Data Structures Homework #5 Due: Saturday, April 3, 2010

This assignment has several parts -- a comparison of dictionary ADTs and a concordance-production application using the dictionary ADTs. Webster's definition of concordance is: "an alphabetical list of the main words in a work." In addition to the main words, I want you to keep track of all the line numbers where these words occur.

A dictionary ADT is a data structure that allows the programmer to access items by specifying key values. A real-world examples would be (1) Webster's English dictionary that allows you to use a word key to find the associated definition as the value, and (2) a phone book that allows you to use a name as a key to find the associated telephone number value.

DICTIONARY ADT

Operations (methods) on dictionaries	
size ()	Returns the size of the dictionary
empty ()	Returns true if the dictionary is empty
full ()	Returns true if the dictionary is full
contains (key)	Returns true if the dictionary contains the specified key
findItem (key)	Returns the value in the dictionary containing the specified key Precondition: key must be in the dictionary
removeItem (key)	Removes the item with the specified key
insertItem (key, value)	Inserts a new key-value pair Precondition: key must not already be in the dictionary
replaceItem (key, value)	Replaces the existing value of the specified key by the specified value. Precondition: key must be in the dictionary.

I want you to implement the dictionary ADT two ways:
1) use a binary-search tree (BST) implementation as the underlying data structure, and
2) use an AVL tree implementation as the underlying data structure. (you do not need to implement the dictionary `pop` operation since that would require an AVL tree delete operation. That means you need to implement the ADT `add` operation so that it replaces if the item already exists in the tree.)

WORD & LINE CONCORDANCE

The goal of this assignment is to process a textual, data file to generate a word concordance with line numbers for each word. A dictionary ADT is perfect to store the word concordance with the word being the dictionary *key* and a list of its line numbers being the associated *value* with the key. Since the concordance should only keep track of the "main" words, there will actually be two files of textual information: the data file to be processed and a stop-word file. The stop-word file will contain a list of stop words (e.g., "a", "the", "of", etc.) -- these words will **not** be included in the concordance even if they do appear in the data file. The stop words are to be stored in a second dictionary which will be accessed to ensure its contents do not appear in the concordance. Words in the data file are to be extracted, compared with the stop words, and, when appropriate, added to the concordance list along with the line number of the current occurrence of the word. Often, a word will be encountered several times--each line of encounter is to be recorded in the concordance list, but each word is to appear only once. The output of the program should be a text file containing the concordance words printed out in alphabetical order along with their corresponding line numbers.

Data Structures Homework #5 Due: Saturday, April 3, 2010

NOTES:
a) Words are defined to be sequences of letters that are delimited by any white space, punctuation, brackets, parentheses, dashes (two hyphens in a row), double quotes, etc. but not an apostrophe or single hyphens. For example, “it’s” and “end-of-line-characters” should be considered words.
b) There is to be no distinction made between upper and lower case characters, i.e., "ADT" is the same word as "adt".
c) Blank lines are to be counted in the line numbering.
d) The word-concordance application should not reference the dictionary ADT implementation, except through the dictionary operations.
e) It is strongly suggested that the logic for reading words and assigning line numbers to them be developed and tested separately from other aspects of the program. This could be accomplished by reading a sample file and printing out the words recognized and the lines they appeared on with no effort to avoid duplicates or associate words with more than one line.

Time the word-concordance application with both dictionary ADT implementations, and complete the following table:

Word-concordance Program	Execution Time (seconds)
Dictionary ADT implemented using a single BST	
Dictionary ADT implemented using a single AVL tree	

DATA FILES
The stop words are in the file
http://www.cs.uni.edu/~fienup/cs052s10/homework/stop_words.txt

The textual information to be examined is in the file
http://www.cs.uni.edu/~fienup/cs052s10/homework/hw5data.txt

HINTS:
0) The word-concordance application code and each dictionary ADT should be developed and tested separately.
1) A user-define or STL queue ADT may be useful to store the lines-of-occurence for each word.

SUBMISSION
You are to submit a single zip file containing:

- a one page overview of the design of your program, the above completed timing table, and directions for running your program, and
- all of your program files