Lab 01 Specification – Docker Installation, and Exploring Binary Match Table
Due (via your git repo) no later than 2 PM, Friday, 3rd September 2021.
50 points

Lab Goals

- Review basic lab tools (Docker and GitHub).
- Practice Binary Match Table.
- Practice Simple C programming.

Summary

You will do a few exercises just to refresh your memory of git, install and set up docker, explore the binary match table, and write a short C program to implement the logical solution of the binary match table, reflect on class materials by answering questions related to binary match table, and prepare to write several more C programs in the course.

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

If you have not done so already, please read all of the relevant ”GitHub Guides”, available at https://guides.github.com/, that explain how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as ”Mastering Markdown” and ”Documenting Your Projects on GitHub”; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- PH chapter 01 - section 1.1 - 1.4.

Assignment Details

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. Those who had not completed this step, the documentation below should provide more details regarding the download and intallation 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 in order to build the container. the following steps should be performed.

   (a) First accept the lab url provided in the Slack. After downloading the lab folder from the GitHub classroom, navigate to the cmpsc200-fall-21-lab01 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 cs200lab01 .

   (c) Note: In the command above, cs200lab01 is the user provided image name. This could be random. But it is recommended to use the same name so as to easily follow the rest of this document. Additionally, it is required to be inside the `cmpsc200-fall-21-lab01` directory in order to run the build command. If you are not inside the `cmpsc200-fall-21-lab01` 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 ”cs200lab01” should be listed as one of the output from the command above.

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

   (a) Compile the C Program by running 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 cs200lab01 gcc hello.c -o
   hello.out
   ```

   Windows:

   ```
   docker run --rm -v $(pwd)/src:/root cs200lab01 ./hello.out
   ```

   (b) To run the above command, it is required to be inside the `cmpsc200-fall-21-lab01` directory. And, please note, you will not receive any output after running the command above. It will simply prepare a binary file for execution.

(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) Execute the C Program by running 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 cs200lab01 ./hello.out
```

Windows:

```
docker run --rm -v "%cd%/src":/root cs200lab01 ./hello.out
```

(e) You are required to compile the program first before executing it. At this point, you will see a hello world message printed on your screen. That is, similar to the line below:

Hello CS-200

3. **[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`

4. **[Modify Hello World C program.]** First, edit the **Honor-code** file to include your name to accept the class honor code policy. Next, modify the `hello.c` program in the `src` directory to include your name in the print statement. That is your program needs to print a similar greeting message as shown below, but using your full name.

   Hello CS-200. I am Aravind Mohan!

   Compile and Execute the modified program using the similar procedure outlined above. You will be graded based on the modification in `hello.c` program file in the `src` directory. This part is worth **10 points**.

5. **Prepare the 5-bits binary match table.**

   (a) By using the labels [b4,b3,b2,b1,b0] left to right, prepare the 5-bits binary match table. A part of this task is to figure out the number of rows in the BMT (Binary Match Table) and fill the table using the **Toggling Method** discussed in class. An important point to note here, what is the toggling sequence for each bit? and in what order does the toggling sequence work? A question to expect in the Quiz/Exam is:

   "In a 8-bit BMT, how frequently does b6 toggle. Assume bit arrangement left to right as b7 to b0." Although no submission is required for this question, you should prepare for scenarios like this.

   Modify the file named **5-bits** in the `tables` folder. Make sure to maintain a similar structure, as the starter solution, to implement your BMT by copy/paste the blank rows, so that it is easier to read the table. This part is worth **15 points**.

6. **Prepare the 4-bits binary match table using the Tabular Approach.**

   (a) The tabular solution for 3-bits BMT is provided inside the `tables` directory in a file named **3-bits.csv** document. First, open **Google Sheet** using google drive. Then select File-Open and upload the **3-bits.csv** document. This document contains the detailed solution for the 3-bits BMT solved using the tabular approach. The algorithm discussed in class is provided below for your reference:

```
1: for (i = 0; i < rows; i = i + 1) do
2:     for (j = cols − 1; j >= 0; j = j − 1) do
3:         Print ( ⌊ i / 2^j ⌋ %2)
4:     end for
5: end for
```

Formula 1: The number of rows, given the number of columns is identified using the formula $rows = 2^{\text{cols}}$

Formula 2: The number of columns, given the number of rows is identified using the formula $cols = \lceil log_2(rows) \rceil$

Prepare a new sheet using the similar structure and name the files as **4-bits.csv** document. In this sheet, solve the 4-bits BMT using the tabular approach. That is, you need to figure out the i and j values and the number of rows for each decimal value in the BMT, and importantly fill the value for each bit in the table. For example, each decimal in **3-bits** document contains three rows (For example: when i is 0, j range from 2 to 0). Similar to that you need to figure out the number of rows for each decimal. That is, what are the i and j values? Once you completed the Google Sheet, download the **csv** format file from google drive, and upload the **4-bits.csv** document in the `tables` directory. This part is worth **20 points**.

Submission Details

1. 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.
2. Questions about the lab? Bring them to class on Tuesday morning!

Grading Rubric

1. Task 1-3 in this assignment is a preliminary requirement to complete this lab assignment. There is no explicit points awarded for completion of this task as such. It is highly recommended to take this seriously and complete it as per the instructions provided, so that it can set you up nicely for the rest of the semester.
2. Task-4, is 10 points. You need to submit the modified version of `hello.c` in the `src` directory.
3. Task-5, is 15 points. You need to submit the modified version of **5-bits** file in the `tables` directory.
4. Task-6, is 25 points. You need to submit a new file named **4-bits.csv** file in the `tables` directory.
5. There will be a partial credit (30% of the points allocated for the task) given for submissions that include partial work.
6. There will be no partial credit awarded if you did not submit any work. After the grade is released, you will still have an opportunity to interact with the instructor and your grade may be changed if deemed appropriate. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus.