**CS47: Section 2, Fall 2020**

**HW #1: Calculator (70 Points)**

**Due Date: Oct 13, 2020**

Description:

You are responsible to implement several assembly functions to perform the simple arithmetic calculations for 2 64-bit integers. These functions will use the C function signature but the main logic within this function should be inline assembly code using the ASM block similar to the assembly example shown in class.

Program Specification:

1. long mult ( long op1, long op2 )

* Can’t use the MUL/IMUL instructions, meaning you use ADD repeatedly
* If there are overflow, return the overflowed values in RAX register

1. long XOR ( long op1, long op2 )

* xor will return the result of bit exclusive OR of op1 / op2
* can use XOR instruction

1. long rotate ( long op1, long direction, long number\_of\_bits )

* rotate will perform logical bit-rotation of input operand (op1)
* direction = 0 for left and 1 for right
* number\_of\_bits will dictate how many bits to rotate left or right
* you need to use rcl and rcr assembly instructions

1. long factorial ( long op1 )

* Input a positive integer (>0) and return the result of op1!
* Must use a loop in ASM to compute the result (no recursion)

Program Checklist:

You will submit p1\_64.cpp on canvas and a file named readme.p1 text file with any comments (e.g Windows or MacOS, 64-bit compilers that you use and what work or not work about your code). You MUST be able to compile and run the program from the command line. If I can't test your program, you're not going to get much credit for your work!

For #1, rewrite mult to use loop. For #2 & 3, write a new function using my examples as a good starting point. Also add 2 new printf() statements to print out the results similar to the others. For #4, rewrite the factorial C code with inline assembly code using a loop. Remember to add the “q” suffix for 64-bit to the assembly instructions.

Sample output in this sequence:

Operand 1 = 10 x000000000000000a Operand 2 = 5 x0000000000000005

Add(): 15 x000000000000000f

XOR(): 15 x000000000000000f

Mult(): 50 x0000000000000032

Mod(): 0 x0000000000000000

ShiftL: 320 x0000000000000140

ShiftR: 0 x0000000000000000

RotateL: 320 x0000000000000140

RotateR:-6341068275337658368 xa800000000000000

Fact(): 3628800 x0000000000375f00