Lab 3 Specification – Exploring Multi Dimensional Arrays in C
Due (via your git repo) no later than 8 AM, Thursday, 29th September 2022.
50 points

Lab Goals

- Exploring the implementation of Arrays.
- Exploring the internal application and the usage of Pointers.

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 chapters 5 and 6**

Assignment Details

Now that we have discussed some more of C Programming Concepts, we are ready to explore and do a few more exercises to think logic, use low end computing tools such as Multi Dimensional Static and Dynamic Arrays, Pointers, and implement it in C programs. We will also get an opportunity to implement these in a practical use case, that is in a simple project level requirement that was discussed in class.

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-22-lab3 directory using terminal (Mac/Ubuntu) or Command Prompt/Docker quick start terminal (windows). It is recommended to use Command Prompt to get better results.

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

   `docker build -t cs200lab3 .`

   Please note: There is a dot towards the end of the command. It is required to include the dot at the end of the command.

   (c) Note: In the command above, cs200lab3 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-22-lab3` directory to run the build command. If you are not inside the `cmpsc200-fall-22-lab3` 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 "cs200lab3" 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 cs200lab3`

   Windows:

   `docker run --rm -v "%cd%/src":/root -it cs200lab3`

   (b) To run the above command, it is required to be inside the `cmpsc200-fall-22-lab3` directory. And, please note, you will log in to the container after entering the above command.

   (c) **Please note:** If you are windows user, you will get better results by running this command using the Command Prompt instead of Powershell.

   (d) 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.

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

   gcc hello.c -o hello.out

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

   ./hello.out

   (g) **[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: FBI Suspect and Nine States

This section is worth 25 points.

We implemented a simple FBI suspect program in class, that uses multiple storage layers to load, process, and streamline the raw simulated data to suggest the suspect location. In this task, we will practice some variations to transform the dataset differently and logically solve the problem to identify and suggest the list (top 3) of states where the suspect may be located. To practice and understand the storage layers, and understand the usage of Pointers, it is required to visualize the requirements, and complete the tasks listed below:

Task 1 → Overview

FBI officers are trying to narrow down and search for a suspect based on their location. FBI currently has access to the data points that lists the locations traveled by the suspect. The data points are interpreted as the points corresponding to Nine different states based on their start digit. A table below is provided to map the start digit to a particular state. This is a high-level requirement.

We need to think, design, develop appropriate logic, and implement the program. It is very important to visualize the requirement, by the use of the whiteboard, and/or your notebook, or pen and brainstorming with your colleagues.

The starter code is provided in the src/fbi.c file. You may use as many variables as additional storage in the program. However, the lesser you declare the more efficient your program is implemented. In general, the efficient implementation has fewer lines of code.

Assumptions:

1. The data points (zip codes) are randomly generated by the program. The points are between 10001 and 100000 in the dataset. The data size is fixed to 100, and this may be changed for your testing purpose.

Core Logic:

1. Modify the report1 method, from class discussions to show the one dimensional array contents using 5 values in one row. In class, the report displayed all the values in the array tab separated. This modification is required to be completed.
2. Add logic to the transform method to show the location count based on the starting digit. The starting digit for the locations is labeled 1 to 9. This assumption is done in the initialization of the 2d array in the main method. There are 9 rows and 2 columns in the 2d array. Questions: Is the 2d array static or dynamic? How about the 1d array, is it static or dynamic?

3. Incorporate the code from class discussions into the report2 method to display the two dimensional array contents in a similar format as we did in the class. It is appropriate to use the same code used in class for this method.

4. Add logic to the suspect method to display the top three suspect states (not their count) from the two-dimensional array. I am looking for the state names, and the state name corresponding to 1 to 9 start digits are shown below for reference. Now, this is the greatest logical challenge in this section. I ask that you team up and find peers, technical leader(s), and the Professor to discuss and brainstorm ideas for formulating the right logic for solving this puzzle. Thinking, Thinking, Thinking ... is the only way we can practice logical problem-solving skills. To give an example from my experience, when I was given this problem for the first time, I thought about this for atleast a day, before coming up with a correct working solution. Now solving problems like this comes with a reward of experience and this experience will open up the doors to solving more such logical problems.

