Assignment 9

Due on **May 1st, at the end of the day**. Please follow the submission instructions in the "notes for all labs" on Moodle. This lab may overlap some reading assignments that may require short responses, but won't overlap other coding projects.

- Read the "Notes for All Labs" document on Moodle. All assignments must be submitted as specified there. **This means .pdf**.
- This assignment is **pair optional**. You can work alone or with one partner. If you work with a partner, all work must be completed together - don't try to split the assignment in half and each do half.
- All code files and associated material you submit must have both your first and last names in it. (Probably in a header at the top.)
- Explain anything about your code that is not obvious in comments.

Assignment Overview 9.0 (0 Points)

For this project, you will be creating a spellcheck system. At a basic level, your system will detect if a word is misspelled (meaning that it does not appear in a dictionary file you provide). It will then make a list of suggestions, in order from most-likely to least-likely, of what the user may have intended to type instead.

Make sure to read the list of specification details at the end of this lab. I will try to keep it updated if anybody has any clarification questions.

Useful files for this assignment are found in ~barbeda/cs330-share/assign-9-files. You may want to spend some time looking through these to understand what is and how it is formatted. Not all of the files in the directory are necessary for this project; some are provided for if you want to try something fancier than the basic implementation.

Project 9.1: (60 Points)

Your code should work by compiling to a form that takes three files as input: A dictionary file, a file of words to be spellchecked, and a results file, in that order. The dictionary file is not guaranteed to be alphabetized, although if you would like to produce one for efficiency, you may do so. All of the results produced by your spellcheck should go in the results file the user specifies.

Your output file should be in the following format:
For each word that was misspelled, that word should be printed, followed by a space, then a colon, then another space, then the suggestions separated by spaces. For example, one line might look like this (this is just an example, not an actual result):

```
natino : nation ration notion station nations action national antlion national neutrino
```

By default, your script should suggest ten words for each misspelled word. However, this should be a value defined in your script somewhere, so it's easy to change.

The words that are suggested should be the ten words with the smallest *edit distance* to the misspelled word. One type of edit distance is *Levenshtein distance.* The Levenshtein distance between two words is the number of additions, deletions, and substitutions that need to occur in order to transform one word into another.

For example, each of these pairs of words are distance 1 from each other:

speak - spear (substitution)
cloud - clod (deletion)
tuck - truck (addition)

This is not a great measure of what sort of mistakes people are likely to make when typing. It does not account for which types of substitutions are likely to be more common, and transpositions (two adjacent letters switching place) have a cost of 2, even though that is a very common typing error to make. However, it is a reasonable starting place.

In addition to your code, please submit a readme file that indicates how to run your code, any custom dictionaries or test cases you created, and at least one sample output file. Your readme should indicate how the output file was generated.

Written Assignment 9.2: (10 Points)

Test your spellchecker with a variety of inputs (long words, short words, common words, rare words, etc.) and using different dictionaries provided. Take notes about the behavior you notice. Then, come up with at least *four* ways in which you might improve the quality of the suggestions the system makes. For example, you might give common words a boost to their score so they are more likely to appear in the suggestions list.

Then, choose one of your four ideas and give a more detailed design sketch of how you might implement it. (You do not have to actually implement it.)

This should take about one page in total. It can be formatted as lab notes, meaning it does not need an introduction or a conclusion or anything, but should be neat and organized.

Written Assignment 9.3: (10 Points)

Describe, **in a few paragraphs**, your design and implementation process for the project components.

Specification Details

- You should assume that any “word” that contains no letters is spelled correctly. For example, you should not make spelling suggestions for “2047,” even though that is unlikely to be in most dictionaries
- You must deal with punctuation. For example, if the input file contains “I love hrsess! They are nice.” then your code should identify that “hrsess” is a misspelled word, but “nice” is not (assuming that “nice” is in the dictionary.)
- Punctuation characters that are internal to a word can be treated as spaces. For example, “loop-de-loop” can be treated as “loop” “de” “loop”.
- While efficiency is not paramount, your code should not do anything grossly inefficient. For example, your code should *not* re-read the dictionary file for each word you’re spellchecking. (It can go through all the words again, it just shouldn’t re-access the file each time.)
- It is very likely that many words will be tied for the same edit distance from a misspelled word. Your code can choose anything you like as a tiebreaker, as long as it’s methodical in some fashion. (For example, your code might include the ones that come earlier in the dictionary.)