Department of Computer Science

**CS260**
**Data Structures**

Fall 2021

# Programming Assignment 2

## 1 Regulations

**Due Date** : Friday, 10/22/2021 - 11:59pm
**Late Submission:** 10% off for each late day, with at most 2 day late submission
**Submission** : via Gradescope. See details below.
**Team** : The homework is to be done and turned in individually. No teaming up.
**Cheating** : All parties involved in cheating get zero from assignment and will be reported to the University.

## 2 Spell Checker

A spell checker is a software feature commonly found in text processors that checks for misspellings in a text. It is also a useful feature provided in most email and texting applications.
A basic spell checker compares each word with a known list of correctly spelled words from a dictionary. In the case that the dictionary contains only root words, spell checker can further compare for different forms of the word, such as plural/singular for names, past/present form for verbs etc. if the query word is not found in the dictionary as it is. If the word and its derivations are not found in the list, the word is flagged as mistyped. In several forms of the spell checker, the system further suggest alternative words to replace the mistyped word, by evaluating common typos. Finally, it can also suggest whether the user wants to add the word into the dictionary.
The performance of spell checker is crucial, where the major operation that is repetitively done is searching for words in the dictionary. Considering that the there are 171,146 words currently in use in the English language, according to Oxford English Dictionary, it will be very time consuming to search through the entire dictionary for each word in a provided

text along with its possible variations. Since there will be multiple search operations for each word of the text file, we need a data structure that will provide fast access to words in the dictionary. Hash tables are perfect tools for efficiently accessing dictinary elements!

## 3 Problem

In this assignment, you will write a C program that will implement a spell checker by using hash tables as the underlying data structure. You are expected to use Open Hash Tables in your implementation.

For this assignment, you will need to accomplish the following to get full credit:

- Load the dictionary into a hash table. You should use Open Hash Tables where you'll implement linked lists for each entry of the hash table.
- You will search for a given word in the dictionary and flag it as mistyped if it does not exists in the dictionary (i.e., no need to check for singular/plural, past/present tense etc. variations).
- If the word is flagged as mistyped, then check for potential typographical mistakes (inverted adjacent letters (ex: word vs wrod), incorrectly ordered letters (word vs wodr), words with a missing (word vs wor) or extra (word vs worid) letter), and suggest valid alternative spellings from the dictionary, if they exist.
- Ask the user if they want to insert the mistyped word into the dictionary, and insert it if prompted to do so by the user.

Your program should run and expect input from the user, until the user types "exit".

### 3.1 Deliverables

- Your code must be running on tux. So, make sure that it compiles and runs on tux before you submit, even if you develop it elsewhere.
- Your submission package will have your **source code**, a **makefile**, and a **report file written in pdf** that explains your approach for solving the problem. Put all these material in a folder named with your userid, (example: abc123), compress it into a zip file named as your student id (example: abc123.zip). Submit this zip file through Gradescope.
- Your **makefile** needs to produce an executable named **"check"**, once we type "./make" in command line. (see this page as to how to make simple makefiles.)

### 3.2 Specifications

- The program "check" is going to take **one command line argument**, which will be the name of the dictionary file. Thus, an example call to your program will be as follows:
  **./check dictionary.txt**
- **Input dictionary file** contains one English word in each line. A sample dictionary file is provided for your reference.
- Once run, your program should load the dictionary into an Open Hash Table, ask the user to enter text, and wait for their input, which is to be entered via standard input. You can use printf and scanf functions to interact with the user (see Tutorial on scanf for an example). You should print the following message to the screen and wait for user input:
  *Enter text to be spell checked:*
- Once the user writes a sentence as query and hits enter, your program will read the input words, and search for each in the dictionary.
  - If it finds a word in the dictionary, it will proceed to the next word until it processes all of the words in the sentence. It should not print anything if all the words are correctly written (i.e., found in the dictionary), and simply ask for the user to enter another input text.
  - For each of the words that it cannot find in the dictionary, your program should print the word to the screen and notify the user that the word is mistyped (using printf). For example, if the incorrect word was "spor", you should print:
    *Mistyped word: spor*
  - Your program then will suggest the user alternative words that might be the correctly spelled versions of the mistyped word. Checking for the following common typos would suffice for this assignment: inverted adjacent letters (ex: word

vs wrod), words with a missing (word vs wor) or extra (word vs worid) letter at the beginning and end. (although a real spell checker does more than that, implementing this much will suffice for the purposes of this assignement) For the incorrectly typed word "spor", your program should print something as follows:
*Suggested words for mistyped word 'spor': spore, sport, por*

  - Finally, you should ask the user whether they want to add the mistyped word into the dictionary, as the following yes/no question:
    *Would you like to add the word into the dictionary? [y/n]*
    If the user enters "y", you will insert the word into your dictionary that you stored inside the hash table, and ignore if the input is "n".

- Thus, you will be graded on whether you can;
  - search for an already existing word in the dictionary,
  - suggest alternative words that would be possible correct spellings of the mistyped word, and
  - add a new word to the dictionary.

- **Report file** should provide the following information:
  - Give an overview of the program on how it loads the dictionary, to what data structure. How did you implement your hash table? What is its size?
  - How does your program check for alternative spellings of the words?
  - What is the running time of your algorithm for a mistyped word? What is its running time for a correctly typed word?
  - How could you improve your spell checker, if you were developing it for Microsoft, rather than for CS260?

### 3.3 Grading

- Assignment is going to be graded out of 100 points.
- You will provide your self assessment for your effort for the 40 point portion of the assignment.
- 40 point portion of the remaining part will be for the test cases of the assignment.
- The final 20 point will be for the report file.