CS 355 Assignment #2 Due: start of class, Wed. 10/17

This is an individual assignment. Do your own work. You may not discuss any aspect of the problem (design, approach, code) with anyone but the course instructor. Searching for code online is not doing your own work (submitting it as your work is plagiarism, a kind of cheating).

This assignment requires you to write a simple spell checker program. Your program should read a document stored in a text file and list all the different words contained in the file that don't occur in a second file that is considered a dictionary of valid words.

Read the document character by character, and create a string for each word in the document. Any sequence of alphabetic characters (length >= 1) on a single line should be considered a word. (Non-alphabetic characters or line breaks should not be considered part of a word, e.g., punctuation characters should be removed.) Words should be converted to all lower-case letters and each word should be stored in an instance of the dictionary/table ADT discussed in class. The ADT should be implemented using an AVL tree. Since words may occur more than once in the document, but the insert operation ignores duplicates, only the first occurrence of each word will be stored in the table.

After reading the document, read the dictionary. The dictionary is a text file with one word per line (easy to read). Don't store the dictionary words in a collection. As each is read, check if it occurs in your table of document words, and if so, delete it from the table. The words remaining in the table after going through the dictionary file are considered spelling errors. Write these words, one per line, to an error file.

INPUT:
Read the names of the document file, dictionary file, and error file from the user (in that order) and then process or create the specified files.

OUTPUT:
Write the number of distinct misspelled words in the document to the error file. Then list the misspelled words in alphabetical order, one per line.

Additional Requirements, Hints, and Suggestions:

A partial implementation of the Avl tree class is provided in the class folder. Complete the incomplete methods. **Do not make any changes to the header file except as instructed below**. Use the provided printTree method to help test your code. Currently the remove method does nothing. Modify this so that lazy deletion is used. (Hint: the private version of find should be useful in implementing this.) Lazy deletion also requires that you **add a field to the node class** to indicate whether a node is marked as deleted. Other methods must be modified to correctly take into account that some nodes in the tree store deleted values. You must modify the class interface to **add a public method called size** that takes no arguments and returns the number of (non-deleted) items in the tree. This should take constant time so you must **store and update the item count** when it changes so it can be checked quickly. You must also **add a public void method called traverse** that takes as its argument a void function that takes one argument of whatever type is being stored in the AvlTree (the class's template parameter). This function argument should be applied to each item in the tree in sorted order (using an inorder traversal). **No other additions/changes to the class interface are permitted**. Use the size and traverse methods to produce the output in the error file.

Extra Credit (15 pts): Provide more information in the error file. Give the total number of misspelled words, the number of distinct misspelled words, and a list (in alphabetical order) of each misspelled word, the number of times it occurred, and the page and line numbers where it occurred (assume 66 lines per

page). You still must read and store the document before reading the dictionary. Hint: for this to work, you'll have to define a struct or class type that lets you store more than just words in your table, and you'll have to update that information when duplicate words are found in the document, but without making changes to the dictionary ADT.