

CPSC 319 – Winter 2024 Assignment 2 Sorting & Searching for Anagrams	Released:	January 29 (Monday)
	Due:	February 15 (Thursday) @ 11:59 PM
	# days to complete:	17 days

GOAL

The objective of this programming assignment is to develop a Java program that organizes a given list of words into separate groups of anagrams, where an anagram refers to a word or phrase formed by rearranging the letters of another word or phrase. The program is required to read input from each of the **four given input files**, each containing a list of words to be sorted into anagrams. The number of words in the input file is variable.


Upon execution, **for each of the 4 input files**, your program should produce the corresponding output file, organizing the words into groups of anagrams according to the following criteria:


1. All words that are anagrams of each other should be printed on a single line of output.
2. Words within each line should be arranged in alphabetical order.
3. The groups of anagrams should be ordered alphabetically based on the string obtained by sorting the characters in each word.
4. There should be exactly one space between words on the same line.
5. There should be no additional spaces.


For example, this **input text file**: → Should yield the **output text file**:


car	arc car
mega	bed
bed	game mega
stop	pots stop tops
game	
pots	
arc	
tops	

Note that the input can be large so attention to the efficiency of the algorithms is essential. Your program should be able to process each of the **4 input files** provided in folder 'PA-2—INPUT-DATA-to-use', with the following input text files to be used as the input for your programs.

 INPUT-DATA-1--with-8_words

 INPUT-DATA-2--with-13_words

 INPUT-DATA-3--with-19_words

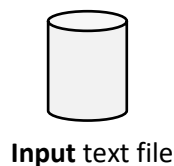
 INPUT-DATA-4--with-267_words

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.

ALGORITHMS & DATA STRUCTURES

You are required to use **1-D arrays** (Sec. 3.1 from textbook) and **singly linked lists** (Sec. 3.2 from textbook) 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):



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

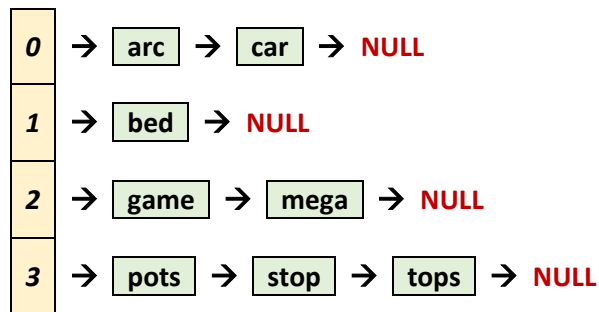
LIST A (array A)
1-D array of words

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 (array A)
Sorted 1-D array of words

Step #3. A new **LIST B** (1-D array of singly linked lists) 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** (1-D array of singly linked lists) to generate the required **output text file** format (refer to previous page).

arc car
bed
game mega
pots stop tops



Output text file

Refer to folder “PA-2--Examples-of-expected-outputs” for examples of how your final output text files should look like.

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:

1-D array					1-D array			
0	1	2	3	<i>after sorting</i> →	'o'	'p'	's'	't'
'p'	'o'	't'	's'		0	1	2	3
's'	't'	'o'	'p'	<i>after sorting</i> →	'o'	'p'	's'	't'
0	1	2	3		0	1	2	3
't'	'o'	'p'	's'	<i>after sorting</i> →	'o'	'p'	's'	't'
0	1	2	3		0	1	2	3

IMPORTANT: Refer to our textbook, **Chapter 3 Fundamental Data Structures**, Sections **3.1 Using Arrays** and **3.2 Singly Linked Lists** and the tutorial material to assist you with the implementations of arrays and singly linked lists. You should use your own implementation of **sorting algorithms**. 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. Use the skeleton code provided writing your code where you find 'TODO'
3. Hand in the program with answers to the questions below.

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.

HAND-IN

1. Cover sheet with your name and ID number only.
2. Grade summary sheet provided in postscript form on the course web page.
3. A written report with answers to the questions to the assignment drop boxes for your tutorial section in this course. 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.
4. Your code.
5. Include all the above items in a zipped folder titled (PA-2--your LAST NAME-ID).zip and submit this folder electronically to D2L dropbox with subject PA-2--your LAST NAME-ID

LATE ASSIGNMENTS WILL NOT BE ACCEPTED

MARKING

- Assignment grades will be based equally on the two submitted components as described on the cover/grading sheet (first page of this document)
- Source code that does not compile or produces run-time errors will receive a grade of 0%.
- Code resulting in incorrect output will incur point deductions.

COLLABORATION

- The assignment must be done **individually** so you must write up the solutions *on your own* in *your own words*.
- 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.)
- 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!
- Contact your TA if you have problems getting your code to work.
- Note that your code may be checked thoroughly for plagiarism by computer.

END OF PA-2