Spellchecker using binary search, trees
Programming assignment due Friday, November 10

This week we will continue looking at file input and searches using the Spellchecker problem. I am assigning a series of short exercises / study materials with the following **goals**:

1. Be able to read a text file into a data structure, such as an array, BST, or AVL tree, using appropriate operations for that data structure.
2. Be able to determine execution time of operations
3. Understand data structures enough to be able to design evaluation methods for them, such as comparing runtimes, comparing data structures, counting memory-intensive operations, etc.
4. Be able to describe and implement a binary search of an array; a binary search tree; and an AVL tree
5. In C++, be able to identify and describe the difference between a struct, a .h file, a class implementation (.cpp) and an inner class

Exercises:

1. Write a program to read the text file into an array and then print out the array. Then, review the binary search algorithms.
2. Implement and test a binary search using the array in problem 1. Then, measure the execution time of reading in the array, printing the array, and searching for a word.
    a. Note: search for a word at different places in the array, such as the first element, last element, middle element, and a word that doesn't exist. Measure execution time and count comparisons for each.
3. Propose a solution that will create a more balanced, three, without using a different type of tree. Explain the scheme and provide pseudocode.
4. Implement a Binary Search Tree. Read the text file into the tree. [Note: what do you expect this tree will look like? What is it's time complexity?]
5. Measure the execution time of finding words in the tree. Use the same words as you used for the binary search implementation and compare execution time.
6. Count the number of levels in the BST.
7. Implement an AVL tree using the examples provided. Measure the execution time of reading in the file and searching for words. Count the levels.
8. Write a comparison of using an array/binary search, BST and AVL tree for the Spellchecker problem. The comparison should be based on your observations and study. It can include notes about efficiency and straightforwardness of implementation. It should be no more than a page. Cite your sources.

Deliverables:

1. Code for Exercises 1 and 2, above
2. Proposal and pseudocode for Exercise 3
3. Code for Exercises 4, 5 and 6, above
4. Code for Exercise 7
5. Comparison write-up for Exercise 8