CPSC 319 Assignment 2: Searching and Sorting Anagrams

Due Date: Friday March 6th, 2020 11:59pm
Weight: (30 * 0.25)% [Assignment 2 of 4]

Collaboration
IMPORTANT

Discussing the assignment requirements with others is a reasonable thing to do, and an excellent way to learn. However, the work you hand-in must ultimately be your work. This is essential for you to benefit from the learning experience, and for the instructors and TAs to grade you fairly. Handing in work that is not your original work, but is represented as such, is plagiarism and academic misconduct. Penalties for academic misconduct are outlined in the university calendar.

Here are some tips to avoid plagiarism in your programming assignments.

1. **Collaborative coding is strictly prohibited.** Your assignment submission must be strictly your code. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing code itself, or modelling code after another student's algorithm. **You can not (even with citation) use another student's code.**
2. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code that it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's console, then this code is not yours.
3. Everything that you hand in must be your original work, except for the code copied from the textbook, lecture material (i.e., slides, notes), web, or that supplied by your TA. When someone else's code is used like this, you must acknowledge the source explicitly, citing the sources in a scientific way (i.e., including author(s), title, page numbers, URLs, etc.)
4. Cite all sources of code that you hand-in that are not your original work. You can put the citation into comments in your program. For example, if you find and use code found on a web site, include a comment that says, for example:
   ```
   # the following code is from
   https://www.quackit.com/python/tutorial/python_hello_world.cfm.
   ```
   Use the complete URL so that the marker can check the source.
5. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if you submit code that is not primarily developed by yourself.
6. We will be looking for plagiarism in all code submissions, possibly using automated software designed for the task. For example, see Measures of Software Similarity (MOSS - https://theory.stanford.edu/~aiken/moss/).
7. Copying another student's work constitutes academic misconduct, a very serious offense that will be dealt with rigorously in all cases. Please read the sections of the University Calendar under the heading "Student Misconduct". If you are in doubt whether a certain form of aid is allowed, ask your instructor!

Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help than it is to plagiarize.

Late Penalty:

LATE ASSIGNMENTS WILL NOT BE ACCEPTED

Goal

The goal of this assignment is to write a Java program that arranges a list of words into separate lists of anagrams. Your program should be named "**CPSC319W20A2**" and will be called from the command line as follows

```
$ java CPSC319W20A2 < file
```

where ***file*** is a file containing a list of words to be sorted into anagrams and is read by the program through standard input. The number of words in the input is arbitrary. The program should print to standard output the lists of anagrams in the following way:

1. all the words that are anagrams of each other are printed on one line of output,
2. the words on each line should be in alphabetic order,
3. the groups should be ordered alphabetically by the string that results by sorting the characters in each word,
4. exactly one space between words on the same line, and
5. no other spaces.

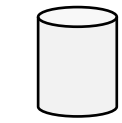
For example, this **input text file**:	→	Should yield the **output text file**:
car mega bed stop game pots arc tops		arc car bed game mega pots stop tops

Note that the input can be large so attention to the efficiency of the algorithms is essential. Also, we will grade the program using utilities that compares text files. That is, we will have a large input file with the known correct output. If your output file does not match the correct output, you will lose some marks. Some test files will be available on D2L.

The Algorithms

You are required to use **1-D arrays** and **singly linked lists** in your program to deal with the arbitrary number of words in the input as follows:

Step #1. The data from the **input** text file should be structured in a 1-D array of words (example):



Input text file

0	car
1	mega
2	bed
3	stop
4	game
5	pots
6	arc
7	tops

LIST A

1-D array of words

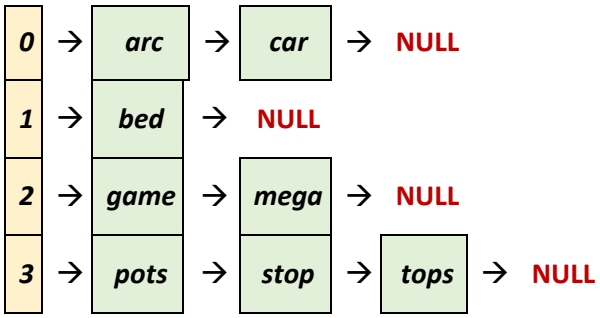
Step #2. LIST A should then be **sorted** (example):

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST A

Sorted 1-D array of words

Step #3. A new **LIST B** should then be created directly from **LIST A** (i.e., the sorted 1-D array of words, step #2) as follows (example):



LIST B

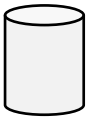
1-D array of **singly linked lists**

LIST B is a **1-D array** storing, at each array entry, a **singly linked list** of all the anagrams found in the sorted **LIST A** (step #2) – i.e., (arc, car), (game, mega), (pots, stop, tops) – as well as any remaining word(s) with no anagrams found in the sorted **LIST A** (step #2) – i.e., (bed).

