CS 116X/106 **Lab 06** **Winter 2015**

Procedural Content Generation

Useful Resources:

Instructions:

All parts of lab questions that have **R** at the beginning are mandatory, and their submission by each student is required. The parts of each question that begin with **O** are open-ended enhancements for further investigation into course material. These should be completed only after the mandatory parts of the lab are completed, and are not explicitly required for submission. If you choose to complete further optional add-ons, you are welcome to explore your own options, and not just the suggestions provided. Please follow the instructions on how to submit labs on LEARN. Save and submit files in the corresponding lab dropbox, within a folder of the same name:

- **L6_VisualiseNumbers**

Lecturette:
Why is recursion used instead of iteration?
What are some of the dangers of recursion (not using base cases, infinite recursion, less intuitive than iteration, etc…)
What is a prime factor (for question 2)?

---

1. Visualizing Prime Factors

You may remember the *fundamental theorem of arithmetic*, which says that every counting number can be written as the product of prime numbers. Let's gather up those prime factors and write them from largest to smallest. We end up with examples like 6=3*2, 40=5*2*2*2, 105=7*5*3, and 150=5*5*3*2. (Don't worry, it's not your job to calculate prime factors here.)

It turns out that every counting number can be visualized through a nested diagram of its prime factors. This web page has excellent animated examples:

http://www.datapointed.net/visualizations/math/factorization/animated-diagrams/

These drawings have a natural recursive structure. To draw 150, for example, we draw a ring of 5 evenly spaced sub-drawings, each of which is a copy of 30=5*3*2. In turn, each copy of 30 is a ring of 5 copies of 6=3*2, and each 6 is a ring of 3 copies of 2. We actually continue for one additional recursive step: each 2 is a ring of 2 copies of 1, and finally each 1 is just a circle.

a. **R** Open the starter code provided in `L6_VisualiseNumbers_StarterCode`. You will find a sketch that contains a simple user interface (courtesy of

`ControlP5`) that lets you enter an integer for visualization, and some helper functions that consume any integer and return an array of its prime factors sorted in descending order. (For the purposes of this lab, that code actually treats 4 as prime; this is a special case that makes the output look better.) Your job is to write a recursive function that draws a nested factor diagram within a unit circle, based on a number's array of prime factors. The base case is when the array is empty (i.e., it has length 0), in which case you should draw a unit circle (a circle of diameter 2). In the recursive case, let K be the current largest factor (the first element of the array). Use a `for` loop to draw a ring of K copies of the result of recursively calling the same function with the rest of the array passed in as an argument. You'll need to know the size of each of the K sub-rings to draw at the current recursive level (the `subRingRadius()` function will tell you this). That value will tell you how to `translate()` and `scale()` the geometric context around each recursive call.

Don't worry if your result doesn't look exactly like the diagrams in the link above. The most important thing is to get circular arrangements with the correct total number of dots.

b. **0** Update your code slightly to become more elegant. This could be done for example by having each dot a different colour, adding text to the screen, or even printing the prime factors of a number.