CCPROG2 Graded Exercise 03 (GE03) – 1D Array of STRINGS

サルバドル・フロランテ

GENERAL INSTRUCTIONS:

HONESTY POLICY and HONOR Applies. Cheating is an unacceptable behavior and will result to 0.0 in this course.

This is a GRADED hands-on programming exercise. You may choose to solve it individually or form a group, learn and collaborate on the solution. There should be a maximum of 3 students per group. Each student has to sign-up in **GE03GROUP** Canvas Groups regardless of whether he/she chose to do the exercise individually or with a group.

It is suggested that you solve the problem first on your own. Once you have produced a solution, discuss it with your groupmates, i.e., do compare your solutions, techniques, and exchange ideas and good programming practices that you learned along the way. You should limit the solution within your own group.

Do not obtain a solution from other individuals/groups, or share your solutions to individuals/groups. Both actions are considered cheating!

SKELETON FILE: you are provided C source code skeleton files. Your primary task is to complete those files by supplying the missing codes. Read, understand and comply with the problem specifications, i.e., RESTRICTIONS, TASKs and the detailed TO DOs specified as comments.

TESTING and SCORING SYSTEM: make sure to compile and test your solution exhaustively by specifying and creating your own test cases. Ensure that there are no errors and no warnings. For each problem:

- a correct solution will get 10 points
- a function with syntax error(s) will automatically get a score of 0 -- this means that you did not check with a compiler your solution
- a solution with logical error(s) will get a score of a minimum of 0 to a maximum of 4 points.
- a solution with warning(s) will get a deduction of 1 point PER unique warning.

DELIVERABLE: submit **GE03-NUMBER.zip** which when uncompressed will produce a folder containing two files:

1. **GE03-NUMBER.c** source file
2. **RESULT-NUMBER.txt** using INPUT.TXT as input data.

Submit your ZIP file via Canvas before the indicated deadline.

QUESTIONS? Post your question in either Q&A Discussion thread on Arrays or Strings.
****
Demonstrate your knowledge on 1D Array of strings. Implement FOUR functions as described in the comments in **GE03-NUMBER.c**.

The **main.c** program when executed should do the following:

1. Read strings (words) via scanf() via input redirection.
    a. You should examine the accompanying **INPUT.TXT** which contains the words used as example input data.
    b. Assume that all words (input data) are in CAPITAL LETTERS.
2. Insert the words into a 1D array of strings such that they are maintained alphabetically.
    a. Note that you should not create/call a sorting function (i.e., selection algorithm that we discussed before).
    b. Think of your own algorithm for the insertion.
3. Compute the frequency counts for each letter based on the words stored in the 1D array of strings.
    a. Let's have an example: for the word "HELLO", the corresponding frequency counts for each letter in the word are: 'H' is 1, 'E' is 1, 'L' is 2 and 'O' is 1.
4. Display the frequency counts for each letter.
5. Delete some "victim" words from the 1D array of strings.
    a. The alphabetical ordering of words is maintained after the deletion.
    b. Think of your own algorithm for the deletion.
6. Display the contents of the 1D array of words.

Let us have some examples of insertions. Assume an initially empty 1D array of strings represented visually below.

0	1	2	4	5	6	7	8	9	10

Insert the word "THE". It will be inserted as the 1st element (since the array is initially empty)

"THE"									
0	1	2	4	5	6	7	8	9	10

Next, insert the word "QUICK". Notice that is inserted before "THE" to maintain an alphabetically sorted list. "THE" had to be moved towards the right of the array to make room for "QUICK".

"QUICK"	"THE"								
0	1	2	4	5	6	7	8	9	10

Next, insert the word "BROWN". Notice that is inserted before "QUICK" to maintain an alphabetically sorted list.

"BROWN"	"QUICK"	"THE"							
0	1	2	4	5	6	7	8	9	10

Next, insert the word "FOX". Notice that is inserted before in between "BROWN" and "QUICK".

"BROWN"	**"FOX"**	"QUICK"	"THE"						
0	1	2	4	5	6	7	8	9	10

Next, insert the word "ZEBRA". Notice that is inserted after "THE".

"BROWN"	"FOX"	"QUICK"	"THE"	**"ZEBRA"**					
0	1	2	4	5	6	7	8	9	10

Next, we try to insert (again) the word "THE". Notice that "THE" is already in the list. In this case, do NOT insert "THE" again. Thus, the 1D array will still be the same.

The 1D array contains UNIQUE words only, i.e., there should not be any repetition.

Next, let us have some examples of deletions. Assume that the current contents of the 1D array are:

"BROWN"	"FOX"	"QUICK"	"THE"	"ZEBRA"					
0	1	2	4	5	6	7	8	9	10

Delete "FOX". Notice that the words after "FOX" are moved towards the left of the array. We refer to "FOX" as the 'victim' (of the delete operation). Note also that the element at index 5 still has a value but it's considered as a garbage value (shown visually as ?).

"BROWN"	"QUICK"	"THE"	"ZEBRA"	?					
0	1	2	4	5	6	7	8	9	10

Delete "BROWN". The resulting array will be:

"QUICK"	"THE"	"ZEBRA"	?	?					
0	1	2	4	5	6	7	8	9	10

That's all folks!

終