School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Winter 2021

Mini Assignment C: File IO and Dynamic Memory

Due Date April 13th, 18:00 EST

This is an individual assignment. You need to solve these questions on your own. If you have questions, post them on Piazza, but do not post major parts of the assignment code. Though small parts of code are acceptable, we do not want you sharing your solutions (or large parts of them) on Piazza. If your question cannot be answered without sharing significant amounts of code, please make a private question on Piazza or utilize TA/Instructors office hours. Late penalty is -15% per day. Even if you are late only by a few minutes it will be rounded up to a day. Maximum of 2 late days are allowed.

You MUST use `mimi.cs.mcgill.ca` to create the solution to this assignment. You must not use your Mac command-line, Windows command-line, nor a Linux distro installed locally on your laptop. You can access `mimi.cs.mcgill.ca` from your personal computer using **ssh** or **putty** and also transfer files to your computer using **filezilla** or **scp** as seen in class and in Lab A and mini assignment 1.

For this assignment, you will have to turn in one C program source code file. Instructors/TAs upon their discretion may ask you to demonstrate/explain your solution. No points are awarded for commands that do not compile/execute at all. (Commands that execute, but provide incorrect behavior/output will be given partial marks.) All questions are graded proportionally. This means that if 40% of the question is correct, you will receive 40% of the grade.

Please read through the entire assignment before you start working on it. You can loose several points for not following the instructions. There are also some helpful hints given at the end of this document that can be useful to you.

Week 8 lectures provides some background help for this mini assignment.

Total Points: 20

Ex. 1 — Data Processing from CSV files Using Linked Lists (XX Points)

In this assignment we will implement a C program, `topcgpas.c` that can read a CSV file containing student information and print a list of top 5 students (based on their CGPAs) by using a linked list to process this data.
A sample format of the input CSV file is as shown below

```
sid,email,lname,fname,cgpa
2123123513,joack@quendeldon.edu,Moe,Jack,3.1
2173413424,leashell@quendeldon.edu,Shell,Leache,3.0
```

The first line is the header (names of fields) followed by actual information. As can be seen, the student ids are 10 digit numbers, followed by email, last name, first name and the CGPA of the student, X.Y a floating point number. The data records themselves are not in any particular order (random).

1. Use `vim` to create a C program source file `topcgpas.c` . All of your source code logic should be contained in this file. You may write multiple functions, etc. (You are encouraged to do so). **Please read through the entire assignment before you start working, as the programming constructs that you are allowed to use are limited in some cases.**
2. **(1 Point)** The source code file should include a comment section at the beginning that describes its purpose (couple of lines), the author (your name), your department, a small "history" section indicating what changes you did on what date.

```
/*
```

```
 Program to generate a report on the students having the top CGPAs
 ****************************************************************
 * Author         Dept.             Date          Notes
 ****************************************************************
 * Joseph D       Comp. Science. Mar 20 2021    Initial version.
 * Joseph D       Comp. Science. Mar 21 2021    Added error handling.
 * Joseph D       Comp. Science. Mar 24 2021    Fixed bug in output logic.
*/
```

The code should be properly indented for readability as well as contain any additional comments required to understand the program logic.

3. The source code should be compilable by (exactly) using the command `gcc -o topcgpas topcgpas.c`. If not, TAs will not grade it and no points will be awarded for the assignment.
4. **(1 Point)** Compilation should not produce any warnings!
5. The program accepts 2 arguments, the first is the name of the input CSV file, and the second is the name of the output CSV file. These file names could be anything, and may have absolute or relative paths.

```
$ ./topcgpas students.csv output.csv
```

6. **(1 Point)** If the program is not passed exactly 2 arguments, it should print a usage message and terminate with code 1.

```
$ ./topcgpas
Usage ./topcgpas <sourcecsv> <outputcsv>
$ echo $?
1
$
```

7. **(1 Point)** If the program cannot open the input file, it should display an error message and terminate with code 1.

```
$ ./topcgpas nosuch.csv output.csv
Error! Unable to open the input file nosuch.csv
$ echo $?
1
$
```

8. **(1 Point)** If the input file is empty, the program should display an error message and terminate with code 1.

```
$ ls -l empty.csv
-rw------- 1 jdsilv2 root 0 Mar 20 15:54 empty.csv
$ ./topcgpas empty.csv output.csv
Error! Input CSV file empty.csv is empty
$ echo $?
1
$
```

9. **(1 Point)** The program should behave the same as above, if the input file only has the header (as from the program's perspective there is no data for it to work on).

```
$ cat headeronly.csv
sid,email,lname,fname,cgpa
$ ./topcgpas headeronly.csv output.csv
Error! Input CSV file headeronly.csv is empty
$ echo $?
1
$
```

10. **(6 Points)** The program should write the sid, email and cgpa of the top 5 (based on their CGPAs) students into the output file. The output file should have a header and the records should be printed with the students with the highest CGPAs on the top (i.e., ordered based on decreasing CGPAs). If two students have the same CGPA, they can be in any order. Existing output files should be overwritten (not appended). On successfull execution, the program should terminate with code 0.

```
$ cat students1.csv
sid,email,lname,fname,cgpa
2123123513,joack@quendeldon.edu,Moe,Jack,3.1
... truncated for brevity ...
2161242311,pcruz@quendeldon.edu,Cruz,Pedro,2.5
2161242311,alope@quendeldon.edu,Lopez,Angela,3.5
$ ./topcgpas students1.csv output.csv
$ echo $?
0
$ cat output.csv
sid,email,cgpa
2161242311,alope@quendeldon.edu,3.5
2169133134,hholmes@quendeldon.edu,3.4
2173413424,leashell@quendeldon.edu,3.3
2178423131,clairec@quendeldon.edu,3.1
2123123513,joack@quendeldon.edu,3.1
$
```

