Department of Computer Science

CS260
Data Structures

Programming Assignment III

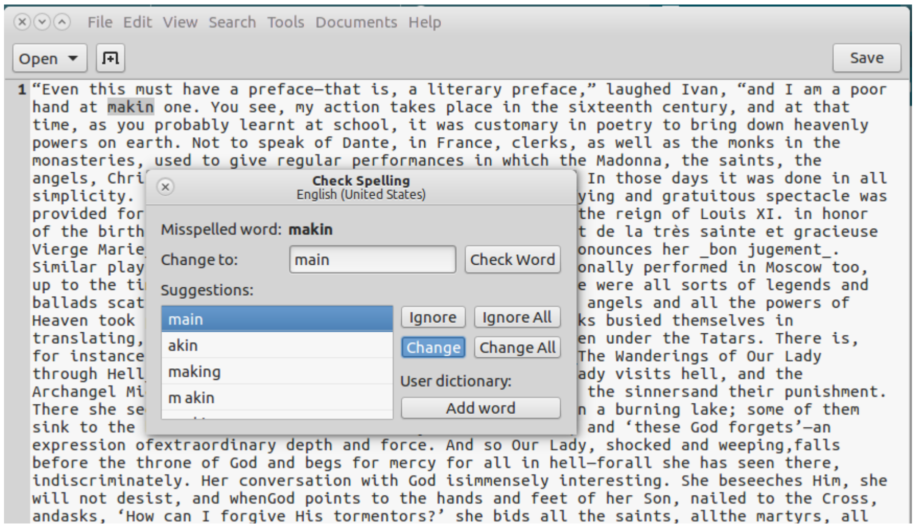
1 Regulations

Due Date	: *Check Blackboard*
Late Submission Policy	: *The assignment can be submitted at most 2 days past the due date. Each late submission will be subjected to a 10 point per day penalty.*
Submission Method	: *The assignment will be submitted via Blackboard Learn/Gradescope*
Collaboration Policy	: *The assignment must be completed individually.*
Cheating Policy	: *Do not use code from sources except lecture slides. "Borrowing" code from sources such as friends, web sites, AI tools for any reason, including "to understand better" is condiered cheating. All parties involved in cheating get a 0 for the 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 misspelled. In several forms of the spell checker, the system further suggest alternative words to replace the misspelled word, by evaluating common typos. Finally, it can also suggest whether the user wants to add the word into the dictionary.



```
“Even this must have a preface—that is, a literary preface,” laughed Ivan, “and I am a poor
hand at makin one. You see, my action takes place in the sixteenth century, and at that
time, as you probably learnt at school, it was customary in poetry to bring down heavenly
powers on earth. Not to speak of Dante, in France, clerks, as well as the monks in the
monasteries, used to give regular performances in which the Madonna, the saints, the
angels, Chri                                            In those days it was done in all
simplicity.                                            ying and gratuitous spectacle was
provided for                                           the reign of Louis XI. in honor
of the birth                                           t de la très sainte et gracieuse
Vierge Marie                                           onounces her _bon jugement_.
Similar play                                           onally performed in Moscow too,
up to the ti                                           e were all sorts of legends and
ballads scat                                            angels and all the powers of
Heaven took                                            ks busied themselves in
translating,                                           en under the Tatars. There is,
for instance                                           The Wanderings of Our Lady
through Hell                                           ady visits hell, and the
Archangel Mi                                            the sinnersand their punishment.
There she se                                           n a burning lake; some of them
sink to the                                             and ‘these God forgets’—an
expression ofextraordinary depth and force. And so Our Lady, shocked and weeping,falls
before the throne of God and begs for mercy for all in hell—forall she has seen there,
indiscriminately. Her conversation with God isimmensely interesting. She beseeches Him, she
will not desist, and whenGod points to the hands and feet of her Son, nailed to the Cross,
andasks, ‘How can I forgive His tormentors?’ she bids all the saints, allthe martyrs, all
```

```
Check Spelling
English (United States)
Misspelled word: makin
Change to: main
Suggestions:
main
akin
making
m akin
User dictionary:
```

Spellchecker of gedit, the default text editor of the GNOME desktop environment

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 dictionary 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. You can come up with any hash function. Refer to the slides for an example hash function to be used on strings.

For this assignment, your program needs to accomplish the following for you 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 misspelled 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 misspelled, then **check for potential typographical mistakes** (inverted adjacent letters (ex: word vs wrod), words with a missing (word vs wor) or extra (word vs wordi) letter at the beginning or at the end, and suggest valid alternative spellings from the dictionary, if they exist.
- If the user chose to insert the misspelled word into the dictionary (which user will tell when running the program initially), your program should be able to update the dictionary by inserting the new word into the hash table.

IMPORTANT: Do not use available C libraries (such as qsort() function or already existing hash or list data structures) that implement data structures and algorithms. You can use C library function rand() if you need a random number generator.

4 Starter Package

- Along with this assignment instructions document, you will be provided with:

  - a **starter code** that handles reading the input dictionary and the input text to be spell checked,
  - a sample **dictionary** file that contains one word each line, and a sample **text input** file that you can use for testing your spell checker,
  - a simple **makefile**, that compiles the code to generate an executable
  - an **executable spellchecker program**, which you can compare against your program to match input/output specifications.

