Assignment #0 readme

Group 53:

Andrew Marshall: alm266

Brandon Diaz-Abreu: brd48

Our implementation of pointersorter is user-friendly and space-time efficient. Our code was written to be robust and handle various user inputs gracefully. The core of our program is the trie data structure. Because there were so many unknowns in terms of the size and complexity of the input string, we needed a data structure that would be extensible. The other requirement was to print out the input in sorted order and so we needed an efficient way to build a sorted list. The trie was the right data structure to use because it could handle all of these requirements for us in a very manageable way.

The program only adds words to the trie as it reads them in and keeps a counter for remembering duplicates. We chose to convert all input words to lowercase because the loss in semantic value is minimal (PIZZA, piZZa and pizza are all the same word!) and because it would require increasing the size of the trie nodes to accommodate capital letters. The worst case is that the input string is one valid word, meaning that a node in the trie is created for each character. Therefore the space efficiency is O(m) where m is the length of the input string. But on average, an input string will be composed of multiple words and we won't use O(m) space.

Printing out the sorted words is trivial with a trie. All the nodes are created and stored in alphabetical order. This means that when we need to print the words, all that we have to do is an in-order depth-first traversal of the trie. This has a worst case where none of the words in the input string have a common root. This implies O(k) time, where k is the combined length of all the distinct words in the input. But again, we can expect that on average, some words will share the same root and therefore produce a more compact trie.

Our test cases were designed to test every possible format of the input (i.e. multiple separators in a row, empty strings, etc.). As for the cases of erroneous input (too many or too few arguments), we chose to have the program exit with an informative error message. It isn't clear what a user's true intention might have been if they entered anything more or less than what was specified in the assignment instructions. We hope our code serves you well.