   However, let us suppose you come to a point where you exhausted all your energy, and need some logical clues, then you may review, analyze, and execute the code top.c in the resources section. This code is a very good indicator of how to display top candidates from one-dimensional array. Although one can get a basic idea from here, it is still required to be applied in a different context for the 2d array to get the method functional.

   1 - Michigan 2 - New Jersey

   3 - Pennsylvania 4 - Massachusetts

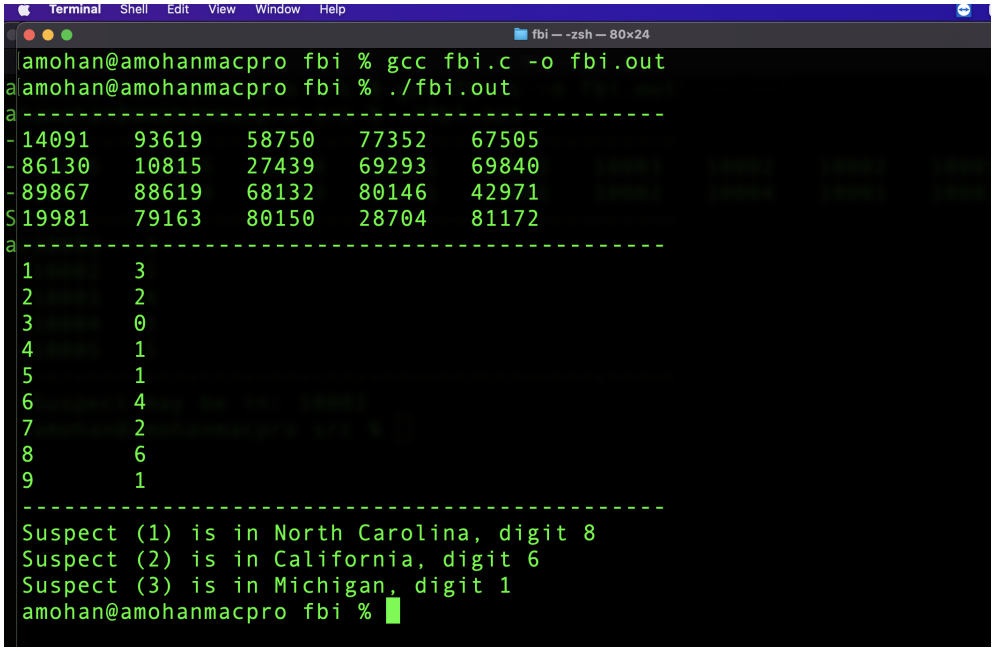
   5 - Georgia 6 - California

   7 - Delaware 8 - North Carolina 9 - West Virginia

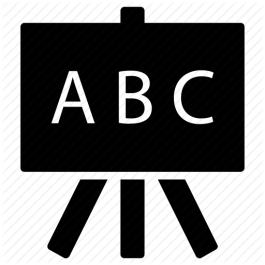
Validation:

You are welcome to make the data size smaller for your testing purpose. It is recommended that you should atleast consider the size between 50 and 100 to make sure that the code is validated correctly and fully functional. A sample output from the fb1 program is provided below.

A sample screenshot for output ....



Section 2: Eye Examination Program.



This section is worth 25 points.

We discussed Dynamic 2 Dimensional Array implementation in C, during our last class. Please refer to the Week4 slides and source code. We iterated through a 2D dynamic array by first setting it up using **malloc** keyword. In this section, we are going to practice setting up and iterating through a 2 Dimensional Dynamic array. In this process we will develop an eye examination program to solve an interesting real life problem. A starter code is provided in the src folder named **eye.c** file.

Task 2 → Overview

This task should be implemented using two-dimensional data storage. The starter code is provided in the src/eye.c directory. You may use as many variables and intermediate arrays as additional storage in the program. However, the lesser you declare the more efficient your program is implemented. In general, the efficient implementation has fewer lines of code.

