CSCI 111, Lab 3
Geoffrey Matthews

Due date: Midnight, Tuesday, September 27, on Canvas. No late work accepted.

File names: Names of files and variables, when specified, must be EXACTLY as specified. This includes simple mistakes such as capitalization.

Individual work: All work must be your own. Do not share code with anyone other than the instructor and teaching assistants. This includes looking over shoulders at screens with the code open. You may discuss ideas, algorithms, approaches, *etc.* with other students but NEVER actual code.

grid: I wrote four solutions to the grid problem from the text, Chapter 3, Exercise 3. These files are available on github.com/geofmatthews/csci111 in the folder with the lecture notes for chapter 3, and called `grid00.py, grid01.py, grid02.py,` and `grid03.py`.

Modify each of these files so that they make a 4x4 grid of squares instead of a $2x2$ grid. Make as few changes to the files as you can.

Each line that is modified from the original should have a comment at the end of the line like this:

```
x = func(y)                # CHANGED
```

Each line that is new and not in the original should have a comment at the end of the line like this:

```
x = func(y)                  # NEW
```

Leave the filenames as `grid00.py`, *etc.*

Minkowski islands: Do Exercise 6, Chapter 5, in the textbook on the Koch curve and the Koch snowflake. Feel free to look at the author's solutions after you have done it yourself. You do NOT have to turn in your snowflake.

- Look up **Minkowski sausage** in Wikipedia. There you will find three generators for Minkowski sausages, illustrated in Figures 1, 2, and 3.
- Each of these can be used to make a **Minkowski Island** by drawing four of them around a square.
- Write a function for each of these generators, and place each in its own file:

  `koch1` in `koch1.py`

  `koch2` in `koch2.py`

  `koch15` in `koch16.py`
- Each function will take as arguments a turtle and a distance, as in the `snowflake` solution accompanying the textbook.

- The recursion is stopped when the distance is less than 10.
- Each generator uses a different reduction in size for the distance in the recursive calls. Figure out which distance will preserve the total length, regardless of how big it is.
- Write a single function `island`, that takes a turtle and a generator as arguments, and makes the turtle draw the generator around all four sides of a square. Islands look like Figures 4, 5, and 6.
- In each of your three files, define a turtle `bob`, set its speed to 0, and call `island(bob, 200,` *generator*`)` so that the island will be drawn when the module is run.
- Replace *generator* in each case with `koch1`, `koch2` or `koch16`, accordingly.

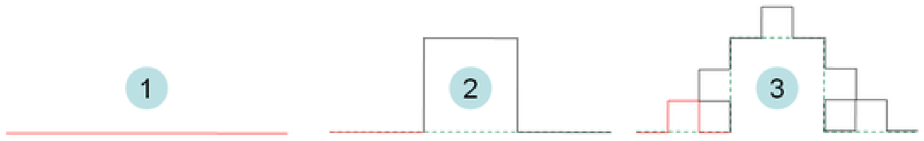

Figure 1: Koch type 1 generator

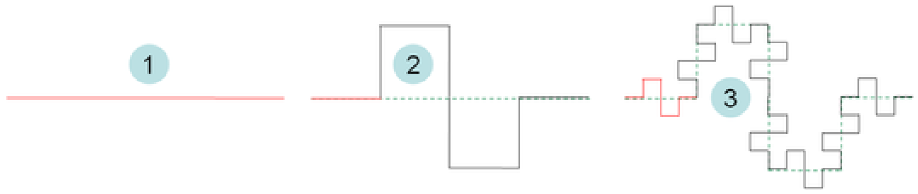

Figure 2: Koch type 2 generator

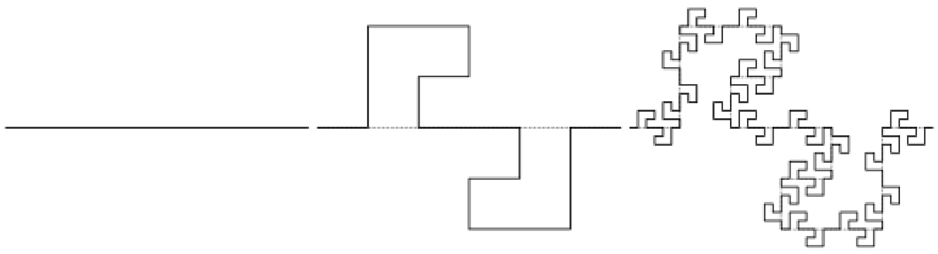

Figure 3: Koch type 1.6 generator

Turn in: Zip your four grid programs and your three island programs into a `lab03` folder and turn in on Canvas before next Tuesday.

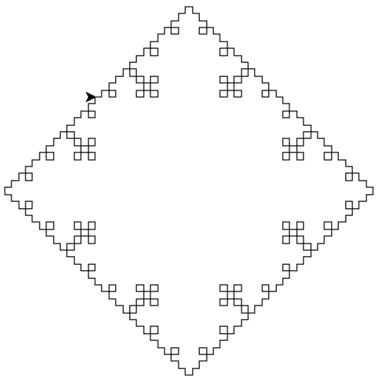
Figure 4: Koch type 1 island

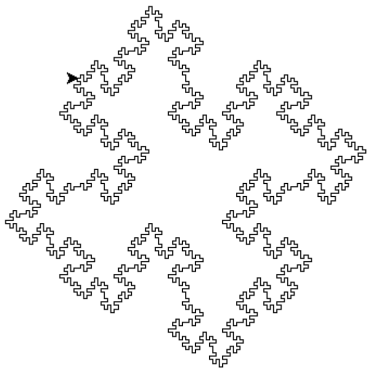
Figure 5: Koch type 2 island

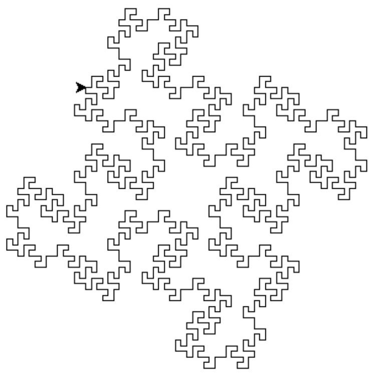
Figure 6: Koch type 1.6 island