EECS268:Lab10

From ITTC

Navigation

Home

Information

Syllabus
Schedule

Classwork

Labs
Homework
Submitting Work

Contents

- 1 Due time
- 2 Overview
  - 2.1 Starting Code
- 3 Dictionary Format
- 4 Requirements
  - 4.1 Phase 1
  - 4.2 Phase 2
  - 4.3 Implementation Details
- 5 Rubric
- 6 Emailing Your Submission

Due time

This lab is due **13 days** from the beginning of your lab.

(Sorry we can't make anything due past 11:59 p.m. the day before Stop Day).

Overview

In this lab you will make a dictionary program. You read in the words and their corresponding definitions from a file. Then, you will allow the user to query your dictionary in various ways.

Starting Code

Here is a starting point for your code. Many methods still need definitions.

- Download (http://people.eecs.ku.edu/~jgibbons/eecs268/2016spring/lab10/forLab10.zip)

Notes on starter code:

- You should still create a DictionaryEntry.h and cpp and add all needed methods (getter/setter)
- The sample main is just an example to show you that the BST will accept DictionaryEntrys as the ItemType
  - After you make the header and cpp for DictionaryEntry, remove the definition in main

Dictionary Format

I found a GIANT sample dictionary file online that contains over 70,000 lines of entries. While you're getting things working, I highly suggest you make a copy of a subset of the dictionary.

Here's the original (https://raw.githubusercontent.com/sujithps/Dictionary/master/Oxford%20English%20Dictionary.txt) format:

```
<letter>

<word> <article> <definition>
<word> <article> <definition>
<word> <article> <definition>
<word> <article> <definition>
...
```

Example of the original:

```
A



A-  prefix (also an- before a vowel sound) not, without (amoral). [greek]

Aa  abbr. 1 automobile association. 2 alcoholics anonymous. 3 anti-aircraft.

Aardvark  n. Mammal with a tubular snout and a long tongue, feeding on termites. [afrikaans]

Ab-  prefix off, away, from (abduct). [latin]

Aback  adv.  take aback surprise, disconcert. [old english: related to *a2]

Abacus  n. (pl. -cuses) 1 frame with wires along which beads are slid for calculating. 2 archit. Flat
```

Notes about reading in from file:

- I've randomized the entries the in the file and created an unsortedDictionary.txt (http://people.eecs.ku.edu/~jgibbons/eecs268/2016spring/lab10/unsortedDictionary.txt), but left in the markers for each letter. You'll need to ignore those markers
- We will lump everything after the word into the definition portion.
- Some words have multiple definitions, but we will again lump all of those definitions into a string to avoid this lab focusing too much on parsing.
    - So for "Abacus", the definition would be "n. (pl. -cuses) 1 frame with wires along which beads are slid for calculating. 2 archit. Flat slab on top of a capital. [latin from greek from hebrew]"
- There are duplicate entries, but a word should only be in your dictionary once
- Some words are, are well, two words, for example "Holy Grail"

Requirements

Since you'll have nearly two weeks to do this lab, we have broken it up into phases. The first table lists the functionality that you can accomplish by loading the dictionary entries into a single Binary Search Tree. Get everything from phase 1 working and tested before moving on.

Notes:

- The user will provide the name a dictionary file from the command line
- Using that file, create an load a BST full of dictionary entries.
- Until the user wants to quit, let them use your dictionary in the ways listed in the table below:
    - The table list the menus labels and desired outcomes, but you will need to the BinarySearchTree method that accomplishes the task
    - Example: If the user wants to **Search** you will need to call the **contains(key)** method from the BinarySearchTree class to verify whether or the word is in the dictionary

Phase 1

Search	Search the dictionary for a given word (case insensitive) and print it's definition
Save	Prompt the user for the following: ▪ Output file name ▪ Traversal order; The user can choose for the dictionary to be written in in, pre, or post order.

Phase 2

Copy	Creates a deep of the dictionary. The user can only edit the copy.
Add	Prompt the user for a new word, then new definition and add the entry to the tree. This can only be done to the copy.
Remove	Given a word, remove that entry from the dictionary. This can only be done to the copy.
Save	Prompt the user for the following: ▪ Output file name ▪ Which dictionary they want to save (original or copy) *Added for phase 2* ▪ Traversal order; The user can choose for the dictionary to be written in in, pre, or post order.

Test	You will add a Test Mode that we will use to test your copy constructor and destructor. When the user enters test mode, ask them which test they want to run.
	▪ void testAdds(BinarySearchTree<DictionaryEntry, std::string> dictionary) // NOTE: Passed by value!!!
	  ▪ Pass in the original dictionary
	  ▪ Prompts the user for a word and definition to add (seperate prompts)
	  ▪ Prints the remaining tree to the screen in in-order
	▪ void testRemoves(BinarySearchTree<DictionaryEntry, std::string> dictionary) // NOTE: Passed by value!!!
	  ▪ Pass in the original dictionary
	  ▪ Prompts the user for a word to remove
	  ▪ Prints the remaining tree to the screen in in-order
	▪ void testWriteToFile(BinarySearchTree<DictionaryEntry, std::string> dictionary) // NOTE: Passed by value!!!
	  ▪ Pass in the original dictionary
	  ▪ Prompt the user for an output file name
	  ▪ Writes the contents of the dictionary to the file in in-order

Quit	Exits the program.

Implementation Details

- We ask that your BST be templated with two types, and ItemType and a KeyType
  - The ItemType would be the type returned from a search
  - The KeyType would be the type that you can search on
  - For example, if I search for "Abacus" I should get back the whole entry for that word when I search.
- The BST should only use the default comparison operators, and **not be coupled** to the methods of any specific class
  - Overload the needed comparison operators for your class
  - We will order them alphanumerically in a case-inse
- If a copy is made and edits are performed, the original should remain unchanged.

Rubric

- 60% Dictionary Interaction
  - 10% Searching
  - 10% Deep copying of BST
  - 20% Removing entry
  - 20% Adding entry
- 20% File output
  - 10% proper traversal order
  - 10% well formatted output
- 10% Program stability
- 5% logical user interface
- 5% Needed Operators overloaded

Emailing Your Submission

Once you have created the tarball with your submission files, email it to your TA. The email subject line **must** look like "[EECS 268] *SubmissionName*":

> [EECS 268] Lab 0#

Note that the subject should be *exactly* like the line above. Do not leave out any of the spaces, or the bracket characters ("[" and "]"). In the body of your email, include your name and student ID.
