Problem Set 2

This problem set is due **at 11:59 PM** on **Thursday, October 21, 2021.**

Please make note of the following instructions:

- This assignment, like later assignments, consists of *exercises* and *problems*. **Hand in solutions to the problems only.** However, we strongly advise that you work out the exercises for yourself, since they will help you learn the course material. You are responsible for the material they cover.
- Remember that the problem set solutions must be submitted on **Gradescope** as a pdf document. If you haven't done so already, please signup for QRMT FC-0309-1 (FCP) Foundations of Computer Programming on Gradescope, with the entry code 4PZV7W, to submit this assignment.
- Code files (.py files) are to be submitted in a zipped folder on **Classroom** with the naming convention **FirstName_LastName_Pset3.zip**.
- We require that your solutions are submitted as a single PDF file. Each submitted solution should start with your name, the course number, the problem number, the date, and the names of any students with whom you collaborated. Each problem should start on a new page (subparts need not follow this rule).
- The code file with the semi-complete functions required for problem *2-3* is uploaded on classroom.
- You may be called upon to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of your essay should provide the following:
    1. A description of the algorithm in English, step by step.
    2. Optionally, a proof (or indication) of the correctness of the algorithm. ("Why is this correct?" – we will explicitly ask for this on occasion). You may find it useful to include a worked example or diagram to show more precisely how and why your algorithm works.
    3. An analysis of the asymptotic running time behavior of the algorithm. (You'll learn what this is later in the semester.)

EXERCISES (NOT TO BE TURNED IN)

Culture

- Read up on Jackson Pollock and David Foster Wallace (you could also watch End of the Tour for the latter).
- For God's sake, please read Hamlet.
- Find out more about fractals. Think about why humans find them beautiful. (What exactly is aesthetically pleasing about it? What makes something aesthetically pleasing in the first place? Could you write a set of instructions to help an alien figure out what humans would find beautiful vs not?)
- Watch some movies / read some books with themes of recursion and themes of looping. Let me know if you find anything particularly interesting! Find something I didn't know about (e.g., Inception doesn't count).
- Check out `https://theuselessweb.com/`

Reading

In general, you should read everything that is posted on Google Classroom. You should be able to find everything listed below for free on the internet.

- Read as many articles as you can from Metamagical Themas by Douglas Hofstadter. Here's a link: `https://archive.org/details/MetamagicalThemas` I suggest you spend about an hour[1] on this. And yes, he did suggest we study memes.
- It's also worth reading a few pages from Godel, Escher, Bach by the same author. Link: `https://archive.org/details/GEBen_201706/page/n1`.
  Specifically, you should read the last three chapters (AI Retrospects, AI Prospects, and Strange Loops)
- Read up on Robert Nozick's experience machine. We will discuss this again near the end of the course, when we talk about AI.

[1] This means an hour of reading, not 5 minutes of reading and 55 minutes of YouTube.

Problem 2-1. We gotchu, Ron! [20 points]

Your friend Ron Measly (who is prone to following instructions in ways that are technically correct but not what was intended) is flying over a field carrying a portable cauldron of sickeningly tainting rainbow-colored potion. Unfortunately, the cauldron has a small hole in it. As Ron flies around, the leaky cauldron will cause his path to be tainted with potion onto the ground. Ron wants to draw a giant spiral[2] on the field but he doesn't know how to move to achieve that. Your instructions can only be simple things like "Fly forward x meters", "Turn y degrees to your right", or "Turn around", etc. Fortunately, Ron will execute a loop as long as it is written clearly in plain English.

(a) [10 points] What are your instructions to Ron? (Give an algorithm!)

(b) [10 points] What if Ron needed to draw concentric pentagons instead? How would you edit your algorithm above? (It is okay if there are lines of potion linking the concentric pentagons.)

Problem 2-2. Sort of Sorting [25 points]

In class we learnt about Insertion sort, Selection Sort, and Quick Sort algorithms. Use the power of the Internet to learn about 2 other kinds of sorts.

(a) [10 points] While doing so, try to learn about the different ways in which sorting algorithms can differ from one another (Look up terms like "stable sort" and "adaptive sort"). Using this knowledge, write 3 differences between the sorting algorithms you picked.

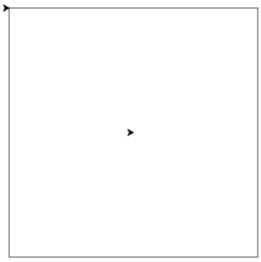
(b) [15 points] Come up with 2 real-life sorting problems. Out of the two sorts you learnt, explain which one would you pick to solve them and why (or why is that problem suited to your algorithm)? Now, solve the problems using the sort you picked i.e., explain the steps followed by the algorithm for your problem.