11. **(1 Point)** If the program was not able to open an output file for writing, it should display an error message and terminate with code 1.

```
$ ./topcgpas students1.csv cannot.csv
Error! Unable to open the output file cannot.csv
$ echo $?
1
$
```

12. **(1 Point)** The program should create the output file **ONLY** if it is going to produce valid output.
13. **(3 Points)** If the input file has less than 5 students, they should still make it to the output, in the proper format and order as described above.
14. **(3 Points)** If there are any other students with the same CPGA as the fifth student that was written to the output, all those additional students must also be written to the output file (i.e., the total number of students written to the output file can go above five in this particular case).
15. Your program must read through the input CSV file **ONLY ONCE**, adding a student record in the linked list as you read through each student data in the input CSV file. After that, your program logic should then process this linked list to find the top CGPA students. You should not copy over the student / CPGA information into a new array, etc. **-10 points** deduction if any of this is not followed.
    You can use the below structure as the "node" of the linked list.

```
struct StudentRecord
{
  long sid;
  char email[30];
  char lname[20];
  char fname[20];
  float cgpa;
  struct StudentRecord *next;
};
```

    See the **HINT** section for some helpful suggestions.
16. If the program runs out of memory when using `malloc` or `calloc`, it should display an error message and terminate with code 1.

```
$ ./topcgpas students1.csv output.csv
Error! program ran out of memory
$ echo $?
1
$
```

17. The program must be completely written in C, and should not involve shell scripts, commands, etc. (for example invoked using the `system` function call. Any such approaches would result in an automatic 0 for the entire assignment.
18. You are only allowed to use the library functions from the libraries `stdio.h` `stdlib.h`, and `string.h`. Feel free to write your own functions.
19. The format of the output file should match what is given above, any variations would result in **-2 points** deduction.
20. If your program crashes for one or more test cases, it will result in an **additional, -4 point** deduction from the total assignment score in addition to losing appropriate points for those test cases.
21. If you program allocates memory using `malloc` or `calloc`, make sure that you free all of them explicitly at the end of the program (even if it terminates without producing an output). Not doing this would result in **-3 points** deduction.
22. If your program is stuck (infinite loop) and the TA has to terminate it forcefully, it may not get tested for the remaining test cases in TA's tester script. There will be a fair attempt made to put complex test cases that might trigger such scenarios only towards the end.

WHAT TO HAND IN

Upload your C program `topcgpas.c` to MyCourses. Please download your submitted files to double check if they are good. Submissions that are corrupted cannot be unfortunately graded.

DO NOT turn in the executable `topcgpas`. TAs will compile your C program on their own as indicated in the problem descriptions above.

ASSUMPTIONS

- You can assume that the values of different fields of student records will fit within the individual members of the structure given as part of the assignment description. This should help you understand how "wide" a record can get and then you can make use of that information in reading the input CSV file.
- You do not have to look for "bad data" in the student records - i.e, no sids with non-numeric characters in it, etc.
- You can assume that the names of the fields in the header and their order remains the same.

HINTS

This is a high-level outline to help get you started with the logic. You are not obliged to follow it.

- Read a line at a time using `fgets` into a character array of sufficient size.
- Use the function `strtok` to parse the individual fields. However, I highly recommend that you read this[1] article on how `strtok` works internally before you venture out.
- Once a student record is created, add it to the linked list. You might find it beneficial for the rest of your program logic to add any new nodes in such a way that the list maintains the decreasing order of CGPAs.
- Once you have read all the student information into the linked list, traverse the linked list and write the required information to the output file. You will have to use some logic to keep track of as to when to stop writing.

[1] `https://icarus.cs.weber.edu/~dab/cs1410/textbook/8.Strings/strtok.html`

RESTRICTIONS

Please follow the instructions given for C program at the end of Ex.1, on the features and functionality that you are allowed to use.

TESTING

- There is no mini tester associated with this assignment. You are encouraged to write tester scripts to test your program's behaviour. Start with the very basic (like no arguments, empty file, file that has only a header, file with couple of records, etc.) so that you can easily look at the output to see if the basic logic is working before getting into more complex test cases.
- **Students are allowed to team up with ONE other student to exchange their tester scripts (not obliged).** However, you must record the name and McGill email id of the student with whom you are exchanging the tester scripts at the top of your C program comment section, at the end of the "history" comments.

  ```
  * Exchanged tester script with Jane Doe, jane.doecoder@mail.mcgill.ca
  ```

  Remember !! do not exchange your C program or parts of the associated code. That will be treated as plagiarism. Make sure that your are not accidentally sending your C program over. We will also be manually verifying the submission of the two of you (along with the regular class-level plagiarism checking softwares) to ensure that you have not plagiarised each other's code.
- TAs will test using their own scripts, but it will follow the assumptions that have been setforth in the assignment description. Do not turn in your own tester script.

QUESTIONS?

Please use piazza. However, before posting your question, use the search functionality to check if it has been already discussed. You should also look under "Mini 5 general clarifications" pinned post to check if a popular question has been already included there. They will not get individual responses again.

TAs will not help you with logical errors, you are already expected to have some idea of developing basic programming logic and debugging them from your introductory programming courses like COMP 202. So do not ask questions of the sort "My code is not printing the correct output" - that is because your logic is not correct. Use `printf` statements to print the values of variables, pointers, etc., as you are debugging the code to verify if your programming logic is correct. If your program crashes, use `gdb` to debug your program. Remember, working on this assignment is going to be a great practice for your final project.

TA help is limited to clarifying assignment requirements and helping with C programming language syntax and semantics related questions.