CptS 223 PA #3 – Dictionary for Hashing

In this task, you will implement a program that creates an English dictionary. You will implement a hash table to store the dictionary. The hash table will be indexed via the words in the English language. The dictionary will store the definition of the words for doing lookups after the dictionary is built.

Here are some of the key components of your implementation:

- Word class
  - holds a word
  - holds the word's definition
  - a key() function which returns an integer based upon converting the word to an int
    - See Figure 5.4 from the book for this algorithm
- Dictionary class
  - Implements a hash table (a vector)
  - The hash table will start with size 101
  - The Dictionary class will keep track of how many Words it holds (which is size N)
  - The hash table uses separate chaining to resolve collisions (Chapter 5.3)
    - Each chain is a vector holding Words
  - The hash table needs to implement:
    - void insert(Word)
      - Hashes the Word's key() and adds it to the hash table if it isn't already there
      - Updates (overwrites) old entry if there's a already the same Word in the hash table
    - bool contains(string word)
      - Searches the hash table by a string.
      - Will need to convert the string to a key, then hash it and search the vector of Words to see if
    - Word delete(string word)
      - Finds an entry using the passed in word, erases it (vector.erase()) from the hash table, then returns it
      - Returns nullptr if word is not in the table
    - void rehash()
      - Is called if the table gets above a load factor (N / Table Size) of 1.0 during an insert of a new Word
      - At least doubles the table size then finds the next prime number for the new table size (See figure 5.22 for an example)
      - Example code for finding the next prime above a given integer can be found at: https://gist.github.com/alabombarda/f3944cd68dda390d25cb

- Parse input file:
    - I have provided a file called: dictionary.json
    - This filename needs to be passed on the command line via argv
        - Example command line: ./a.out dictionary.json
        - That will make argv[1] be a char* to the string "dictionary.json"
        - Your program should detect if the filename was passed on the command line and print out help if the dictionary database wasn't specified. For an example of this, you can go to a Linux/OSX command line and just run "ssh" with no options. It will print out a usage message.
    - It is in JSON format and includes our English words and definitions, one per line
    - You may either write your own parser, or use a JSON parsing library

Once your Dictionary class is full of our Words, it needs to print out:

1) How many words are in the dictionary
2) The current table size
3) The current load factor

After that, it goes into query mode. This will output a prompt for the user such as the one below.

Word to define:

You should be able to enter a word, hit enter and have it either print out the definition or "Word not found" if it's not in the dictionary. I won't ask you to do fuzzy matching for misspellings like Google does, though. Note: the dictionary.json file's words are all uppercase, but users are going to be entering queries in whatever case they feel like. Make sure that Word == word == wOrD.

The program should exit if the user enters End of File (EOF), which is Control-D (^D). An example of detecting EOF can be found at:

http://www.cplusplus.com/reference/ios/ios/eof/

Testing

Your program will be tested first by running without passing in the dictionary.json filename on the command line – usage/help should be printed out. Then, it will be run with the dictionary.json filename. That should load the dictionary, and go into the query mode. I will be using a test file of words to search for. It should print out the words and their definitions or "Word not found" if it's an invalid search.

It is very easy to run on the command line with a pipe (or a redirect):

```
cat testcases.txt | ./a.out dictionary.json
```

The program should build the hash from the dictionary, then do the queries for the words listed in that testcases.txt file and quit.

Deliverables

You must upload your program through Blackboard no later than midnight on Friday, November 18, 2016.

Grading Criteria

Your assignment will be judged by the following criteria:

- [80] Your code compiles on the EECS servers using g++ -std=c++0x *.cpp, passes inspection with a set of test cases (words), and also handles outputs a usage message if it isn't given the dictionary.json filename on the command line (via argv)
- [10] Data structure usage. Your program only uses vectors, plus your Word and Dictionary classes.
- [10] Your code is well documented and generally easy to read.