

Data Structures and Algorithms

**Lab 6: DSA\_Lab6.h**

**The Scenario**

Using your Hash Map/Dictionary from last lab, you’ve finished the underpainting of a beautiful masterpiece. It has everything from the dragon to the landscape. With all underpaintings, next comes the overpainting. You need a finer brush—and so you reach for your **Unordered Map** to start doing detail work. Your Unordered Map is nothing new—it is merely the nicely wrapped up version of what you previously used. It allows you to focus on the details, which is just what you need. You open Visual Studio and begin to paint.

**Unordered Maps** are the C++ Standard Template Library (**STL**) implementation of the Hash Map. The hash function, order of the buckets, the lists, and the arrays are all handled internally by the Unordered Map.

**Pairs** are a utility in the C++ STL. You’ll remember that last lab we created a struct named pair. The pair struct also exists within the C++ STL by default. Pairs simply let you collect two data types together. i.e. (key, value), <int, string>, or even the most dangerous of all: <auto, auto>.

**Scrabble** is a board game originally published in 1938. The game is centered around spelling words with tiles. Your ultimate goal is to achieve the highest score by spelling words worth the most points. Each letter has a point-value—i.e. ‘Q’ is worth 10 points, while ‘T’ is worth 1. The total value of a word spelled is calculated by adding up the points from each letter. (The word “Qualm” is worth 16 points. (Q:10 + U:1 + A:1 + L:1 + M:3).

Today’s lab will have you creating a Dictionary of legal words for a game of scrabble. If you wanted to play with only words found in *The Lord of The Rings: The Two Towers*, this program would let you do so—but for our purposes, we will use the lesser-known work, “words.txt”.

**What To Do…**

Open DSA\_Lab6.h. There will be instructions written in the comments on what is expected. Below is the gist of each function and variable…

***Variables:***

**mLetterValues** Used to store the values associated with each Scrabble letter. (If you’ve never played scrabble, each letter in a word has a point-value associated with it. Q is a less common letter, so it is worth 10 points, while A is a very common letter and worth 1 point. This array holds that information.)

**mScrabbleMap** This is your unordered\_map. It stores all known Scrabble words and their associated values.

**values** This is the array of letter values. You’ll use this in FindValueInMap once but nowhere else!

***Functions:***

**PopulateLetterValues** This method accepts an array of size 26 and populates mLetterValues based on it.

**GetLetterValue** This method returns the letter’s value. It can be assumed that the letter passed in will always be upper-case. (This poses a problem.)

**GetWordValue** Calculates the value of the string passed in and returns it.

**CreatePair** Returns a pair created based on the word passed in and its calculated score. Having trouble? Look at the return type.

**LoadWords** Loads a file of all possible Scrabble words and stores them in the map along with their value. The file loaded will have one word per line.

**FindValueInMap** This method will search for a word in your map, find it, and return the associated score. If a word is not a valid word (i.e. the word is not found in the map), it will then return -1.

**Note:** FindValueInMap will be the only method that uses the variable “**values**” and also must do the ‘setup’ of your program. i.e. loading “words.txt”, etc.

**Tips, Tricks, and Resources**

* Functions/Data Members available in the unordered\_map, fstream, and string classes ( i.e. myList.pop\_back( ) ) can be found on the Cplusplus.com documentation. The documentation for pairs is also included here, as you will have to use it for this lab:
  + <http://www.cplusplus.com/reference/unordered_map/unordered_map/>
  + <http://www.cplusplus.com/reference/fstream/fstream/>
  + <http://www.cplusplus.com/reference/string/string/>
  + <http://www.cplusplus.com/reference/utility/pair/>

**Plagiarism**

Plagiarism and Academic Dishonesty are considered a **very** serious offense in this class and can have a range of consequences including suspension, and in very serious cases, expulsion. If you either share your code or copy someone else’s code, you will be given a **0** on your lab and can face further disciplinary action.

In other words, don’t cheat please!