Lab 05 Specification – Exploring Structs in C and Try out Performance Assessment.
Due (via your git repo) no later than 2 PM, Friday, 1st October 2021.
50 points

Lab Goals

- Exploring the implementation of Structs in C.
- Exploring logic by developing projects using Structs and Arrays.
- Exploring performance assessment strategies.

Learning Assignment

If not done previously, it is strongly recommended to read all of the relevant "GitHub Guides", available at the following website:
`https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading assignment is useful to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, it is also recommended to do the reading assignment from the section of the course textbook outlined below:

- **KR chapter 5 - [5.6 - 5.9] and chapter 06 - [6.1 - 6.4], PH Chapter 1.6**

Assignment Details

Now that we have discussed some more of C Programming, which is in Structs and Arrays, we are ready to explore and do a few more exercises to think logic, use low end computing tools such as Pointers and Arrays, and implement Structs in C programs. We will also have an opportunity to more practice performance assessment strategies.

At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL(s) to clarify if there is any confusion related to the lab and/or class materials. The Professor proofread the document more than once, if there is an error in the document, it will be much appreciated if you can communicate that to the Professor. The class will be then informed as soon as possible regarding the error in the document. Additionally, it is highly recommended that students will reach out to the Professor in advance of the lab submission with any questions. Waiting till the last minute will minimize the student's chances to get proper assistance from the Professor and the Technical Leader(s).

Students are recommended to get started with this part in the laboratory session, by discussing ideas and clarifying with the Professor and the Technical Leader(s). It is acceptable to discuss high-level ideas with your peers, while all the work should be done individually. Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.
2. Submitting a copy of the other's program(s) and technical reports is strictly not allowed. Please note that all work done during lab sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as Quizzes, exams, etc · · ·

Preliminary Steps

It is important that you can set up Docker and GitHub to complete the rest of the lab. Please follow the guidelines below to complete the preliminary steps.

1. **[Docker Setup.]** At this point, I expect the MAC, Linux, and Windows Pro users, to have this step completed based on our previous class discussions. For those who had not completed this step, the documentation below should provide more details regarding the download and installation setup.

   - Get Docker setup completed on your laptops:
   - Docker Mac Setup:

     `https:/docs.docker.com/docker-for-mac/install/`
   - Docker Ubuntu Setup

     `https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04`
   - Docker Windows Setup:

     `https:/docs.docker.com/docker-for-windows/install/`
   - If the setup goes correctly as desired, you should be able to get started and validate the Docker version and run the hello world docker container using the following commands:

     docker –version

     docker run hello-world
   - There are some more documentation for Docker get started to test your installation in the link provided below:

     `https:/docs.docker.com/docker-for-mac/`

     `https:/docs.docker.com/docker-for-windows/`

2. **[Loading Docker Container.]** There are two steps in loading the container, namely:

   - Build the container
   - Connect and Run the container

   Build the container: So to build the container. the following steps should be performed.

   (a) First, accept the lab URL provided in Slack. After downloading the lab folder from the GitHub classroom, navigate to the cmpsc200-fall-21-lab05 directory using terminal (Mac/Ubuntu) or Command Prompt/Docker quick start terminal (windows).

   (b) Build the docker image using the following command:

   docker build -t cs200lab05 .

   Please note, you are required to have the period in the command above.

(c) Note: In the command above, cs200lab05 is the user-provided image name. This could be random. But it is recommended to use the same name to easily follow the rest of this document. Additionally, it is required to be inside the `cmpsc200-fall-21-lab05` directory to run the build command. If you are not inside the `cmpsc200-fall-21-lab05` directory, you may receive an error message.

(d) Upon successful build, it is recommended to verify the correctness of image creation by using the following command:

docker image ls

(e) The image named "cs200lab05" should be listed as one of the outputs from the command above.

Connect and Run the container: So to create and run the container. the following steps should be performed.

(a) Run the docker container based on the image created in the previous steps using the following command:

Mac/Ubuntu:

```
docker run --rm -v $(pwd)/src:/root -it cs200lab05
```

Windows:

```
docker run --rm -v "%cd%/src":/root -it cs200lab05
```

(b) To run the above command, it is required to be inside the `cmpsc200-fall-21-lab05` directory. And, please note, you will log in to the container after entering the above command.

