Lab 07 Specification – Binary Operators
Due (via your git repo) no later than 2 PM, Friday, 29th October 2021.
50 points

Lab Goals

- Refresh our memory on the add, subtract, and shift operators.
- Solidify our understanding of Multiply and Divide operators.

Learning Assignment

If you have not done so already, please read all of the relevant "GitHub Guides", available at https://guides.github.com/, which explains 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

- Section 3.3 in **PH**
- Class discussion notes, slides, and in-class coding files.

Assignment Details

Now that we have discussed some fundamental principles behind low-end operators such as the left and right shift, multiply and divide, and got a lead towards the development of logical operators, it is now time to implement some challenging requirements from an operational perspective. In this process, we will also solve multiply, and divide operations using the computing algorithms 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-21-lab07 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 cs200lab07 .

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

(c) Note: In the command above, cs200lab07 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-lab07` directory to run the build command. If you are not inside the `cmpsc200-fall-21-lab07` 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 "cs200lab07" 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 cs200lab07
```

Windows:

```
docker run --rm -v "%cd%/src":/root -it cs200lab07
```

(b) To run the above command, it is required to be inside the `cmpsc200-fall-21-lab07` 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

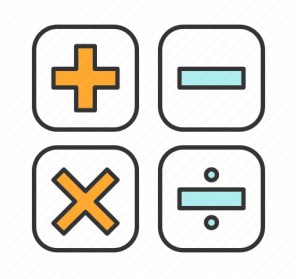
(e) After compiler the program, we can execute C Programs using the command within the container:

./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: Multiply and Divide Operators



This section is worth 30 points. The points breakdown is provided below:

- Task 1 = 15 points
- Task 2 = 15 points

We discussed in-class examples to understand the low-level implementation (and the algorithm) related to multiply and divide operators. In this section, we will further solidify our understanding by practicing the implementation details of these operators. This is also an opportunity for us to start thinking through a final course project idea on implementing a tool that features these operators (add, sub, multiply, divide, log, power) using a C program (from the scratch implementation).

1. **Task 1:** Review the multiply algorithm discussed in lesson-6 (part-2) slides and refer to your class notes. A sample solution of 15×7 is provided in the starter-code solution file named multiply markdown file. Solve 18×13 and edit the multiply markdown file to summarize your solution. Note: final output should be displayed in the last row of the product register.
2. **Task 2:** Review the divide algorithm discussed in lesson-6 (part-2) slides and refer to your class notes. A sample solution of $\frac{12}{5}$ is provided in the starter-code solution file named divide markdown file. Solve $\frac{19}{7}$ and edit the divide markdown file to summarize your solution. Note: final output, remainder and quotient should be displayed in the A and Q registers of the last row respectively.

Section 2: Shift Operator

This section is worth 20 points. The points breakdown is provided below:

- Task 3 = 10 points
- Task 4 = 10 points

It is worth making a note, that the data conversion using shift operators is done by our computers, every nanosecond, tirelessly. In this part, we will implement two distinct methods in the Shift Operator tool. These methods are expected to perform left and right shift execution. To complete this part, it is required to have a solid understanding of the basic concepts of C Programming from class discussions and the previous labs. It is important to review the slides, reason through and understand the logical outflow of the algorithms discussed in class, and also complete the reading assignments as required. There is a C program provided in the starter-code provided repository, to help stay focused on the implementation details, and to get started with the development of this part. To complete this part, it is required to do the following:

1. Review the starter-code in a file named `shift.c`. Execute the code a few times to understand the program flow in the code file. The overall goal of this program is to implement shift operators. Read the program completely, to make sure you understand how the program works. The program prompts to get the user input. This understanding is important to complete the tasks outlined below:
2. **Task 3:** Complete the **left ()** method by implementing the following steps. This method contains one line implementation using the built-in left shift operator. Please go ahead and comment the current line in the left method of the starter-code. The goal here is to develop a series of logical steps, to implement the left shift operator. The steps are outlined below for reference:
   (a) Convert the value stored in the (pass by value) argument input1, from decimal form to binary form. To do this part, please revisit our previous slides and your previous labs, and implement the algorithm discussed.
   (b) Initialize a new array with the size equivalent to the constant no_of_bits. The program assumes that the total number of bits used for shifting is 16 bits. Hence the size of the array is 16.
   (c) Transform the array initialized in the previous step. This transformation should be done using a for/while loop by shuffling the contents and to apply the left shift operator.
   (d) At this point, the array contains the final result in the binary form. The result refers to the output after applying the shift operator. Convert the array contents from binary form back to decimal. Note: this part is already implemented. You need to look at how this part was previously performed. The only difference between this and the one we implemented earlier is that we are using Arrays for this part.
   (e) Store the final output, that is in the decimal form in the (pass by reference) argument res.

3. **Task 4:** Complete the **right()** method by implementing the following. This method contains one line implementation using the right shift operator. Please go ahead and comment the current line in the right method of the starter-code. The goal here is to develop a series of logical steps, to implement the right shift operator from scratch. The steps are outlined below for reference:

   (a) Convert the value stored in the (pass by value) argument input1, from decimal form to binary form. To do this part, please revisit our previous slides, your previous labs, and implement the algorithm discussed.

   (b) Initialize a new array with the size equivalent to the constant no_of_bits. The program assumes that the total number of bits used for shifting is 16 bits. Hence the size of the array is 16.

   (c) Transform the array initialized in the previous step. This transformation should be done using a for/while loop by shuffling the contents and to apply the right shift operator.

   (d) At this point, the array contains the final result in the binary form. The result refers to the output after applying the shift operator. Convert the array contents from binary form back to decimal.

   (e) Store the final output, that is in the decimal form in the (pass by reference) argument res.

   Please note that the description for left and right shift operators are quite similar. This is the case for the implementation as well. If one of the method implementations is completed, then implementing the other method is straightforward and not too time-consuming. A sample test run is for an input 10:

   10 left shift 2 is equal to 40

   10 right shift 2 is equal to 2

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 **shift.c** file.
2. A document containing the technical solution to the multiply problem in the file named `multiply`.
3. A document containing the technical solution to the divide problem in the file named `divide`.
4. A signed honor code file, named `Honorcode`.
5. 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.