IIT Ropar
CSL201 Data Structures
Semester 1, AY 2018/19

Lab Assignment 4 - 40 marks Due on October 18, 11:59 PM

Objective

To apply hash table design techniques on real-life data and study various performance measures.

Instructions

1. You are to use C++ programming language to complete the assignment.
2. Provide a Makefile to compile your final code.
3. All function and class declarations should be in ".hpp" files while the definitions should be in ".cpp" files. It is recommended that you have one header file and one cpp file for each class. If the class is templated, declare it in the hpp file itself.
4. This is an individual assignment. You can have high level discussions with other students, though.
5. Include a "Readme.txt" file on how to compile and run the code.
6. Upload your submission to moodle by the due date and time.
7. Late submission is not allowed. No assignment will be accepted through email after submission system closes. I suggest you to upload the assignment at least 1 hour in advance.
8. If you find assignment description is vague, take appropriate assumption and write that in the Readme. Clearly state assumptions and tell those assumptions during viva.
9. If any student asks for deadline extension, he or she will get 5% penalty and deadline will not be extended. Submissions through email won't be considered.
10. I should be able to use the class you have implemented in my program the way we use STL classes. There will be a master program to test the classes.

Program Description

In this assignment you need to implement a dictionary using hash table. The program should take an input file name as argument and add all the words in the file to the dictionary. The file format is as follows:
Word1 meaning1
Word2 meaning2
…..

In other words, the first word in each line is the “word” and the remaining text is its meaning. You have to use the “word” as the key of an entry. See files words.txt for example. The “entry” structure/class should store word and meaning separately, i.e., entry = (w,m). The hash table should store entry pointers.

Your program should support the following tasks:

- add(s): extract the key from s (the first word), create an entry, and add that entry to the table, if it is not already there; else update words meaning.
- remove(w): remove the entry with key k from the table, if it's there.
- size(): return the number of strings in the table.
- **spellCheck(w):** spellCheck(w) performs a spell check on the string w with respect to the set of words in the dictionary. If w is in the dictionary, then the call to spellCheck(w) returns its meaning. Otherwise, if w is not in the dictionary, then the call to spellCheck(w) returns a list of every word in dictionary that could be a correct spelling of w. Your program should be able to handle all the common ways that w might be a misspelling of a word including swapping adjacent characters in a word, inserting a single character in-between two adjacent characters in a word, deleting a single character from a word, and replacing a character in a word with another character.

You have to design a hash function to compute the hash code of a string. This function should be private. You also need to implement a constructor that accepts the size of the table as a parameter. Use open addressing with double-hashing for collision resolution. You have to justify all design decisions taken for hash functions and probe strategy during viva.

Program Interface

Design your own interface. The interface should allow user to all the operations described above. User should be able to choose one of the tasks and then provide input and get output.