

Lab 10: Bit manipulating Problems

CPSC 2310 – Fall 2023

# Due Date: Monday, November 20, 2023 @ Midnight

# Lab Objective

This lab is going to be a hodgepodge of functions or expressions you will implement/write that will give you practice using bit manipulation.

# Introduction

Below I will describe 4 tasks you are to complete. Each of these tasks are designed to give you practice with bit manipulation and conditional expressions. Please read the instructions for each Task.

Below is a set of rules that you must follow only when the instructions indicate you must follow the **Bit-Level Rules**. In other words, when **Bit-Level Rules** is required, any code you write must follow the rules below:

**Assumptions**

Integers are represented in two’s complement form.

Right shifts of signed data are performed arithmetically.

Data type int is ***w*** bits long. For some of the problems, you will be given a specific value for w, but otherwise your code should work as long as w is a multiple of 8. You can use the expression sizeof(int) << 3 to compute the value of w.

**Forbidden**

Conditionals (if or ?:(ternary operation)), loops, switch statements, function calls, and macro invocations.

Division, modulus, and multiplication.

Relative comparison operators (<, >, <=, and >=)

**Allowed operations**

All bit-level and logic operations.

Left and right shifts, but only with shift amounts between 0 and w-1.

Addition and subtraction.

Equality (==) and inequality (!=) test. Some of the problems do not allow these so pay attention to the individual instructions.

Integer constants INT\_MIN AND INT\_MAX

Casting between data type int and unsigned, either explicitly or implicitly

**Task 1.**

**For this problem you must follow the Bit-Level Rules listed above, with the additional restriction that you may not use equality (==) or inequality (!=) tests. If you violate any of these Rules you will receive a 0 on this task.**

You are to write a C expression that evaluates to 1 when the following conditions are true and 0 when false.

A. Any bit of a given int equals 1.

B. Any bit of a given int equals 0.

C. Any bit in the least significant byte of a given int equals 1.

D. Any bit in the most significant byte of a given int equals 0.

I have written a program, called Task1.c, that you will fill in the expression to test your answer. Make sure you test each expression with several values.

Hint:

These exercises require thinking about the logical operator ! in a nontraditional way. Normally we think

of it as logical negation. More generally, it detects whether there is any nonzero bit in a word. In addition,

it gives practice with masking.

**Task2:**

You will implement in Task2.c

You are going to write a function that returns 1 when ran on a machine that uses arithmetic right shifts for data type int and yields 0 otherwise. FYI – I tested the solution on the SoC servers on a cerf15 machine so you can assume that the cerf machines use arithmetic right shifts. I have provided you with starter code called Task2.c You are to implement the isArithmetic() function. You are **not** allowed to change the prototype of the function. You will receive a 0 for this task if you change the prototype.

**Task 3:**

**For this problem you must follow the Bit-Level Rules listed above. If you violate any of these Rules you will receive a 0 on this task.**

You are going to write a function that returns 1 if any of the odd bits in an unsigned int equal one. As an example, the bit pattern 0100 0000 has a one in the 2nd bit position so this would return 0. However, bit pattern 1000 0000 has a 1 in bit position 1 therefore it will return 1.

I have provided you with starter code called Task3.c. You are to implement the isOneOdd function. You are **not** allowed to change the prototype of the function. You will receive a 0 for this task if you change the prototype.

**Task 4:**

For Task 4 you will implement 5 functions in Task4.c. I have given you Task4.h. You will need to implement Task4Driver.c that will test each of the functions you implemented in Task4.c. You are to use the C macro assert to test your functions.

Each of the functions you are going to implement has a short description of what the function is designed to do. It also lists the operations allowed when implementing the function.

Ex. int binAnd(int, int); this function basically implements the “&” bit manipulator for integers. The legal ops represent the only operators you can use to implement the function. This means no other conditional or bit wise operators are allowed to be used.

Unless explicitly listed under legal ops, you **cannot** use any of the following:

Control constructs such as if, do, while, for, switch, etc.

Do not define any additional functions.

Do not call any functions within your function implementation.

Do not do any form of casting.

No arrays, structs, or unions allowed.

With respect to Task4, all of the functions you are implementing, can be implemented using the operations listed under legal ops.

Add header guards to Task4.h file.

You are to add a header to each file with the following information.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\*Your name \*

\*CPSC2310 Lab10 \*

\*UserName: \*

\*Lab Section: \*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

If you do not add a header to each of the files points for each infraction will be deducted. Your files must compile and run in order to receive credit. If any of the files do not compile you will receive a zero for that task. In other words, if Task1.c does not compile you will receive a 0 for Task1, same applies for Task2.c, Task3.c, and Task4.c.

Each task will be graded with additional test cases so please do not write your function to the specific test cases in the mains provided.

# Submission Instructions

Tar all the files with the name <YourusernameLab10.tar.gz>and use handin (http://handin.cs.clemson.edu) to submit the tarred file to the Lab10 folder. It is your responsibility to check your files after you have submitted. This will take time so do not wait until the last minute to submit. If any of your files are corrupt or missing you will receive a 0 on this lab.