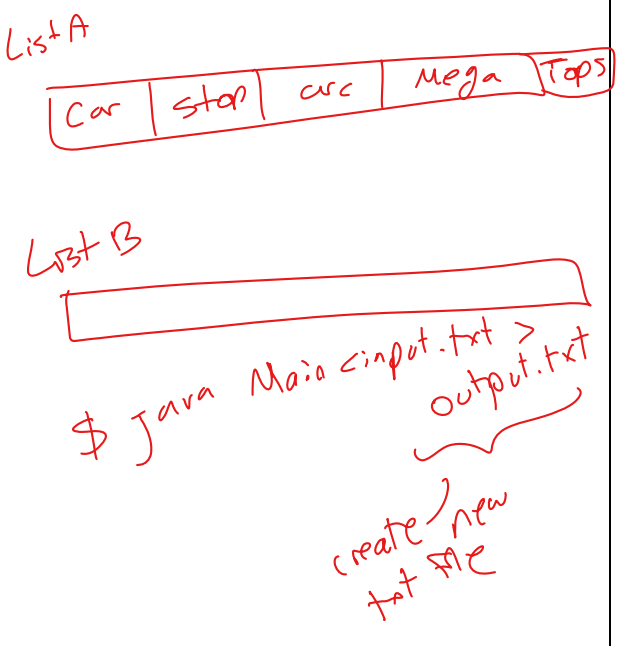
Note: all the words in each of the singly linked lists are stored in alphabetical order and subsequently stored at each array entry also in alphabetical order – e.g., note the first word at LIST B [0] → (**arc**, car), LIST B [1] → (**bed**), LIST B [2] → (**game**, mega), and LIST B [3] → (**pots**, stop, tops). **This is achieved by** structuring **LIST B** (i.e., both 1-D array and singly linked lists) directly from **LIST A** which is sorted already (i.e., step #2).

Step #4. Traverse **LIST B** to generate the required **output text file** format (refer to previous page).

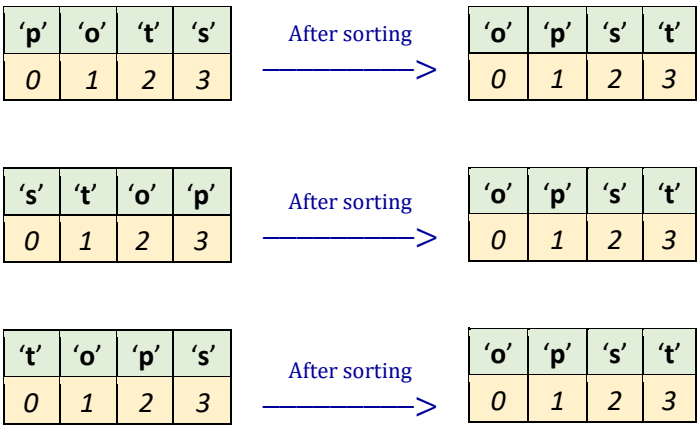
arc car
bed
game mega
pots stop tops



Output text file



Finding Anagrams: A good way to determine if two words are anagrams is to sort the letters in both words. If the two sorted words are the same, then the original two words are anagrams. For example, array entries #5, #6, and #7 (i.e., "pots", "stop", and "tops") from the sorted **LIST A** of words (i.e., step #2) are anagrams:



IMPORTANT: You should use your own implementation of **singly linked lists** and **sorting algorithms**, NOT Java Collection Classes or other such libraries. You may use our textbook, lecture and tutorial materials or other sources of information as guidance but be sure to cite them.

Instructions

1. Write the program described above.
2. Hand in the program with answers to the questions below.

Hand-In (digitally to D2L dropbox)

1. A written report (PDF format) with your name and ID number in the grading summary, including the answers to the questions. Keep your answers short and to the point. The maximum length for a report is 2 pages. Any extra pages should be placed as an Appendix. The Appendix should not be required reading to understand the provided answer in the first two pages.
2. Your source code file that generates the output in the form requested (in specified format [Java8]). The README.md file should include description of which class begins execution of your submitted code. For example, "CPSC319W20A2.java" is my main class and should be executed, after compiling with "Makefile", with no arguments.
3. Include all of the above items in a zipped folder (standard ZIP) titled CPSC319W20A2-LASTNAME-ID.zip
   before submission to the **D2L dropbox** for assignment 2.

Grading

- Assignment grades will be base equally on the two submitted components as described on grading summary

- Source code that does not compile with the supplied 'Makefile' from assignment 1 (zipped folder “CPSC319W20A1-Makefile”) or produces run-time errors will receive a grade of 0%.

- Code that produces incorrect output (Ex. wrong format of output) will be penalized appropriately.

- Your written answers should include well structured sentences, proper grammar, and arguments of reasoning that can be understood. Giving singular formula answers, single word statements, or leaving details implicit will be marked accordingly. It is not the job of the TA to assume you meant something you didn’t say.

The conversion between a percentage grade and letter grade that will be posted online is as follows.

Letter	A+	A	A-	B+	B	B-	C+	C	C-	D+	D	F
Min. Perc.	95%	90%	85%	80%	75%	70%	65%	60%	55%	50%	45%	Below 45%

Assignment grading and questions continue on the next pages.

GRADING SUMMARY

Name: _____

ID: _____

PROGRAM:		/20
REPORT:		/20
TOTAL		**/40**

PROGRAM

Program builds correctly	Yes	No
Runs: completes with zero run-time errors	Yes	No

IMPORTANT: Source code that does not compile or produces run-time errors will receive 0%

Criteria	TOTAL
Functionality: produces correct output	
Readability: code is clear and easy to read	
Modularity: code is easy to extend, or use	
Generality: easy to extend to more sophisticated applications	
Documented: code classes/functions/inline commented	
TOTAL:	**/20**

REPORT

Criteria	TOTAL
Efficiency: can the program handle a large input file	/4
Correctness: all guidelines followed input/linked list usage/etc.	/4
Question #1:	/4
Question #2:	/4
Question #3:	/4
TOTAL:	**/20**

Questions

1. Describe the sorting method(s) used in your program – i.e., sorting the array storing all words from the input file (LIST A, step #2) and sorting the letters of two words from LIST A (step #2) to determine whether they are anagrams. Justify your selection of algorithm(s).
2. Let N be the number of words in the input word list and L be the maximum length of any word. Give an estimate of the big-O running time of your program. Justify your answer.
3. What is the big-O running time when the input word list contains only two words?