**CSE2421 Lab 2 (6 points)**

Bit operations and I/Os

**Step 1: Implement a bit function**

1.1 Download the source code and take a look at it

1.2 In step 1, you do not need to change lab2main.c. You are only required to implement the take_bits function in bit_op.c

unsigned long take_bits(unsigned long number, int startIndex, int length)

This function should take some bits from "number", represented as binary format. These bits should start from position startIndex with the given length. For example, suppose number is 1010010110100101111100001111000…0, startIndex = 1, length = 5, this function should return 000 ….. 01001

Note 1: startIndex starts from 0, which means the left most bit has startIndex = 0.

Note 2: You should move the found bits to the rightmost part of your return value, as shown in the above example.

Note 3: Your function should check the validity of startIndex and length, and print an error if they are not valid. You should try your best to figure out what values are valid and what are not. After printing the error, you can call "exit(1);" to terminate the current process.

Note 4: You should use bit operations to implement this function. DO NOT convert "number" to a string. DO NOT use a loop.

**Step 2: Change your program to take number and arguments from user input**

In the above step, the program takes input from command line arguments. In this step, you should change the lab2main.c so that it takes input from keyboard, like Lab1.

2.1 It should first output "How many numbers do you need to compute?" to ask the user how many numbers s/he needs.

2.2 Then it should output "Please enter number, startIndex, and length.", and parse the user's input into three numbers.

2.3 Then it should perform the bit operation and output the result, same as in Step 1.

2.4. Then it should repeat 2.2-2.3 multiple times, specified by the number in 2.1.

Note 1: You do not need input from command line arguments anymore, so you can remove related code in main.c

Note 2: Use scanf to parse the user's input.

**Step 3: submit your code on Carmen**

Note 1: do not forget your Makefile

Note 2: you only need to submit your code after you finish Step 2, which should include your code from Step 1. No need to submit code for Step 1 separately

**Test cases:**

1234, 56, 5 = 26

1234, 56, 8 = 210

12345678, 32, 16 = 188
12345678, 32, 8 = 0
1234567890876, 32, 8 = 113
1234, 56, 9 (ERROR)
1234, 65, 3 (ERROR)

Grading criteria:

1. We will unzip your zip file, and run “make”. If it reports any errors, you will lose all points
2. We will then run your program. We will check both steps
    a. (2 points) Your program should be able to take inputs from keyboard multiple times.
    b. (3 points) Your program should be able to perform bit operations properly on the test cases shown above and also on some other hidden test cases
3. (1 point) your code should be properly indented and should have proper names for your variables