CPSC 319
Assignment 2

Linked Lists and Sorting

The goal of this assignment is to write a Java program that arranges a list of words into separate lists of anagrams. The input is a text file that contains a list of words, each word on its own line. The number of words in the input is arbitrary and could be very large.

The program should print to an output file the lists of anagrams in the following way:

- All the words that are anagrams of each other are displayed together on a single line; any word without any corresponding anagrams is displayed alone on a line.
- The words on each line should be in alphabetic order.
- Lines of words are sorted into ascending alphabetic order according to the first word of a line.
- Words in a line should be separated by a single space.

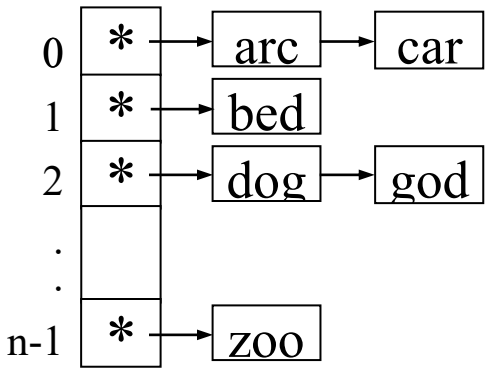
For example, this input:

```
car
dog
bed
stop
god
pots
arc
tops
```

should yield the output:

```
arc car
bed
dog god
pots stop tops
```

You must use linked lists to deal with the arbitrary number of anagrams in a line, and use an array of references to keep track of all the linked lists. For example:



An acceptable alternative to using an array is to use a vector, provided you program your own vector class (i.e. you cannot use a class such as Vector or ArrayList from the Java libraries for this).

Adapt the insertion sort given in class so that it works on the items in the linked lists, and adapt the quick sort so that it sorts the array of references. Be sure to cite the sources of any code you use or adapt. You must write your own implementation of linked lists and sorting algorithms rather than using calls to a Java library.

Your program should read and store all the words in the file to your data structure first, and then sort as a second step, applying the insertion sort to each row of words, and then the quick sort to the array, using the first word in the list as the sort key. An acceptable alternative to the insertion sort is to insert each word in its row in ascending order as you read it from file. Your program must also print out to the screen the time in seconds it takes to process an input file.

One way to determine if two words are anagrams is to sort the letters in both words. If the sorted words are the same, then the original two words are anagrams of each other. For example, "dog" and "god" are both sorted into "dgo", so they are anagrams.

Write a program in Java to implement the above requirements. Your program will be invoked from the command line as follows:

```
java Assign2 inputfile outputfile
```

where *Assign2* is the name of the file containing executable bytecode for your program, *inputfile* is the name of the input file, and *outputfile* is the name of the output file.

Complexity Analysis

1. What is the worst-case complexity of your algorithm when checking if two words are anagrams of each other? Express this using big-O notation, and use the variable k to represent the number of letters in each word. Support this with a theoretical analysis of your code.
2. Let N be the number of words in the input word list, and L be the maximum length of any word. What is the big-O running time of your program? Justify your answer using both a theoretical analysis and experimental data (i.e. timing data).

Submit electronically using the D2L dropbox:

1. A *readme* file, indicating how to compile and run your program.
2. Your source code files. Your TA will run your program to verify that it works correctly. Make sure your Java program compiles and runs on the Computer Science Linux servers.
3. Your complexity analysis in Word or PDF format.

Collaboration

The assignment must be done individually, so everything that you hand in must be your original work, except for the code adapted from the text, lectures, or other sources to implement your sorting algorithms and data structures. When someone else's code is used like this, you must acknowledge the source explicitly in your program. Copying another student's work, in whole, or in part, will be deemed academic misconduct. Contact your TA if you have problems getting your code to execute properly.

CPSC 319
Assignment 2 Grading

Student:_____

Program			
	Command-line arguments	2	_____
	Error checking of arguments	2	_____
	Reading of input text file	2	_____
	Linked list data structure	4	_____
	Array of references data structure	2	_____
	Insertion sort implementation	4	_____
	Quick sort implementation	4	_____
	Algorithm to detect anagrams	4	_____
	Correct output to file	2	_____
	Timing code	2	_____
	Code structure (OO style, documentation, formatting, etc.)	4	_____
Complexity Analysis			
	Question 1	3	_____
	Question 2	5	_____
Total		**40**	**_____** **_____%**