Homework Assignment 1: Data Lab

Due – February 17th 2017 Friday 11:59pm

---

1. Introduction

The purpose of this assignment is to become more familiar with bit-level representations of integers and floating point numbers. You'll do this by solving a series of programming "puzzles." Many of these puzzles are quite artificial, but you'll find yourself thinking much more about bits in working your way through them.

2. Programming Rules

This is an individual project and you are required to submit your completed code on Canvas by the due date mentioned above. You are required to only submit the `bits.c` file.

The `bits.c` file contains a skeleton for each of the 15 programming puzzles. Your assignment is to complete each function skeleton using only straight line code for the integer puzzles (i.e., no loops or conditionals) and a limited number of C arithmetic and logical operators. Specifically, you are only allowed to

use the following eight operators:

! ~ & ^ | + << >>

A few of the functions further restrict this list. Also, you are not allowed to use any constants longer than 8 bits. See the comments in `bits.c` for detailed rules and a discussion of the desired coding style.

3. The Puzzles

This section describes the puzzles that you will be solving in `bits.c`.

Bit Manipulations

Table 1 describes a set of functions that manipulate and test sets of bits. The "Rating" field gives the difficulty rating (the number of points) for the puzzle, and the "Max ops" field gives the maximum number of operators you are allowed to use to implement each function. See the comments in `bits.c` for more details on the desired behavior of the functions.

Name	Description	Rating	Max Ops
bitAnd(x,y)	bitAnd(6, 5) = 4 x & y using only | and ~	1	8
copyLSB	copyLSB(5) = 0xFFFFFFFF, copyLSB(6) = 0x00000000 using ! ~ & ^ | + << >>	2	5
bitMask	Examples: bitMask(5,3) = 0x38 Using ! ~ & ^ | + << >>	3	16
bitCount	bitCount(5) = 2, bitCount(7) = 3 Using ! ~ & ^ | + << >>	4	40

Table 1 - Bit-Level Manipulation Functions

Two's Complement

Table 2 describes a set of functions that make use of the two's complement representation of integers. Again, refer to the comments in `bits.c`.

Name	Description	Rating	Max Ops
tmax()	return maximum two's complement integer	1	4
isNonNegative	return 1 if x >= 0, return 0 otherwise	3	6
rempwr2	Compute x%(2^n), for 0 <= n <= 30	3	20
isPower2	returns 1 if x is a power of 2, and 0 otherwise	4	20

Table 2 - Arithmetic Functions

Floating-Point Operations

For this part of the assignment, you will implement some common single-precision floating-point operations. In this section, you are allowed to use standard control structures (conditionals, loops), and you may use both int and unsigned data types, including arbitrary unsigned and integer constants. You may not use any unions, structs, or arrays. Most significantly, you may not use any floating point data types, operations, or constants.

Instead, any floating-point operand will be passed to the function as having type unsigned, and any returned floating-point value will be of type unsigned. Your code should perform the bit manipulations that implement the specified floating point operations.

Functions float_neg and float_abs must handle the full range of possible argument values, including not-a-number (NaN) and infinity. The IEEE standard does not specify precisely how to handle NaN's, and the IA32 behavior is a bit obscure. We will follow a convention that for any function returning a NaN value, it will return the one with bit representation 0x7FC00000.

4. Evaluation

Your score will be computed out of a maximum of 100 points based on the following distribution:

50 Correctness points.

45 Performance points.

5 Style points.

Correctness points. The 15 puzzles you must solve have been given a difficulty rating between 1 and 4, such that their weighted sum totals to 50. We will evaluate your functions using our test program using many input combinations. You will get full credit for a puzzle if it passes all of the tests performed by our tests, and no credit otherwise.

Performance points. Our main concern at this point in the course is that you can get the right answer. However, we want to instill in you a sense of keeping things as short and simple as you can. Furthermore, some of the puzzles can be solved by brute force, but we want you to be more clever. Thus, for each function we've established a maximum number of operators that you are allowed to use for each function. This limit is very generous and is designed only to catch egregiously inefficient solutions.

Style points. Finally, we've reserved 5 points for a subjective evaluation of the style of your solutions and your commenting. Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they need not be extensive.

5. Advice

Don't include the <stdio.h> header file in your `bits.c` file, as it will mess with our test program. You can write your own test script that will test your functions in `bits.c`.

5. Files Provided

`bits.c and bits.h`