ECE-C301 Advanced Programming for Engineers
Instructor: James A. Shackleford
Programming Assignment 2

Part 1 – Problem Background

In this assignment you will create a virtual spirograph using the principles of object oriented programming. Figure 1 shows a physical spirograph creation setup consisting of a circular toothed track (red) within which the user would place toothed plastic wheels (blue) having pen placement holes placed slightly off the wheel center.

Figure 1: An "old fashion" manual spirograph toy.

The geometry of the toy can be represented as shown in Figure 2, where C is the center of the smaller circle, and P is the pen's tip.

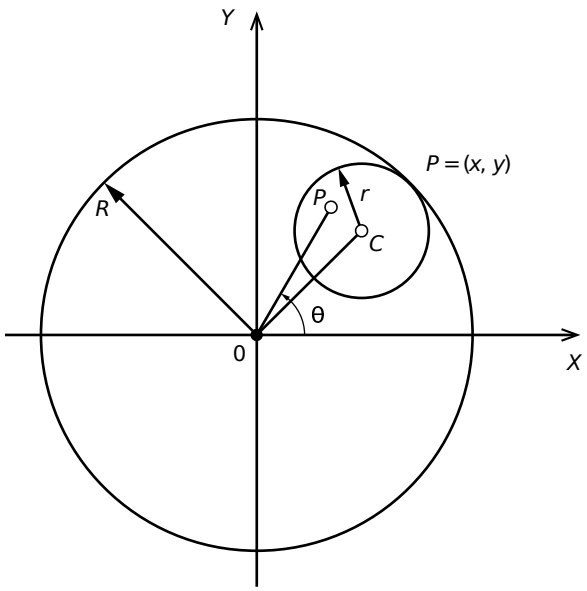

Figure 2: Geometry of the spirograph problem.

The radius of the bigger circle is R and the smaller circle r. For convenience, we will express the ratio between these two radii as k:

$$k = \frac{r}{R} \tag{1}$$

Let d represent the distance of the pen tip P from the center of the smaller circle C. Again, for convenience, we will define the variable l to represent distance of our pen tip from C normalized to within 0 and 1. Here, $l = 0$ places the pen tip at C and $l = 1$ at the edge of the circle):

$$l = \frac{d}{r} \tag{2}$$

We can now represent the coordinates of the pen tip as $\vec{P}(\theta) = [P_x(\theta), P_y(\theta)]$ via:

$$P_x(\theta) = R\left((1-k)cos(\theta) + lkcos\left(\frac{1-k}{k}\theta\right)\right) \tag{3}$$

$$P_y(\theta) = R\left((1-k)sin(\theta) - lksin\left(\frac{1-k}{k}\theta\right)\right) \tag{4}$$

A spirograph curve is complete when the starting and ending points of the continuous line forming the curve meet. If you have ever played with a real spirograph, you will know that the pen will need to move through multiple complete cycles of $\theta \in [0, 2\pi]$ before the final spirograph curve is complete. A simple way of doing this is to compute the periodicity of the spirograph curve by examining the ratio of the small inner radius r to that of the larger outer radius R.
Take, for example, the spirograph shown in Figure 3 generated using parameters $R = 220, r = 65, l = 0.8$. The radii ratio for this spirograph is:

$$\frac{r}{R} = \frac{65}{220} \tag{5}$$

The period is found by reducing this ratio and taking the numerator. For our example, this would be:

$$\frac{(65/5)}{(220/5)} = \frac{13}{44} \tag{6}$$

So, the period of the spirograph is 13 revolutions. Additionally, the smaller circle of radius r will revolve 44 times about its center, which will play a major factor in the final shape of the resulting curve. Take a moment to count the number of "petals" on the spirograph shown in Figure 3–it's 44.

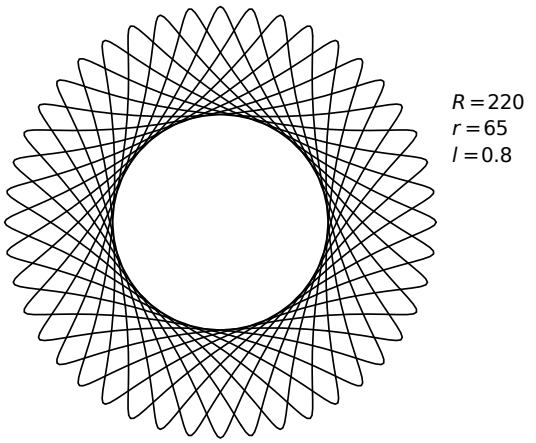

Figure 3: Example of a Spirograph curve.

Part 2 – Deliverables

For graphics operations, such as drawing, you will use the `Turtle` module. First lookup the `Turtle` module documentation before starting. Remember, you can always get the documentation for a module easily in Python's interactive mode:

```
$ python
>>> import turtle
>>> help(turtle)
```

You will write an importable Python module that implements `class Spirograph` having, at minimum, the following methods:

- `__init__(self, R)` – Create a new spirograph toy with outer radius R (i.e. the red part shown in Figure 1).
- `setSmallCircle(self, r)` – Set the radius of the small circle used to draw (i.e. the blue part shown in Figure 1).
- `setPen(self, l, color)` – Set the pen color and its distance from C
- `draw(self)` – Draw a spirograph using the current small circle and pen settings
- `clear(self)` – Reset the drawing surface

If you have written your program correctly, you should be able to draw multiple spirographs of different shapes and colors on the same drawing surface, for example, by using the following code:

```
import spirograph

# Create a new Spirograph toy with R = 500
my_spirograph = Spirograph(500)

# Draw one curve
my_spirograph.setSmallCircle(85)
my_spirograph.setPen(0.65, 'red')
my_spirograph.draw()

# ...and then draw another on top of the first
my_spirograph.setSmallCircle(120)
my_spirograph.setPen(0.22, 'blue')
my_spirograph.draw()

# ...and then get a new sheet of paper
my_spirograph.clear()

# ...and draw another
my_spirograph.setSmallCircle(20)
my_spirograph.setPen(0.8, 'purple')
my_spirograph.draw()
```

You will submit your complete code to BBLearn. This includes your `spirograph.py` module as well as another file named `test.py` which runs the above example that draws three superimposed curves in red, blue, and purple. The TA should be able to run your code without needing to download, move, copy, or modify additional files. Code that does not run will receive a zero. It is better to have code that runs and gives the wrong answer than code that does not run.

You will also submit a report detailing the project in PDF format. This report should include, at the minimum:

- an introduction detailing the mathematical background
- a detailed documentation of your code and its internal operation
- examples showing how somebody who has downloaded your code can use your `Spirograph` module to generate various spirographs (include color figures).