[2] You can choose how big, how many whorls, etc. But it does need to be a satisfactory spiral. Yes, that is vague. Life is unfair.

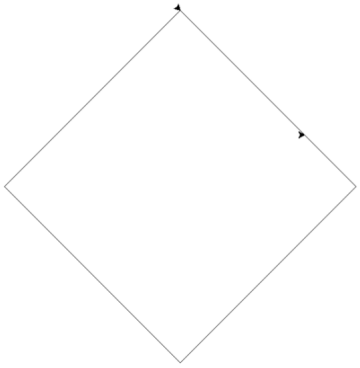
Problem 2-3. Be There, Or Be Square! [40 (+ 15 Extra Credit) points]

In this problem, you would be given incomplete pieces of code that make use of the turtle library[3] in python to draw figures. Your task is to add code to the marked lines and obtain the required output as shown in the figures

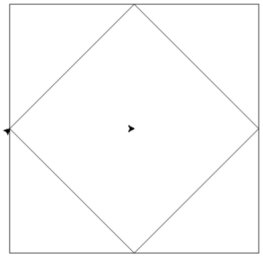
(a) [7 points] Complete the function *drawSquare* in file *Squares.py* to obtain a square.



(b) [7 points] Complete the function *drawTiltedSquare* in file *Squares.py* to obtain a - no points for guessing this - tilted square.

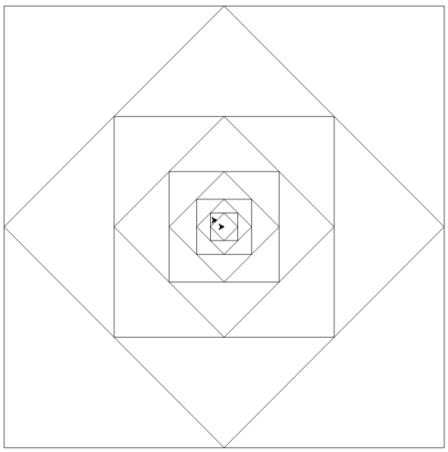


(c) [8 points] Now create a tilted square within another square. Complete the function *squareWithinSquare* in the file *Squares.py* to do this.

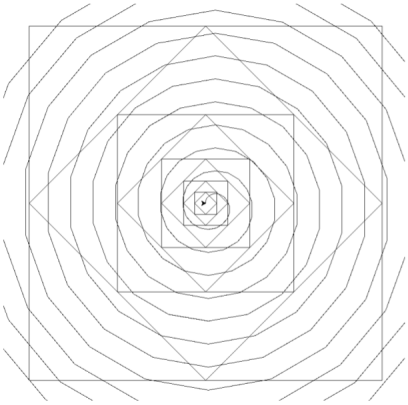


[3]Documentation for turtle library- `https://docs.python.org/3/library/turtle.html`

(d) [8 points] Use your existing knowledge about squares and tilted squares to create concentric pairs of squares and tilted squares. Complete the function *fractal* in the file *Squares.py* to do this.



(e) [10 points] You are now required to add a spiral to the output of the previous part. You have been given an incomplete function to draw a spiral in the file *Squares.py*[4].



Extra Credit [15 points]: A spiral is a curve revolving around its origin while moving farther away from it. You can also construct a spiral from semi-circles of increasing radii. Use this fact to construct a new spiral function whose radius increases with respect to the terms of the Fibonacci sequence and repeat **(e)** with this new function. What is the special name given to this spiral?

[4]For motivation: `https://www.youtube.com/watch?v=8YqvOLfkwq8`

Problem 2-4. Digital Infinity and The Computational Theory of Mind [20 points]

Throughout human history, we've always compared our own thinking to whatever was the most complicated thing we built. The Greeks compared thinking to catapults, the medieval Europeans to mills, etc. We graduated to cogs and wheels in the 1900s, and now, to computers. This time, of course (like all the other times), we think we're right. Are we? Currently, how do experts define "thinking"? Can a machine think? Why or why not? Do Androids Dream of Electric Sheep (Add this to your to-read list)? Include citations to things like Searle's Chinese Room problem etc.

If you haven't at least *skimmed* the reading, you're going to get a low grade on this question. You can start by reading this wikipedia article: `https://en.wikipedia.org/wiki/Digital_infinity` and branch out from there. (The usual path of starting from definitions of intelligence and artificial intelligence is boring.)