Lab 04 Specification – Exploring Arrays and Pointers in C
Due (via your git repo) no later than 2 PM, Friday, 24th September 2021.
50 points

Lab Goals

- Exploring the implementation of Dynamic 1 Dimensional Arrays.
- Exploring the implementation of Dynamic 2 Dimensional Arrays.
- Exploring the logic behind the usage of Arrays.

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]**

Assignment Details

Now that we have discussed some more of C Programming, that is in Dynamic Arrays, we are ready to explore and do a few more exercises to think logic, use low end computing tools such as Pointers, and implement Dynamic arrays in C programs.

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-lab04 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 cs200lab04 .

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

(c) Note: In the command above, cs200lab04 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-lab04` directory to run the build command. If you are not inside the `cmpsc200-fall-21-lab04` 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 "cs200lab04" 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 cs200lab04
```

Windows:

```
docker run --rm -v "%cd%/src":/root -it cs200lab04
```

(b) To run the above command, it is required to be inside the `cmpsc200-fall-21-lab04` 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: An Investigation Program

This section is worth 25 points.

We discussed Dynamic 1 Dimensional Array implementation in C, during our last class. Please refer to the Week4 slides and source code. We iterated through a dynamic array by first setting it up using **malloc** keyword. In this section, we are going to practice setting up and iterating through a 1 Dimensional Dynamic array. In this process we will develop an investigation program to solve an interesting real life problem. A starter code is provided in the src folder named **fbi.c** file.

Let us suppose the FBI is investigating a case and looking for a criminal. As part of the investigation they were able to collect a dataset that contains the phone numbers dialed by the criminal. The phone number dataset contains a series of numbers starting from 1 to 9. These numbers are generated randomly in the starter code. Now the next step is to store these numbers into an 1 Dimensional Dynamic Array and then produce a report to indicate the total number of phone numbers under each group. The groups are based on the starting digit in the phone number, that is 1 to 9.

We make an assumption that each digit (starting digit of the phone number) is connected to a state. So there are 9 states according to the group definition and the randomly generated dataset. The overall purpose of this program is that once the report is provided then we can easily identify the numbers dialed and narrow down the search for the criminal in the states that has the most dialed numbers. The starter code makes these assumptions and generate the phone numbers in a file named `data.txt` when the user executes the program.

So you need to do three tasks to complete this investigation program:

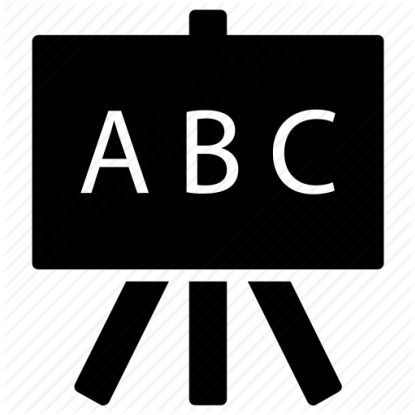
- **Task-1** Complete the load method by iterating through the characters from the file and segregating all the individual phone numbers. The dataset has 1000 phone numbers, which is defined by the global constant SIZE. Now the structure is already provided in which the while loop in the method is able to extract the characters one at a time. The logic part need to be figured out. Similar to the previous lab, you may need to identify the phone numbers using newline and tab separation. Additionally, you may need to look at the **ascii.c** program in Week2/src folder to review how we developed the logic to orchestrate numbers from characters by assembling digits. The starter code is using a special data type called unsigned long. This is similar to an int. The key difference is that unsigned long allows us to store more bytes and hence able to store large numbers. The format specifier for unsigned long is **%lu**. The expectation for this method to be complete is to use pointer arithmetic to load the data from the file to the array named **arr**. Note, here we use pass by reference, and hence the caller, that is **main** method, can access the modification to the structure outside the function **load**.

- **Task-2** Complete the report method by iterating through the elements in the array named **arr**. The expectation is that this method should retrieve the total count of phone numbers for every group. The groups are defined by the starting digit from 1 to 9. That is, there is a group for all phone numbers starting from 1, and a group for all phone numbers starting from 2, and so on. The last group is for all phone numbers starting from 9. Additionally, indicate the top three suspected states based on the dialed numbers. A screenshot of the sample output is shown in next page. Note: The output is expected to change every time we run the program, because the dataset is randomly generated in the **write** method. You may need to change the size of the phone number dataset during the development phase so you can test it better.

```
1:212
2:90
3:115
4:310
5:35
6:33
7:10
8:150
9:45
Most likely states: 1, 4, 8
amohan@amohanmacpro Desktop %
```

- **Task-3** Analyze the starter code in the **write** method and answer the questions provided in the **Technical-Report** file as detailed as possible.
- For both the previous tasks, you are expected to modify the **fbi.c** program as part of the deliverables!

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.

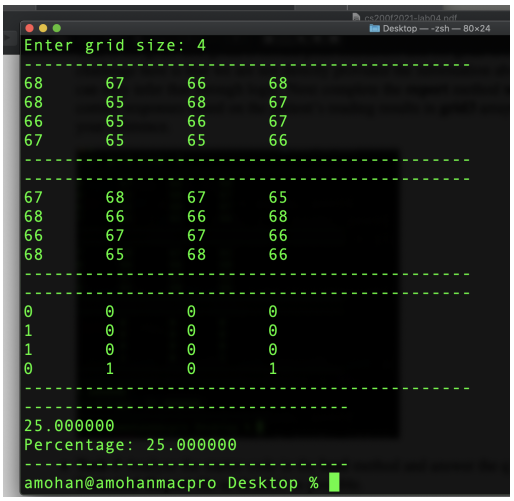
We have two grids of random alphabets. The first grid is for the eye examiner to test the patient. The second grid is the results after the patient reads out the letters from the grid. The program is expected to utilize both these grids and find the match and build a new grid to show the results. The third grid is supposed to indicate a 1 for correct reading (match) and a 0 for incorrect reading (match) 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.

So you need to do three tasks to complete this investigation program:

- **Task-1** 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**.

- **Task-2** 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. 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.



- **Task-3** Analyze the starter code in the **load** method and answer the questions provided in the **Technical-Report** file as detailed as possible.
- For both the previous tasks, you are expected to modify the **eye.c** program as part of the deliverables!

Section 03 - 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 04 - 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 **fbi.c** file.
2. updated version of **eye.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.