SHP Spring 2018
February 10th 2017 Assignment (Accelerated)
The goal of today's exercises is to try writing functions in programming games and learn how to write recursive functions. Try to finish at least one of your preference! No worries about doing all of them.

1. Program Mastermind using functions and encapsulation. The details about this game are provided on the assignments sheet for last week.

2. Recursive functions and Turtle Graphics

A turtle module for simple graphics has been included with standard distributions of Python, starting with Python 2. The "turtle" is a cursor that you can control to draw on a two-dimensional palette. The key of this approach is that you are the turtle and therefore all the drawing is done relative to the present position of the turtle. It could turn or move forward/backward but it can never go to a specific Cartesian coordinate.

All you'll need to know for this project about the Turtle module is the following:
First,

```
import turtle
from turtle import *
```

The "turtle" cursor in its window won't appear until you actually run the program. But remember that the default starting point is always a cursor in the center of its window facing the right side of the screen.

The methods you need here are:
`forward(`*`distance`*`)` : Move forward distance in current direction.
`backward(`*`distance`*`)` : Move backward distance in the opposite direction.
`right(`*`angle`*`)` : Turn right by angle degrees
`left(`*`angle`*`)` : Turn left by angle degrees
`speed(`*`speed`*`)` : You can also set the drawing speed as int in range 1–10 (1 slowest, 10 fastest)
`dot()` : mark current position with dot
`stamp()` : mark current position with a triangular shape

Before starting this project, I would suggest practicing the above commands several times.

Write a recursive function to create one of the diagrams below. (left: recursive tree, right: Sierpinski triangle) The number of stages (for e.g., 5 levels for the recursive tree) shouldn't be hard coded in the recursive program. But when you call it in the main function, test the recursive function by specifying the layers.

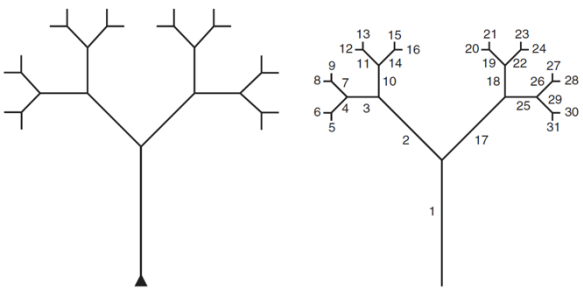

FIGURE 16.4 Recursive tree: (a) Python-drawn on left; (b) order-of-drawing on right.

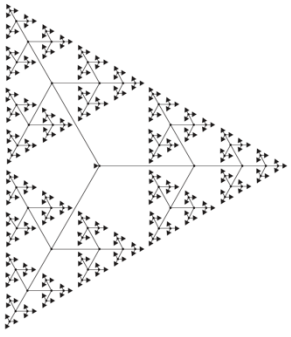
FIGURE 16.5 Sierpinski triangle.

IMPORTANT TIP:
For Python 2.7, it seems that when your turtle window doesn't pop up or creates an error when running the

program, it sometimes helps if you end the main function with `done()` to end the turtle window.
For e.g. your code should look like this for the recursive tree:

```
import turtle
from turtle import *

def branch(length, level):
    # base case

    # recursive calls


def main():
    #call to recursive function

    done()# finish program

main()
```

3. If you don't like any of the above, this is another option! Using functions for lists to solve the puzzle below:

Jim Loy poses this puzzle: I have before me three numeric palindromes (numbers that read the same backward and forward, like 838). The first is two digits long, the second is three digits long, and when we add those two numbers together, we get the third number, which is four digits long. What are the three numbers?

- Hint 1: All variations of numeric palidromes can be easily created by making strings of digits first.
- Hint 2: Use lists!