

# CSE 2012 Fundamentals of Computing II

## Spring 2008

Lab handout for Week of March 31

### Objectives

1. Learn about trees and tries
2. Develop an efficient spellchecking program
3. Have fun!

### Prelab activities

1. Rest up and continue working on your projects
2. Read the wikipedia article on tries at: <http://en.wikipedia.org/wiki/Trie>

### In-lab activities

1. (1 point) Report to lab **on time**. Attendance will be taken at the scheduled lab time.
2. Tries, which come from the word retrieval, are tree data structures used to store an associative array. Unlike other data structures you may have encountered, the data (in this lab words) are not stored as specific elements, but rather as a path through the tree. This allows optimal searching even for very large databases, although it can be memory-inefficient at times.
3. Download the sample dictionary file and text file from the course website.
4. Develop a Trie class that stores the following information: the character of its incoming edge (char), if it is a leaf (bool), and a list of pointers to objects of this class type. This is different from a linked list in fundamentals I in that each node of the tree can have at most 26 links instead of the one (or sometimes two) of a linked list.
5. Read in words from the input text file and “clean up” these data by removing punctuation and converting all characters to lower case.
6. Add a default constructor to the node class that will correspond to the “root” of the tree. The incoming edge should be a space, it is not a leaf, and its pointer list should be empty.
7. Instantiate a node object in the driver program. This will correspond to the root of the trie.
8. Implement an “addWord” member function in the class Trie. This function will have two separate steps. The first will “search” the current Trie walking down the

tree until it cannot proceed further or the string has been found. The second is “add”, where the remaining characters should be placed into Trie nodes in a successive fashion. You are encouraged to use the STL list to hold pointers, and to use a member function “findNext” to find the pointer whose incoming edge is the character being searched for. Details were explained in class.

9. Implement a “findWord” member function in the class Trie. This will be identical to the “search” implemented above, except it will need to know if the word was found and if it was a leaf in the tree. For example, suppose the dictionary has the words “food” and “fool”, and others. A search for the word “foo” should return false because although foo and food/fool share a common prefix, foo is not a leaf (i.e., that specific word does not occur). In contrast, “i” and “inn” as found in the Wikipedia example should both return true because the internal node above the blue 11 is also a leaf (i.e., it does occur in the dictionary).
10. Combine the “addWord” and “findWord” functions to place the words in the dictionary file into a trie and check the words in the sample file if they are possibly misspelled. Your program should output each questionable word to the screen, one per line. Note: TAs will check your code based on the # of lines produced, so this is important.
11. (4 points) Flag down the lab TA and have them examine and check off that you have a basic understanding of what you will need in the postlab and you have made progress on the in-lab portion.

### **Post-lab:**

Write and submit (in your dropbox) a lab report with the following sections.

- (10 points) Code for your Trie class and the simple spellchecking program.
- (10 points) Submit a roughly one-page report on the following: 1.) What are the advantages (if any) of using your Trie as opposed to a binary search for small collections of known words? (2.5 points) 2.) How does this improve for very large collections of known words? (2.5 points) 3.) You have implemented a search and insert function. How would you develop a delete function? Feel free to use words or pseudocode for this question (5 points).

Code is expected to be well-commented, and the narrative portions of your report must be written professionally (i.e., complete sentences, correct grammar and punctuation, consistent tense and voice, formal tone).