(c) After creating the container, the run command above creates a mount between the host machine and the container with a shared folder space. So, any files placed inside the host mount directory can be easily accessible inside the container mount directory and vice versa.

(d) After connecting to the container, we can compile C Programs using the command below within the container:

gcc hello.c -o hello.out

(e) After compiler the program, we can execute C Programs using the command within the container:

./hello.out

(f) **[GitHub Setup.]** Take a look at the detailed documentation for getting started with GitHub, which is available at: `https://docs.github.com/en/get-started`

You are required to know the procedure to git clone, git pull, git add, git commit, and git push to access the lab specification folder and to submit your lab for grading purposes. If there is an issue with your GitHub setup please discuss it with your Technical Leader(s) and/or the Professor.

Section 1: Performance Assessment

This section is worth 15 points.

We discussed Performance Assessment strategies, during our last two classes. Please refer to the Week5 slides and activities. Complete the task by adding your detailed answers to the `Technical-Report` file.

Let us suppose that we are given with five machines with the following specifications.

- **MACHINE-A:** 2 GHz clock rate with 6s CPU time.
- **MACHINE-B:** 8s CPU time and number of clock cycles is 2.2 times as much as the clock cycles of A.
- **MACHINE-C:** 12s CPU time and number of clock cycles is 1.2 times as much as the clock cycles of B.
- **MACHINE-D:** 6s CPU time and number of clock cycles is 1.6 times as much as the clock cycles of C.
- **MACHINE-E:** 5s CPU time and number of clock cycles is 1.8 times as much as the clock cycles of D.

First, identify the clock rate of machines B, C, D, and E respectively. Next, identify which machine is the fastest. Add your solution in the `Technical-Report` file.

Section 2: Struct Swapping Program

This section is worth 15 points.

We discussed Struct implementation in C, during our last two classes. Please refer to the Week4 slides and source code. We defined a struct, declared a variable of the struct type and initialized the members of the struct, and processed the struct. We did this for both a single struct element and an array of struct elements. The goal of this program is two create two arrays of structs for all the patients in floor 1 and 2 respectively. After creating this array of structs, the program should be able to swap the patients in the two floors by using a separate method. To complete this section, implement the `patient.c` program in the starter code by adding code to handle the following requirements. The starter code is totally incomplete and we need you to complete the code by carefully implementing the requirements listed below:

- **R1:** Define a struct named **patient** with six members, that is, Patient ID **(int)**, Patient Full Name **(char *)**, Weight **(float)**, Height **(float)**, Disease **(char *)**, and Floor **(int)**.

- **R2:** Declare two struct arrays of type **patient** with the name floor1 and floor2 respectively of size 5 each. We may assume that the total number of patients in both floors combined is 10. You may create a static array of structs or dynamic array of structs using the pointer approach. Refer students.c in Week4 folder and products.c in Week5 folder for creating an array of structs. Next assign the values for each element in the struct array with the corresponding patient details provided in the patient table given above. This declaration should be done in the main method. You may load the struct members directly by hard coding the values in the code or by creating a separate file to store the data, and generating the array of structs by reading from that file. This part is open ended and you may select an approach that works well for you.

Patient ID	Patient Full Name	Weight	Height	Disease	Floor
10011001	Alex Crowe	210.5	5.6	Tuberclosis	1
10011002	Amelia Kaur	161.6	6.2	Asthma	1
10011003	Amanda Daya	171.3	5.8	Heart Attack	1
10011004	Ben Krish	151.3	5.3	Stroke	1
10011005	Brian Miller	181.5	5.9	Urinary Tract Infection	1
10011006	Chris Miller	251.3	6.4	Asthma	2
10011007	Drew Millan	178.4	5.7	Hypertension	2
10011008	Frank Derrick	191.9	6.0	Sarcoidosis	2
10011009	Robin Meade	271.3	6.8	Lung Cancer	2
10011010	Rosy David	131.3	5.3	Diabetes	2

Table 1: Patient Database Table

- **R3:** Add your implementation to the **Swap** method. First add two parameters for the method, that is the first parameter, a struct array named floor1, and the second parameter a struct array named floor2 to the method definition. Next add your logic to the method for swapping the patients in floor1 with floor2 and vice versa. Please note, you need to have the parameters defined as pass by reference. In this way, the caller that is main method, is able to see the changes. The changes are simply the modified content.

