Project 2 (Bit manipulation Operations in C)

CSCI 2461

September 2020

1 Objective

In this assignment you will examine a program consisting of a number of functions - labelled badger to shrew – each of which tests your knowledge of bitwise C operations. Your goal is to (a) determine what each function does and (b) implement a 'simpler' equivalent function (if there is one).
Important: Please read these instructions very carefully – your submission **must** follow the specifications in this document. Failure to follow our specifications could lead to a zero on this project.

2 Assignment

2.1 Folder Contents

When you unzip the Project2 folder you will find four files:

- A C file titled Project2.c – this is the file (a C program) that contains the questions and this is the file you will edit and submit.
- A main.c file - this file is included as a sample to run your code and includes an example of calling one of the functions (in Project2.c) and printing the output so you can check for code correctness.
- A tester.h file - you should not change this; it include simple testing code (we will use a comprehensive testing code during grading).
- This document - Project2-Specs.pdf

2.2 Submission Requirements

You will submit two files (on Github): (1) a report in PDF format, and (2) your C code. Details are provided in the next section.

3 Requirements

There are two parts to the Project.

3.1 Part A

You need to determine what each function is doing by examining the code (each function) in Project2.c , *i.e.,* determine what each function is doing WITHOUT running the code. You must provide an explanation for your answer - you will get no credit if you correctly identify the function but provide an incorrect or insufficient/incomplete explanation. Simply providing an example is NOT an explanation - so don't expect any credit if your answers are simply a collection of examples or code comments or the function without an explanation of how you derived your answer. You are welcome to provide formal proofs where appropriate, but use your judgement on whether to pursue a formal proof or not.

Once you identify/describe the function, you should run the code to check your answer. To run the code to test your function, you will need to insert appropriate C code to call the function and print out the result – the next section provides instructions for running the code. Once you are done showing what each function does, you should then answer part b.

You will submit your answers for Part A as a PDF document titled First-Initial_LastName_partA.pdf. For example, Graham Schock will need to submit a file named: G_Schock_partA.pdf

An example of what we expect is shown for Question 0 in the Project2.c file. The answer for Part A for this question could be:

- Ques.0: The function ques0 returns 1 if $x = y$ else returns 0, i.e., it checks if $x = y$. By definition of 2's complement representation, $temp1 = -y$ and therefore $temp2 = x - y$. If $x = y$ then $temp2 = 0$ and $!(0) = 1$ and hence the function returns 1 if $x = y$.

3.2 Part B

In this part, you are required to write a simpler equivalent C code to implement each of the ten functions. You will write your code by editing the C file with the answer (a simpler code, if it exists) after each of the assigned functions in a manner similar to the example shown in the function ques0 (i.e., the answer to function ques_shrew should be named ans_shrew). Your submission for part B will be this C file renamed using the same naming convention First-Initial_Lastname_partB.c, *i.e.,* Graham Schock will submit G_Schock_partB.c

The first thing you should do is rename the Project2.c using this command (substitute your first and last name in the command below).

```
mv Project2-Fall2020.c FirstInit_Lastname_PartB.c
```

IMPORTANT: Your code must compile with the commands provided. If the code does not compile you will receive a 0 for part B. **You should not insert any print statements in file you submit** - you will lose points if you do not follow these specifications!

4 Testing

As mentioned above you will need to test your part B. We have provided the template for a very basic testing harness. The tester checks if the outputs of the question and answer are identical, *i.e.,* if ques0 and ans0 give the same output. To test for equality of other functions you will need to edit main.c. For example, to test function `pony` you will need to insert the appropriate printf statement in main.c and call both `pony` and `ans_pony` with the arguments you use for testing. **You should not insert print statements in the Project2.c file – make sure you have removed all such statements from the file before you submit your code.**

You will need to run the following command to compile your main.c with your partB.c and then to execute the program:

```
gcc main.c First\_Last\_PartB.c -o test
./test
```