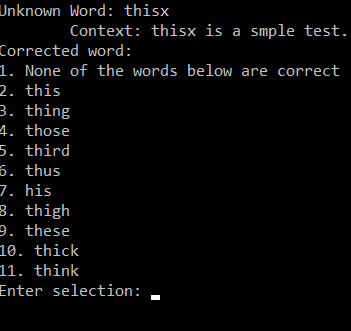
# CS 212 PA #5

In this task, you will write a simple program that suggests corrections to misspelled words. Corrections are suggested based on the number of transformations required to get the misspelled word into a word found in a supplied dictionary. These transformations will be calculated using the [Levenshtein](http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm) edit distance discussed in class. Below, I describe how your program should work.

1. Begin by reading the "words.txt" dictionary into your program.
2. Prompt the user for both an input file to autocorrect and a destination to output the corrected text.
3. For each misspelled word:
   1. Compute the 10 most probable suggestions based on edit distance. Also provide an opportunity for the user to specify their own autocorrected answer:



Your program should gracefully handle improper user input (e.g. values outside the range of 1-11)

* 1. Having obtained the correct spelling, place the correct spelling at the top of the list. In this way, if we encounter the misspelling again, it will be the first suggestion made by our autocorrect.
  2. Because computing Levenshtein distance is fairly expensive, we will save the autocorrect results to a file. Therefore, when we encounter the misspelling again, we don't have to recompute all of the distances.

1. Having corrected all of the misspelled words in the document, write the corrected text to the user-specified output file.

## Header Comment, and Formatting

1. Be sure to modify the file header comment at the top of your program to indicate your name, student ID, completion time, and the names of any individuals that you collaborated with on the assignment.
2. Remember to follow the basic coding style guide. For a list of basic rules, [see my website](http://adamcarter.com/teaching/cpts121/style) or examine my example files from previous assignments and labs.

## Reflection Essay

In addition to the programming tasks listed above, your submission must include an essay that reflects on your experiences with this homework. This essay must be at least 350 words long. Note that the focus of this paper should be on your reflection, ***not*** on structure (e.g. introductory paragraph, conclusion, etc.). The essay is graded on content (i.e. it shows deep though) rather than syntax (e.g. spelling) and structure. Below are some prompts that can be used to get you thinking. Feel free to use these or to make up your own.

* Describe a particular struggle that you overcame when working on this programming assignment.
* Conversely, describe an issue with your assignment that you were unable to resolve.
* Provide advice to a future student on how he or she might succeed on this assignment.
* Describe the most fun aspect of the assignment.
* Describe the most challenging aspect of the assignment.
* Describe the most difficult aspect of the assignment to understand.
* Provide any suggestions for improving the assignment in the future.

## Deliverables

You must upload your assignment through Canvas no later than midnight on Monday, April 22, 2019.

## Grading Criteria

Your assignment will be judged by the following criteria:

### Reflection essay (5pts)

1. Your reflection meets the minimum requirements as specified earlier in this document.

### Basic Functionality (100pts)

1. [20] Your program correctly suggests the most probable corrections to any word
2. [20] Your program places the user's last selection for a given word at the top of the autocorrect list
3. [20] Your program saves autocorrect results to a file for faster lookup
4. [20] Your program doesn't eliminate punctuation marks
5. [20] Your program correctly writes the autocorrected text to an output file

## Advanced Functionality (30pts BONUS)

In the "real world," autocorrect results would likely be stored in some sort of database. In addition to implementing Basic Functionality #4, you save the results to a SQLite database for more realistic operation and (potentially) faster lookup. The design of the SQLite database is completely up to you (feel free to ask me for ideas!). I recommend [SQLiteStudio](https://sqlitestudio.pl/index.rvt) for all platforms for building the database. For incorporating SQLite into your C++ project, you'll need to [download the appropriate files](https://system.data.sqlite.org/index.html/doc/trunk/www/downloads.wiki) for your system. For performing queries in C++, refer to the [SQLite API](https://www.sqlite.org/cintro.html).

When launching your advanced program, prompt the user for basic file store (#4 in basic functionality) or load / store in a database.