# CSED 211, Fall 2021
# Data Lab: Manipulating Bits
# Assigned: Sept. 09, Due: Sept. 16, 11:59PM

Ko Youngjoo (y0108009@postech.ac.kr) is the lead person for this assignment.

## 1 Introduction

The purpose of this assignment is to become more familiar with bit-level representations of integers and floating point numbers. You'll solve five problems in the presentation.

## 2 Logistics

This is an individual project. All handins are electronic. Clarifications and corrections will be posted on the course Web page.

## 3 Handout Instructions

Start by copying `bits.c` to a (protected) directory on a Linux machine in which you plan to do your work. Then give the command

```
unix> cp bits.c ./path/your/directory
```

You will be modifying and turning in is `bits.c`.

The `bits.c` file contains a skeleton for each of the 5 programming puzzles. Your assignment is to complete each function skeleton using only *straightline* 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.

# 4 The Puzzles

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

We have 5 puzzles bitAnd, addOK, isNegative, logicalShift, and bitCount. Using legal operation that we allow you, you need to complete puzzles to execute the desired behavior of the functions.

Please refer to the presentation for detailed instructions.

| Name | Description | Rating |
|---|---|---|
| `bitAnd(x,y)` | `x & y` using only `\|` and ~ | 1 |
| `addOK(x,n)` | Determine if can compute x+y without overflow | 3 |
| `isNegative(x)` | isNegative - return 1 if x ¡ 0, return 0 otherwise | 2 |
| `logicalShift(x,n)` | Shift right logical. | 3 |
| `bitCount(x)` | Count the number of 1's in `x`. | 2 |

Table 1: Bit-Level Manipulation Functions.

bitCount(x) is optional problem. You don't need to solve it, but we will give you bonus points if you solve it. Bonus points only apply for this lap, and if you exceed your total socore for this lap due to the bonus score, it will not affect other laps.

# 5 Evaluation

Your score will be computed out of a maximum of 9 points.

*Correctness points.* We will evaluate your functions . You will get full credit for a puzzle if it passes all of the tests, and no credit otherwise.

# 6 Handin Instructions

Upload your source file `bits.c` and report in plms. You need to explain your answer in the report. The format of file is (student number)_(your name).c / .doc.

# 7 Advice

- Don't include the `<stdio.h>` header file in your `bits.c` file, as it confuses `dlc` and results in some non-intuitive error messages. You will still be able to use `printf` in your `bits.c` file for debugging without including the `<stdio.h>` header, although `gcc` will print a warning that you can ignore.

- The `dlc` program enforces a stricter form of C declarations than is the case for C++ or that is enforced by `gcc`. In particular, any declaration must appear in a block (what you enclose in curly braces) before any statement that is not a declaration. For example, it will complain about the following code:

```
int foo(int x)
{
  int a = x;
  a *= 3;      /* Statement that is not a declaration */
  int b = a;  /* ERROR: Declaration not allowed here */
}
```