMPLE OUTPUT

The text result of running the `A5Q2` program with the provided `TestMaze.txt` input file is shown here.

```
The path of size 21 is:
[(0,0), (0,1), (1,1), (2,1), (2,2), (2,3), (3,3), (3,4), (4,4), (5,4), (5,5), (6,5),
(6,6), (6,7), (7,7), (7,8), (8,8), (9,8), (9,9), (9,10), (10,10)]
```

4.2 HAND-IN

Submit your `A5Q2.java` file (but **not** the input file). Use the `save` command in the `StdDraw` window to save the final image displaying the shortest path as `A5Q2-output.jpg` and hand in this file as well.