CS 240 - Homework 4

Assigned: Monday, 2.27.17	Due: Monday, 3.6.17

Make a subdirectory "hw4" in your cs240 folder for this assignment and copy the files from /courses/cs240/s17/jmcurran/GROUP/hw4.

The objective of this assignment is to get some practice with bitwise operations.
The only operators you may use in this assignment are assignment operators, bitwise operators, and relational operators.

bitPractice.h

This is a header file like the one we used in hw3.

I've added an include guard, which prevents compiler errors that result from including a header file more than once. We will cover preprocessor directives in more detail later.

Note the macro INT_BITS. This creates a constant that contains the number of bits in an int. We know this number is 32 for our system at UMB, but this constant allows our code to run on a different implementation, where int might have a different number of bits. Recall that symbolic constants do not require memory to be allocated, so they can be used in a header file.

You should not edit this file at all.

test.c

I will use my own driver to test your work, so I will not collect this file.

You should use it to test your code to make sure it's producing the output you expect.

I've included sample code for how you might test your implementation of eighth() in the stub.

bitPractice.c

There are four functions for you to implement in this file.

The first three can be implemented in one line of code (though you may use more if you want to).

I've included some hints to get you started.

REMINDER

Your work must be your own. You should never look at classmates' code or allow them to see yours.

Start early and ask me questions if you are confused about what the assignment is asking for.

If you get stuck, ask for help, but show me that you have thought about how to solve the problem first.

DELIVERABLES

The only file you need in your hw4 directory is bitPractice.c.

HINTS

Terms	
msb	Most Significant Bit This is the leftmost digit in a binary number. e.g., 01001010 01010101 11101010 10010101
lsb	Least Significant Bit This is the rightmost digit in a binary number. e.g., 01001010 01010101 11101010 10010101

isEven

What do we know to be true about the lsb of an odd number?

eighth

We covered a bitwise operation that is synonymous with integer division by a certain set of numbers when applied to an unsigned integer.

isNthBitOn

We want to do a bitwise comparison between our input and a number whose binary representation is comprised of all 0s and a 1 in the nth bit.

Think about the value of a number with a 1 in the 0th bit and 0s in every other bit position. How can we manipulate that number to create a number with a 1 at index 3 and 0s everywhere else?

flipLeading0s

There are multiple ways to do this, here is one strategy. You may use another strategy as long as you stick to the rules of the assignment.

The symbolic constant INT_BITS will be useful in this function.

Variables:	
num	our input value
msb	a value we create with a 1 in the msb and 0s in all other bit positions
compare	a comparison value we will build to have 1s in the bit positions of all leading 0s in num

We want to look at the leading bits (the bit positions starting from the msb and going to the right) of num and stop when we see the first 1.

We can do this in a loop. Each time through the loop, we left shift num by an incrementing value and then do a bitwise comparison to msb.

Each time we successfully enter the loop body (i.e., each time the next leading bit is still 0), we put a 1 in the corresponding position of compare. In the above example, this means compare ends up with a value of 1100.

Now all we have to do is another bitwise operation on compare and num and return the result.

Example on a 4 bit data type with input value of 2:
```
num = 0010
msb = 1000

first time through loop, left shift by 0
  compare 0010 and 1000
second time through loop, left shift by 1
  compare 0100 and 1000
third time through loop, left shift by 2
  compare 1000 and 1000
```

Example on a 4 bit data type with input value of 2:

loop iteration	left shifted num	msb	compare after executing loop body
0	0010	1000	1000
1	0100	1000	1100
2	1000	1000	loop has exited