CIS 122 Summer 2019
Challenge Problems – Set 3

Challenge problems will not be guaranteed to follow our material exactly – meaning that solutions to these problems may involve concepts or techniques that we haven't seen yet, or may not formally cover at all. Additionally, these problems are neither required nor testable material.

This week's theme: Turtle challenges

1) Turtle clock

Write a program that uses turtle graphics to illustrate the face of a clock. Then, use Python's datetime module (import datetime) to get the current time and draw the hands correctly on the clock. Practice reading the official Python documentation to find what you need inside the datetime module: https://docs.python.org/3/library/datetime.html. Start by drawing the hands correctly for the time at which your program is run. If you really want to challenge yourself, consider how you might create a clock that updates its hands every second or minute.
Note: A useful turtle function is turtle.tracer(0, 0) which disables screen refreshing. Include that line before drawing anything, and then include turtle.update() at the point in your code when you're ready for the graphics to appear.

2) Recursive tree with turtle

Trees are a very recursive structure, because they contain a lot of self-similarity. If you cut a branch off of a tree, it itself looks a lot like the original tree, but smaller. For those of you who completed the recursion problems from last week, write a program that uses turtle graphics to recursively draw a tree shape.

In this case, the base case should be some minimum branch length after which we stop recursing. The recursive relationship here is basically that a tree can be thought of as a trunk plus two or more baby trees coming off of it. Each of which, in turn, can be thought of as a trunk, with two or more baby trees coming off of it, etc. etc.

The trickiest part is putting code in the right place to get the turtle back to the correct position before each recursive call, such that the baby tree that it is about to embark on will land in the right place.

This is a great way to visualize recursion, and understand the order in which the recursive calls are executed.

Once you have some recursive structure working, add some extra parameterization to make it change color or size, perhaps at each different level (for example, perhaps the pensize is proportional in some way to the branch lengths, growing smaller as we approach the "leaves").