Assumptions:

1. We have two grids of random alphabets. The first grid is for the eye examiner to test the patient. The second grid is the results after the patient reads out the letters from the grid.

Core Logic:

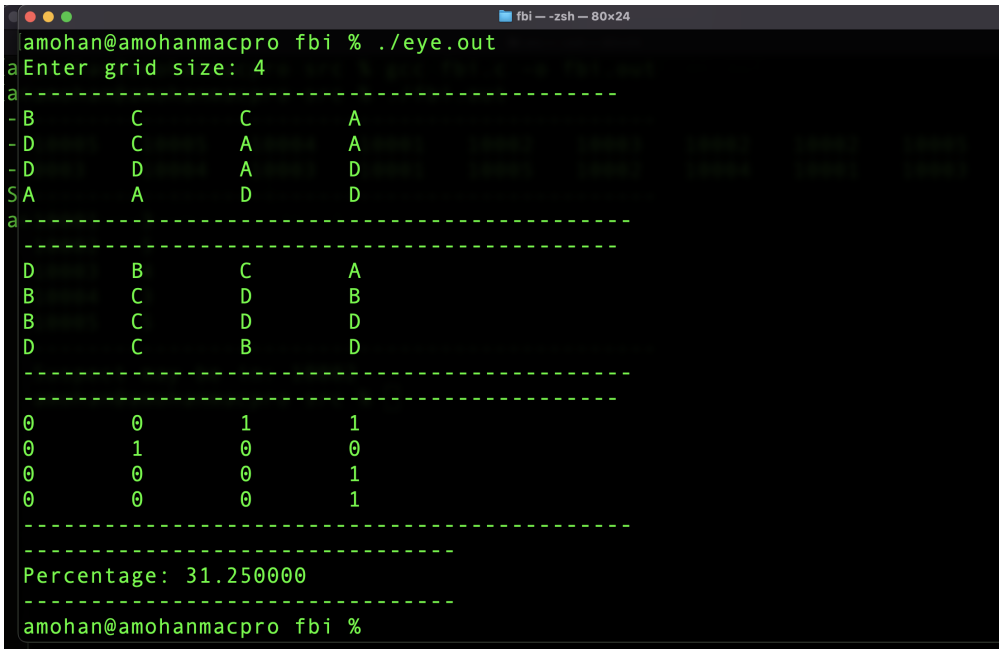
1. The program is expected to utilize both the grids and find the match and build a new grid to show the results. The third grid is supposed to indicate a 1 for correct reading (match) and a 0 for incorrect reading (mismatch) by the patient. In the other words the program is expected to compare both grids and show the comparison results in the third grid using 1's and 0's. The program then is expected to show a percentage result to indicate the total correct percentage.
2. Complete the **generate** method by adding your logic to generate the new grid with 1's and 0's inside grid3. You are expected to use pointer arithmetic for computing the new grid. The data from grid1 and grid2 is already loaded inside the main method using random distribution of characters and passed by reference. You are expected to build the new grid and bind it to the pointer variable **grid3**.
3. Complete the **display** method by changing the print logic to accomodate both integers and characters. That is for grid1 and grid2 it should print characters and for grid3 it should print decimals. Also the key challenge here is that we are not directly provided the information about all three grids at the same time so we can only infer that through logic.

4. Next complete the **report** method to show the percentage of total number of correct responses based on the patient's reading results in **grid3** array. A sample screenshot is shown below for your reference.

Validation:

The first and second grids are loaded using random Alphabets. It is important to make sure the following test case (similar format) is validated.

A sample screenshot for user input ....



Section 3 - 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 4 - 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, completed version of **fbi.c** file.
2. updated, completed version of **eye.c** file.
3. 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.
4. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is completed before the deadline. The honor code policy can be accessed through the course syllabus. Make sure to sign the honor-code file.

Grading Rubric

1. Details including the points breakdown are provided in the individual sections above.
2. If a student needs any clarification on their lab credits, it is strongly recommended to talk to the Professor. The lab credits may be changed if deemed appropriate.