Homework 4

Due: November 9, 11:59 PM

WARNING. Students may not work together. Students may discuss the problems with each other, but do not give any other student your solutions.

Aside: This file is written using markdown. markdown renders reasonably well inside pycharm. If you use pandoc, markdown can be converted to a pdf file.

Place all files containing your answers to homework 4 in a directory named `hw4_last_first` where `last_first` is the student's last and first names separated by an underscore. For me, the directory would be `hw4_harrison_david`.

Place each answer in its own file or directory. When you are done your directory structure should look like.

```
$ ls -F
hw4_harrison_david/
$ cd hw4_harrison_david
$ ls -F
p1/  p2/  p3/  p4/
```

Submit `hw4_harrisond_david.tgz` to blackboard. If you are on windows, you may use zip, in which case the file submitted would be `hw4_harrison_david.zip`.

Problem 1

Chapter 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?

The answer MUST have the followig form:

```
def fibo(n) -> list:
   """Returns a list wherein the ith member is the ith number in the
   Fibonnaci sequence"""
```

Problem 2

Section 5.17 Problem 11. Write a program that solves the following problem: Three missionaries and three cannibals come to a river and find a boat that holds two people. Everyone must get across the river to continue on the journey. However, if the cannibals ever outnumber the missionaries on either bank, the missionaries will be eaten. Find a series of crossings that will get everyone safely to the other side of the river.

Note: the solution must be recursive. The solution must output using the following form:

```
Near to far with X cannibals and Y missionaries.
Far to near with X cannibals and Y missionaries.
...
```

where X is the number of cannibals in the boat, and Y is the number of missionaries. There should be no extraneous output.

Problem 3

Section 6.16 Problem 1

Set up a random experiment to test the difference between a sequential search and a binary search on a list of integers.

Vary the size of the list of integers n from 1 to 1000. Repeat the experiment m times for each n and take the average run time per call, where $m = 100$. Output the average run times in a plot where the x-axis is n and the y-axis is the average run time per search.

Problem 4

Section 6.16 Problem 7

In the hash table map implementation, the hash table size was chosen to be 101. If the table gets full, this needs to be increased. Re-implement the put method so that the table will automatically resize itself when the loading factor reaches a predetermined value.

Modification: Double the size each time the load factor reaches a predetermined value. Plot the average run-time per call as a function of n showing the difference in performance when you allow the hash table to resize when it reaches a load factor of 0.5 vs. a load factor of 0.95.