National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2017/2018

Mission 5 - Side Quest
Dragonize

Release date: 24 February 2018
Due: 25 March 2018, 23:59

Required Files

- sidequest05.3-template.py
- hi_graph_connect_ends.py

Background:

Grandmaster Ben is delighted that his disciples have all made it thus far. He prepares for his final lesson on curves and also gives an encouraging speech to his disciples to prepare them.

"That's it, you are well on your way to wandless magic. Before we start on the final mind strengthening training, we must first analyse how your mind can work differently to get the same image. Yes you must find the fastest way to get that image. . . "

"With this in mind, you will now prepare for the final training - The Dragon Curve. Use all the techniques I have taught you earlier to get this right. Once you get it right, you will appreciate the beauty of telekinesis. . . "

Information:

The Python source file has been renamed to **`hi_graph_connect_ends.py`** and modified to include the function `connect_ends` which should have been previously coded. You may now use the function directly from this source file.

For your convenience, the template file **`sidequest05.3-template.py`** contains a line to load the Python source file **`hi_graph_connect_ends.py`**. Use the template file to answer the questions.

This side quest consists of **one** task.

Task 1: (4 marks)

A model of a Heighway dragon curve can be made by repeatedly folding a strip of paper always to the same side, and then unfolding it so that all angles are at 90 degrees, as shown in the figure below.

Figure 1: Dragon curve paper strip.

Figure 1 presents a way to obtain a dragon curve of order 4. A curve of order of magnitude 12 is presented in Figure 2.

Another fellow disciple, Yi Jie, is trying to write a function that would replicate this process in order to draw the dragon curve. He says, "Nothing easier, all I need to do to produce a curve of order n is connect the curve of order $n-1$ and its rotation by 90 degrees at their ends, and then put the resulting curve in standard position." So, Yi Jie comes up with the following function:

```
def yj_dragonize(num_points, curve):
    if num_points == 0:
        return curve
    else:
        c = yj_dragonize(num_points-1, curve)
        return put_in_standard_position(connect_ends(rotate(-pi/2)(c), c))
```

However, the function doesn't quite work. Yi Jie's brother, Yi Jiang, suggests that the following function should also be employed in the code:

```
def revert(curve):
    return lambda t: curve(1-t)
```

This function "reverts" the curve, in the sense that it traverses the curve in the reverse direction (the same curve would appear on the screen). To understand the difference, draw in separate windows the following two curves.

```
>>> connect_ends(unit_line, unit_circle)
>>> connect_ends(revert(unit_line), unit_circle)
```

(Note: use `draw_connected_scaled` to make the curves fit in the windows.)

Help Yi Jie to solve the problem.

Write a function `dragonize` that produces Heighway's Dragon Curve. The function should be similar to Yi Jie's, and it should also make use of the `revert` function. Test your code and make sure you can reproduce the drawing in Figure 2 with:

```
draw_connected_scaled(4096, dragonize(12, unit_line))
```

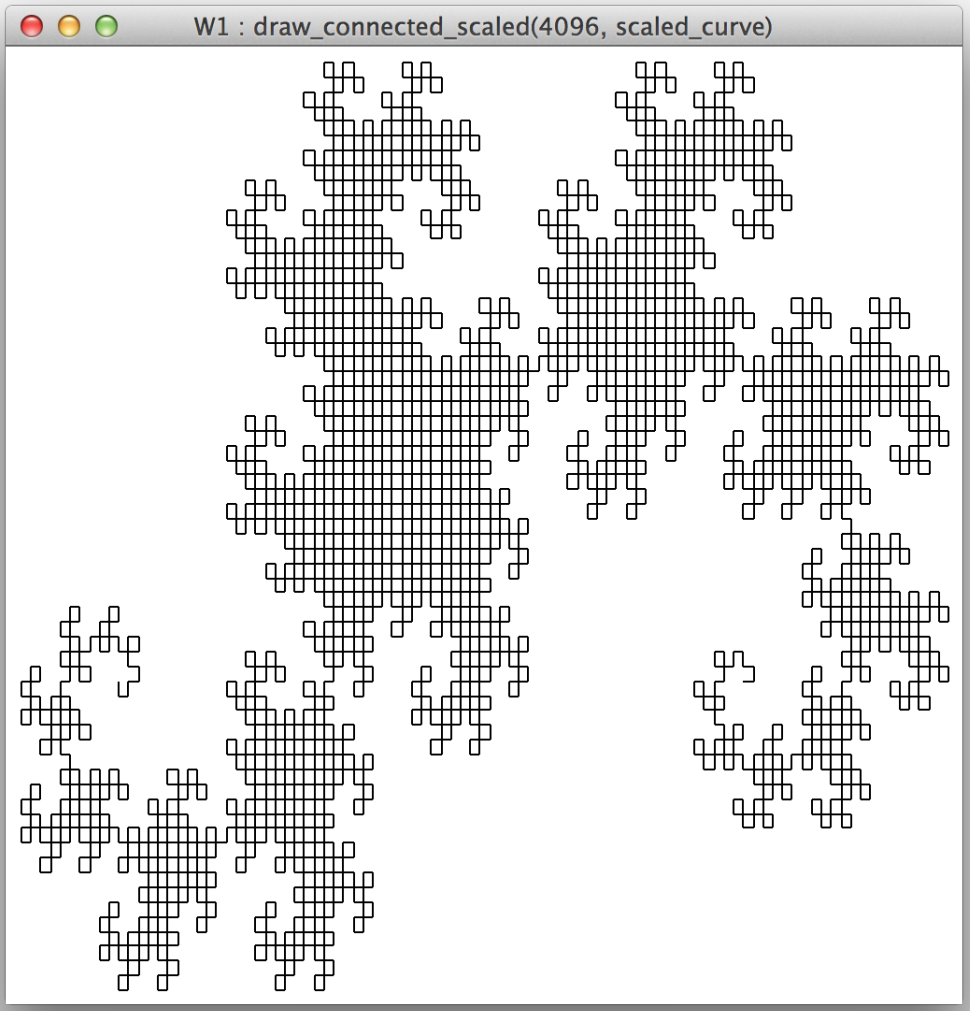

Figure 2: `draw_connected_scaled(4096, dragonize(12, unit_line))` displays the dragon curve above.