CPSC 319 – Summer 2020	**Released:**	July 1, 2020 @ 11:59 PM
Assignment 1	**Due:**	July 15, 2020 @ 11:59 PM
Searching & Sorting 10% of total grade	# days to complete:	14 days

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 "**asgmt1**" and will be called from the command line. The only input to your program is a file containing a list of words to be sorted into anagrams and is read by your program through standard input. The number of words in the input is arbitrary. Your 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 no other spaces.

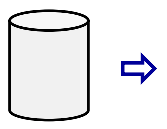
For example, this **input text file**:	→	Should yield this **output text fil**e:
car		arc car
mega		bed
bed		game mega
stop		pots stop tops
game		
pots		
arc		
tops		

Note that the input can be large, so attention to the efficiency of the algorithms is essential. 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. You must test your program with 4 test files (available on D2L) as shown in the right:

Input File	**Expected Output**
example_1--8_words	example_1_out
example_1--13_words	example_2_out
example_1--19_words	example_3_out
example_1--267_words	example_4_out

ALGORITHMS & DATA STRUCTURES

Step #1. The data from the **input** text file should be structured in a 1-D array of words (example: input text file "example_1--8_words"):



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**:

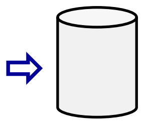
0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST A

Sorted
1-D array of words

Step #3. Traverse **LIST A** to generate the required **output text file** format of found anagrams **exactly** as shown in the example below for all the 4 input files.

arc car
bed
game mega
pots stop tops



Output
text file for "example_1_out"

Finding Anagrams: A good way to determine if <u>two</u> words are anagrams is to sort the letters in <u>both</u> 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 <u>sorted</u> **LIST A** of words (i.e., step #2) are anagrams:

‘p’	‘o’	‘t’	‘s’		‘o’	‘p’	‘s’	‘t’
0	*1*	*2*	*3*	$\xrightarrow{\textit{after sorting}}$	*0*	*1*	*2*	*3*

‘s’	‘t’	‘o’	‘p’		‘o’	‘p’	‘s’	‘t’
0	*1*	*2*	*3*	$\xrightarrow{\textit{after sorting}}$	*0*	*1*	*2*	*3*

‘t’	‘o’	‘p’	‘s’		‘o’	‘p’	‘s’	‘t’
0	*1*	*2*	*3*	$\xrightarrow{\textit{after sorting}}$	*0*	*1*	*2*	*3*

IMPORTANT: You should use your **own implementation** of the **sorting algorithm(s)** covered in class (e.g., Selection, Bubble, Insertion Sort) You may use our textbook, lecture and tutorial materials or other sources of information as guidance but be sure to <u>cite</u> them. For the required **data structure** (1-D arrays) you can either use Java classes or implement your own. **No hard coding** is allowed. You need to have your code explicitly reading the input text files (i.e., Step #1 above), sorting the arrays, and explicitly save the solution (i.e., Step #3 above). Our TAs will assist you with every aspect of the assignment.

HAND-IN

- Submit your source code electronically to **D2L dropbox**.
- Please name your source code file as follows: “asgmt-1-your-last name-UCID”
- **Late assignments will not be accepted.**

MARKING

Source code that does not compile or produces run-time errors will receive a grade of 0%

Input File	**Expected Output**	**# Points**	
		for correct output (Step #3)	**for partially correct output (Step #2)**
example_1--8_words	example_1_out	2	1
example_1--13_words	example_2_out	2	1
example_1--19_words	example_3_out	2	1
example_1--267_words	example_4_out	2	1
	Total	**8**	**4**

COLLABORATION

- The assignment must be done **individually** so you must write up the solutions *on your own.*
- 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 grave offence 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 particular form of aid is allowed, ask your instructor!
- Contact your TA if you have problems getting your code to work.
- **Note that your code will be checked thoroughly for plagiarism by computer.**

END OF THE ASSIGNMENT