- **R4:** Add your implementation to the **Display** method. First add two parameters for the method, that is the first parameter, a struct array named floor1, and the second parameter a struct array named floor2 to the method definition. Next add your logic to the method to iterate and print the contents of the struct array. Make sure to print the output tab separated and in new line for every row. Please note, you may define the parameters as pass by value or pass by reference. Please note, the caller, that is main method, can call this to display both the initial and modified values of the two structs.

- **R5:** Add more implementation to the **main** method. In **R2** method, you had already defined the struct array, initialized with the values in the main method. Call the display method to print the contents of the struct array on the console. Next call the swap method by using pass by reference. After the call returned the struct arrays floor1 and floor2 should be swapped. That is the patients in floor1 and floor2 should be swapped. Now call the display method again to print the modified content or the swapped content of both the struct array. This completes the section.

Section 03: A Songs Statistics Program

This section is worth 20 points.

A program named `songs.c` is provided in the starter code. This program includes the implementation to generate a text file using randomized data points generated for each song. Each song, has a number (ID), count (number of times heard), rank (a number between 0 to 5), and duration (a number between 0 to 6.5). These randomized data points are generated and written to a file named `data.txt` file. Next the program initializes a struct array and store the data points into the array by segregating the members of the struct. The program requires some additional implementation to be complete. That is, a new set of logic should be added to the `report` method to display some statistics on the songs dataset. Please make sure to complete the following requirements:

- **R1:** First, read through the source code in `songs.c` program file. Write a detailed technical report to include what each section of the code. It is recommended to highlight the **technical details** by referencing the line numbers. Please don't be abstract, I am looking for the technical details. Reference line number and add the technical details to summarize what is going on the different sections of the program, starting from the main method to the other methods such as write, load, and report.
- **R2:** Complete the implementation in the `report` method to answer the top 8 questions on the dataset. Add the required logic to figure out the answers for these 8 questions. Once you added the logic, and bind the results to the corresponding variables, then the output will be displayed by the starter code. For more information on the variable bindings, refer to the comments in the source code. Also, please refer to the **product.c** program that we discussed in class. This program should be located in the Week5 folder. The 8 questions are provided below for your reference:
    1. Identify the least played song.
    2. Identify the most played song
    3. Identify the least heard song
    4. Identify the most heard song
    5. Identify the highly ranked song
    6. Identify the least ranked song
    7. Identify the shortest song
    8. Identify the longest song

Section 04 - Honor Code

Make sure to **Sign** the following statement in the `honor-code.txt` file in your repository. To sign your name, simply replace Student Name with your name. The lab work will not be graded unless the honor code file is signed by you.

This work is mine unless otherwise cited - Student Name

Section 05 - Reflection

Add a Reflection to the repository by modifying the `reflection` file in the lab repository. List out the biggest learning points and any challenges that you have encountered during this lab.

Submission Details

For this assignment, please submit the following to your GitHub lab repository.

1. updated version of **songs.c** file.
2. updated version of **swap.c** file.
3. A document containing the technical points in the file named `Technical-report`.
4. A document containing the reflection of the lab in the file named `Reflection`.
5. A signed honor code file, named `Honorcode`.
6. To reiterate, it is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus.

Grading Rubric

1. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits may be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your GitHub repository will lead to receiving no points given for the lab submission. In this case, there is no solid base to grade the work.
3. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before the due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in GitHub. In this way, an updated version of the student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then, in this case, there are no points awarded to the student.
4. If a student needs any clarification on their lab grade, it is strongly recommended to talk to the Professor. The lab grade may be changed if deemed appropriate.