Lab 5 Specification – Exploring the Shift Operators
Due (via your git repo) no later than 2 p.m., Wednesday, 28th October 2020.
50 points

Lab Goals

- Refresh our memory on binary to decimal and decimal to binary conversion techniques from lab 4.
- Solidify our understanding of Pass by value and reference.
- Practice implementing the shift 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 1.6 in **KR**
- Section 1.10 in **PH**
- 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, and got a lead towards multiply and divide operators, it is now time to implement some challenging requirements from an operational perspective. In this process, we will also implement a tool using a C program to apply left and right operator on a user-defined number. Additionally, we will watch and reflect on a video to learn some intricacies of the shift operators and signed numbers, to retain the knowledge from the class discussions so far.

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) is strictly not allowed. Please note that all work done during laboratory 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 lab assignments, skill tests, projects, etc.

At any duration during and/or after the lab session, students are recommended to team up with the Professor and/or the Technical Leader(s) to clarify if there is any confusion related to the items in the lab sheet and/or class materials.

Section 1: Reflection on low end computing

This section is worth 10 points. The points breakdown is provided below:

- Task 1 = 10 points, a maximum of 2 points awarded for each question.

The bit shifts are sometimes considered bitwise operations because they treat a value as a series of bits rather than as a numerical quantity. In these operations, the digits are moved, or shifted, to the left or right. Registers in a computer processor have a fixed width, so some bits will be ”shifted out” of the register at one end, while the same number of bits are ”shifted in” from the other end; the differences between bit shift operators lie in how they determine the values of the shifted-in bits (from wiki). In general bitwise shift, operators are much faster than applying the multiply, and divide operators. The limitation of the bitwise shift operator is that it is complicated to represent a number that is not in the power of two's. The following article expands on the advantages offered by the bitwise shift operators: `https://softwareengineering.stackexchange.com/questions/13798/what-are-the-advantages-of-using-bitwise-operations`

Additional details related to bitwise operators are discussed in the video. To complete this part, it is required to do the following:

- **Task 1:** Watch this 10-minute video, by using the link below:

  `https://www.youtube.com/watch?v=NLKQEOgBAnw`

  After watching the video, create a markdown file and name it as `video-reflection`. In this file, provide a detailed description of the questions presented below:

  1. How is the roll out similar in base 10 and base 2?
  2. What is the difference between logical and arithmetic shifting?
  3. How does the right shift work on a negative number? For example, how does -23 work using logical and arithmetic shift?
  4. What is the rounding procedure discussed in the video? in conjunction with shift operators.
  5. What is the masking procedure discussed in the video?

Section 2: Shift Operator

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

- Task 2 = 20 points
- Task 3 = 20 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 2:** 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 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.
   (e) Store the final output, that is in the decimal form in the (pass by reference) argument res.

3. **Task 3:** 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, 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

Submission Details

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

1. **video-reflection** markdown file
2. updated version of **shift.c** file
3. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is made before the deadline. The honor code policy can be accessed through the course syllabus. Make sure to add the statement ”This work is mine unless otherwise cited.” in all your deliverables such as source code and PDF files.
4. It is recommended to upload a summary file, with the details that you would like the Professor to know while grading the work. For example, it may be a reflection of your experience in the lab by highlighting some of the challenges that you had faced and a brief mention of how you addressed those challenges while implementing this lab. The summary file may also include a brief mention of any details that one should know about executing your program and what to expect during the execution.

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.


Wish you
all the best