COMP 3270 SPRING 2020
Programming Project: Autocomplete

Name: _____ Date Submitted: _____

1. **Pseudocode**: Understand the strategy provided for *TrieAutoComplete*. State the algorithm for the functions <u>precisely using numbered steps that follow the pseudocode conventions</u> that we use. Provide an approximate efficiency analysis by filling the table given below, for your algorithm.

Add
- Pseudocode:

- Complexity analysis:

Step #	Complexity stated as O(_)

Complexity of the algorithm = O(__)

topMatch
- Pseudocode:

- Complexity analysis:

Step #	Complexity stated as O(_)

Complexity of the algorithm = O(__)

topMatches
- Pseudocode:

- Complexity analysis:

Step #	Complexity stated as O(_)

Complexity of the algorithm = O(__)

2.**Testing**: Complete your test cases to test the *TrieAutoComplete* functions based upon the criteria mentioned below.

Test of correctness:

Assuming the trie already contains the terms {"ape, 6", "app, 4", "ban, 2", "bat, 3", "bee, 5", "car, 7", "cat, 1"}, you would expect results based on the following table:

Query	k	Result
""	8	{"car", "ape", "bee", "app", "bat", "ban", "cat"}
""	1	{"car"}
""	2	{"car", "ape"}
""	3	{"car", "ape", "bee"}
"a"	1	{"ape"}
"ap"	1	{"ape"}
"b"	2	{"bee", "bat"}
"ba"	2	{"bee", "bat"}
"d"	100	{}

3.**Analysis**: Answer the following questions. Use data wherever possible to justify your answers, and keep explanations brief but accurate:

i. What is the order of growth (big-Oh) of the number of compares (in the worst case) that each of the operations in the *Autocompletor* data type make?

ii. How does the runtime of *topMatches()* vary with k, assuming a fixed prefix and set of terms? Provide answers for *BruteAutocomplete* and *TrieAutocomplete*. Justify your answer, with both data and algorithmic analysis.

iii. How does increasing the size of the source and increasing the size of the prefix argument affect the runtime of *topMatch* and *topMatches*? (Tip: Benchmark each implementation using fourletterwords.txt, which has all four-letter combinations from aaaa to zzzz, and fourletterwordshalf.txt, which has all four-letter word combinations from aaaa to mzzz. These datasets provide a very clean distribution of words and an exact 1-to-2 ratio of words in source files.)

4. Graphical Analysis: Provide a graphical analysis by comparing the following:

       i.   The big-Oh for *TrieAutoComplete* after analyzing the pseudocode and big-Oh for *TrieAutoComplete* after the implementation.

      ii.   Compare the *TrieAutoComplete* with *BruteAutoComplete* and *BinarySearchAutoComplete*.