CS2321 Lab 10

Lab Instructions:
Save the code you write for each exercise in this lab as a *library* -- that is, a textfile with a .py extension containing only executable python code (i.e. no angle-bracket prompts, etc). Name each file according to the exercise number (e.g. ex1.py, ex2.py, etc.) and save them to a directory containing the report file (in PDF), when completed, compress them together in a single zip file to be submitted on D2L.

Each function should have a docstring explaining what the function does.
Any Follow-up Questions and their Answers should be included in a **docstring** following the `main()` function.

e.g. the structure for a Python module should look like:

```
'''
   modulename.py
   Doc-string explaining what this module does
'''
# imports, such as math, random, etc., as needed

# Your code, includes definitions of classes, functions, etc.
def ...
def ...
.
.
.
def main():
   # Do what is needed.


if __name__ == "__main__":
      main()

'''
   Doc-string answering follow up questions
'''
```

Lab Deliverable: Once all your programs run correctly, collect their code and the results of their test-cases in a nicely-formatted **PDF** file exported from Word Processing document (e.g. MS Word or LibreOffice) to be included in the submission on D2L.

This **report** should consist of each lab exercise, clearly **labeled** in order, consisting of code, then copy/pasted text output, or, for GUI, screen-captured, of its four test-cases.
In this lab, take series of screen captures of your turtle graphics and insert them into the report.

<u>Exercise</u>

All recursive algorithms must obey three important laws:

1. A recursive algorithm must have a base case.
2. A recursive algorithm must change its state and move toward the base case.
3. A recursive algorithm must call itself, recursively.

Here is the list of projects for this lab:

1. From page 183, #2: Write a recursive function to reverse a list.

2. From page 183-184, #3: Modify the recursive tree program using one or all of the following ideas:

    a. Modify the thickness of the branches so that as the branchLen gets smaller, the line gets thinner.
    b. Modify the color of the branches so that as the branchLen gets very short it is colored like a leaf.
    c. Modify the angle used in turning the turtle so that at each branch point the angle is selected at random in some range. For example choose the angle between 15 and 25 degrees. Play around to see what looks good.
    d. Modify the branchLen recursively so that instead of always subtracting the same amount you subtract a random amount in some range.

    If you implement all of the above ideas you will have a very realistic looking tree.

3. From page 184, #4: Find or invent an algorithm for drawing a fractal mountain. Hint: One approach to this uses triangles again. Or use the mid-point shifted upward a random amount.

4. From page 184, #5: Write a recursive function to compute the Fibonacci sequence. How does the performance of the recursive function compare to that of an iterative version? Use a graph to show the comparison of the time they take for the first 30 numbers.

Reference to Turtle graphics:
https://docs.python.org/3/library/turtle.html#module-turtle