5 Input/Output Specifications

- The **executable program** generated from your C code is supposed to take three command line arguments:
  1. the name of the dictionary file, which contains the correct writings of words
  2. the name of the input text file, contents of which you will spell check
  3. the word *add* or **ignore**, indicating whether to add misspelled words into the dictionary or not.

  Thus, an example call to your program will be as either one of the following. (assuming that you used the following command to compile your code that was written inside main.c: "gcc main.c -o check", which will produce an executable named 'check')

  ```
  $pa3/> ./check dictionary.txt inputText.txt add
  $pa3/> ./check dictionary.txt inputText.txt ignore
  ```

- **Input dictionary file** contains one English word in each line. A sample dictionary file is provided for your reference in the starter package.
- **Input text file** contains some plain text written in English. A sample input text file is provided for your reference in the starter package.
  - Note that, since the dictionary that we will use contains words that exclusively start with lowercase letters, you can assume that the input text will always consist of lower case letters.
  - Also note that, the only non letter characters in the input test will be white space, comma, dot, colon, semicolon, exclamation mark, and new line. The provided starter code includes a line that splits the input text into words by these delimiters.
- **Updating the dictionary** is an all-or-none behavior for your program. If the user has run your program with the third command line parameter being **add**, then you should insert all misspelled words that your spellchecker detects into the dictionary. On the other hand, if the user has run your program with the third command line parameter being **ignore**, you shouldn't insert any new word into the dictionary.

  Thus, as an example, if the input text contains the same misspelled word twice and the program was run with *add* parameter, you should not tag the second appearance of the word as misspelled, as that word would have been inserted into the dictionary after its first appearance in the text, making the second appearance a valid word. On the other hand, if the program was run with *ignore* parameter, then each appearance of the same misspelled word should be reported (details of which will be listed below).
- Once the user enters these three command line parameters and runs your program, your program should load the dictionary into memory, read the input text word by word, and **make a search for each word** in the dictionary.
  - **If it finds the word in the dictionary**, it will proceed to the next word until it processes all of the words in the input text. If the input text does not have any misspelled word (i.e., all words in the input text are found in the dictionary), your program should print:

    No typo!

    to standard output (with a new line after exclamation mark) and quit the program.

  - For each of the words that are **not found in the dictionary**, your program should print the word to the screen and notify the user that the word is misspelled. For example, if the incorrect word was “spor”, you should print:

    Misspelled word: spor

    to standard output (with a new line after the misspelled word).
  - For the words that are not found in the dictionary, your program will then **suggest alternative words that might be the correctly spelled versions of the misspelled word**. For this assignment, you are expected to check for the following three common typing errors:
    1. **inverted adjacent letters** (ex: ’wrod’ would be an incorrect spelling of ’word’ with inadvertent adjacent letters)
    2. words with **a missing letter at the beginning OR at the end** (ex: ’wor’ would be an incorrect spelling of ’word’ with a missing last letter)
    3. words with **an extra letter at the beginning OR at the end** (ex: ’wordi’ would be an incorrect spelling of ’word’ with an extra letter at the end)

    Although a real spell checker does more than that, implementing this much will suffice for the purposes of this assignment. When checking for missing/extra letters, only check for the 26 English letters. For the incorrectly typed word “spor”, for example, your program’s output to stdout should be as follows:

    Suggestions: spore sport por

    If alternate typings of the misspelled word are not found in the dictionary, your program should print:

    Suggestions:

    that is, it should print an empty statement as above.
  - Finally, **if** the user have called the program with the third command line parameter as *add*, then you will **insert the misspelled word into the dictionary** that you stored inside the hash table. **If** the third command line parameter was *ignore*, then you do not make any change to the internal dictionary that you hold in your hash table.

- Thus, you will be graded on whether your program can;
  - search for an already existing word in the dictionary,
  - suggest alternative words that would be possible correct spellings of the misspelled word, and
  - add a new word to the dictionary.
- **Output** will be printed to standard output and needs to strictly satisfy structure quoted above. Make sure you double check that your output matches that which is being produced by the provided executable file.
- **Test cases and grading:** Your program will be tested across several test cases, and grading for each test case will be as follows:
  1. You get full credit from a test case if your output is both correct and is produced within at most 2 times to produce the output compared to the provided executable
  2. You get half the credit from a test case if your output is correct but it took longer than the above mentioned time.
  3. You get no credit from a test case if your output is wrong.

NOTE: Adding a word to the dictionary means, adding it to the dictionary on memory. You will not modify the dictionary file stored on the hard drive. Thus, once your program quits, the changes made to the dictionary will get lost.

6 Submission

- Even if you develop it elsewhere, your code must run on **tux**. So, make sure that it compiles and runs on tux prior to submitting it.
- Your submission will consist of two separate files (do not compress/zip the files).
  1. A single C file named **main.c**, which includes your code for the assignment.
  2. A single file named **self_evaluation**, without any file extension, which contains answers to the following questions. Your answers can be as long or as short as you would like.
     (a) How many hours did you spend on this assignment?
     (b) What did you struggle with the most?
     (c) What did you learn from this assignment?
- Submit your assignment by following the Gradescope link for the assignment through Blackboard Learn.

7 Grading

- Assignment will be graded out of 100 points.
- You will earn 10 points for submitting a file with your self-evaluation.
- The remaining 90 points will be awarded according to how many test cases your program is able to pass.