Project Title: Autocorrect (Levenshtein distance), autocomplete (prefix tree), and predictive text (pairs of words).

Group Members:
- Grace Mao (SID: 011416470)
- Joseph Lee (SID: 014511094)
- Thinh Manh (SID: 012050571)

Abstract:

Communicating over a digital medium often requires the digitization of the written word into digital text. From articles to proposals to everyday "texting", the presence of digital typing is ubiquitous and thus is the need to type accurately, especially given the relative size of thumbs to the size of letters on a mobile phone. Accuracy can be ensured through the use of a spell checker, or autocorrect feature. Digital typing can be augmented further with the ability to predict partial, current words (autocomplete) as well as future words (predictive text). We implement autocorrect (Levenshtein distance), autocomplete (prefix tree), and predictive text (pairs of words) using Python.

To implement autocorrect, we take in an input of a (misspelled) word and calculate the Levenshtein distance to a dictionary of English words stored in a hash structure. The shortest distance will represent the shortest "edit distance" from our input string to a valid English word; and "edit" being a single-character edit (insertions, deletions or substitutions).

Autocomplete could be implemented similarly to autocorrect, but partial words might we mapped to unwanted outputs via Levenshtein distance. Assuming the input words have accurate initial characters, we implement autocomplete using a prefix-trie generated from our dictionary of English words that dynamically suggests words upon user input.

Lastly, we implement predictive text to suggest the next word given an valid input word. This can be done by referencing pairs of English words found by analyzing a large quantity of text and generating a frequency table of pairs. Predictive text can be personalized to a user by caching a user's text input and weighting those pairs more heavily than from our standard dictionary.

Schedule:

Week	**Date**	**Checkpoint**	**Deliverable**
0	3/5		
1	3/12	Choosing algorithm and method for the project	Submit Abstract
2	3/19		
3	3/26	Autocorrect	
4	4/2		
5	4/9	Autocomplete	
6	4/16		
7	4/23	Predictive Text	
8	4/30	Optimizations	
	5/3	Test	Submission Due
	5/7; 5/9		Presentation, Demo