PROG20799: Data Structures and Algorithms in C

Evaluation: **10 points, 7.5%** of your final grade.
Due date: See SLATE.
Late submission: 10% per day penalty, up to 3 days.

In this exercise you are given file **quotes.txt** that has Computer Science quotes. Your goal is to sort the quotes by length using **Selection Sort** algorithm and save them in the file **output.txt**.
You must use the program template **main.txt** file and **quotes.txt** file given to you.
Please check the demo version: https://youtu.be/GQpa3l5Tby4

Main Requirements:

1. You absolutely must use the program template **main.txt** file. You **cannot** modify prototypes or code inside **main()** function! You must add required header files and create functions using the function prototypes set in the template.
   You'll get a **-3** points penalty if this requirement not met.

2. Please check **output.txt** file – that's the file your program must generate (**OUT_FILE** macros) when **MAX_QUOTES** is set to **300.** Please check demo posted on YouTube what the program does when **MAX_QUOTES** is set to 1 or 5.

3. Your program must correctly allocate memory on the HEAP for each quote (i.e. for each element of the array of pointers **quotes**).
   You'll get a **-3** points penalty if this requirement not met.

4. You must check and deal with potential problems with memory allocations or file IO. For instance, malloc/calloc can return 0, file(s) cannot be opened or created, etc.

5. When reading quotes from text file using **fgets(tmp, ..**), you need to know how many characters are in the longest quote. You'd need to pre-scan the input file **quotes.txt**, find the length of the longest string (see Hints), create a temporary string **tmp** on the HEAP that can read the longest string and use this string in the **fgets()** function when you read the file again.

6. You must use **Selection Sort** algorithm to sort by length of the quotes. **No other sorting algorithm will be accepted!** Quotes with the same length must be sorted lexicographically. See Hints.

7. You must allocate memory on the HEAP for quotes and temporary string(s), your solution should have optimal time and space complexity, and support modular design.
   There should be no compilation warnings or commented out "debugging" code.

Hints:

1. The find the length of the longest string in the file you can use function fgetc() and count number of characters up to (and including) new line character.

2. Each pointer of the array of pointers **quotes** should point to a memory allocated on the heap. See in-class exercises for many useful hints.

3. The length of the string/quote determines how much memory must be allocated for it. Don't forget about null-terminating character! I.e. for a string "hi" you'd need to allocate 2+1 bytes.

4. For **selection sort** algorithm it's enough to "swap" pointers! I.e. if pointer **ptr[0]** points to string "hello" and pointer **ptr[1]** points to string "hi", then you simply "swap" them so pointer **ptr[0]** points to shorter string "hi".

5. To compare strings **lexicographically** you might want to check function **strcoll().**

6. Check the **Selection Sort** algorithm and code: https://www.geeksforgeeks.org/selection-sort/
   In the **selection_sort()** function you'd need to use the length of the string/quote for sorting.

7. You'd need to apply **Selection Sort** every time you add new quote to array of quotes.
   Note that array of quotes has **MAX_QUOTES+1** elements. The algorithm works as follows:

   - Assume you want to output only **two shortest strings**.
     Assume you have the following strings in the **quotes.txt** file: CCCCC, BBDA, BADA, ZZ.

   - First, you read string "CCCCC", allocate memory block for it on the HEAP, and copy the string to that memory block. **quotes[0]** will point to that memory block on the HEAP (we'll use terminology **quotes[0]** points to "CCCCC".)

   - Then you read string "BBDA", **quotes[1]** will point to "BBDA" on the HEAP.
     You apply **selection_sort** to **quotes[]** array.
     quotes[0] now points to "BBDA" and **quotes[1]** points to "CCCCC" on the HEAP.

   - Then you read "BADA" and **quotes[2]** now points to "BADA" on the HEAP.
     You apply **selection_sort** to **quotes[]** array again.
     quotes[0] now points to "BATA", **quotes[1]** points to "BBDA" and **quotes[2]** to "CCCCC".

   - Then you read "ZZ", old memory block **quotes[2]** is freed, **quotes[2]** now points to "ZZ".
     You apply **selection_sort** to **quotes[]** array again.
     quotes[0] now points to "ZZ", **quotes[1]** points to "BATA" and **quotes[2]** to "BBDA".

   - Since you only need **two shortest quotes**, you output **quotes[0]** and **quotes[1]** and ignore **quotes[2]**.

8. To find how many characters in the longest string you can use **fgetc()** function and count characters until new line character '\n'.

Submission:

This is an individual assignment. Even partially copied code will be subject to regulations against academic integrity. Do NOT discuss or share your solution with anybody. Posting this assignment or solution on the Internet is a violation of the Student Code of Conduct.

- Please make sure your code is POSIX-compliant (works in NetBeans/Cygwin)!
- Your submission must be one **text file only**: **assignment2.txt**
- Please copy **main.c** as text file **assignment2.txt** with extension **.txt**
- You must upload **assignment2.txt** file to the Assignment Dropbox by due date.
- Verify that **assignment2.txt** file you submitted can be opened by Notepad in Windows.
- Your submission must be unique or have references.
- Please **self-evaluate your code** in the comments section of the Dropbox.
- **Late submissions are penalized 10% / day (1 point / day), up to -5 points.**
  Submissions won't be accepted after 3 days.

Grading Scheme:

- See **Main requirements** and also **Course_Introduction.pdf**.
  Deductions will be applied if partial functionality is provided.
- You'll get **zero grade** if your code doesn't compile.
- Compilation warnings are considered to be major mistakes.
- Comments are not required but recommended. However, "debugging code" or commented out "old code" is a minor mistake (unless it's clearly stated in the comments that you want me to take a look at such code).
- You'll get **-3 points** if you submit incorrect file (.zip file, or file with incorrect name/extension).
- Please check submission requirements for any additional deductions.