Programming Assignment - Bit Manipulations

This assignment is meant to give you experience using bitwise AND, bitwise OR, and shifting. There are other ways to write these functions that are much longer and far less elegant.

My solution used 3 lines of code for highLow, nine lines of code for countPairs, and thirteen lines of code for printDashOct. If any of your code is longer than 30 lines, you are probably doing something very wrong.

RUNTIME ERRORS: If your code has a runtime error with any of my tests you will receive a zero for the attempt. Make sure to thoroughly test your code before submitting. If you receive a zero, EVEN IF IT LOOKS LIKE YOU PASSED SOME TESTS, it is because a test after the ones you passed caused a runtime error and caused the whole testing system to crash. Since the program crashed, it means no output is generated which means the system cannot determine your grade and you receive a zero.

You DO NOT need a power function. DO NOT WRITE A POWER FUNCTION OR CALCULATE POWERS OF 2 WITH A LOOP OF ANY KIND. Shift will calculate powers of 2 with NO function call and NO loop.

1 << 4 = 2^4

You are not allowed to create any function other than those specifically listed below.

You are ONLY allowed to include stdio.h.

YOU ARE REQUIRED TO USE BITWISE OPERATORS AND SHIFTING FOR THIS ASSIGNMENT. IF YOUR CODE CONTAINS LONG SETS OF NESTED IFS OR SWITCHES OR DOING ANYTHING ELSE TO GET AROUND USING MASKING AND SHIFTING YOU WILL RECEIVE A ZERO FOR THE GRADE.

NOTE THAT YOU CANNOT MAKE A SINGLE SHIFT GREATER THAN 32 FOR AN INTEGER. IF YOUR ALGORITHM IS TO BUILD UP A SHIFT VALUE IN A VARIABLE, THAT VARIABLE CANNOT EXCEED 31.

1. Create a C program file named ***bits.cc***. Create methods named ***highLow, countPairs*****,** and ***printDashOct***. These functions will be described in the steps on the following pages.

2. Function ***highLow*** should take two integers as parameters and returns an integer. Break each argument into 16-bit values and return a single integer which consists of the upper 16 bits of the first argument and the bottom 16 bits of the second argument.

   For example,

   highLow (0x045F3E47, 0xAB001F23); //Returns 0x045F1F23.

   should set the bits 2 through 6 to a 1 and all other bits to zero.

   First argument 0x045F3E45

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	1	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	0	0	0	1	1	1

   Second argument: 0xAB001F23

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	0	0	0	1	1

   The top 16 bits [31:16] are from the first number. The bottom 16 bits [15:0] are from the second.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	1	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	1	0	0	0	1	1

   Your function should return 0x045F1F23. If you print all of these in hex it will be much easier to see what is going on.

   Function highLow can easily be done in 5 or fewer lines of code.

3. Function ***countPairs*** takes a single integer and returns an integer which is the number of adjacent pairs of digits. You are to count pairs (and only pairs) of 1s.
   For example, if the number 0x31C61EDF is passed to countPairs.

   00 11 000111000 11 000011110 11 011111

   Your countPairs function should return 3, since there are 3 sets of pairs of 1s.

   What should each of the following return when passed to countPairs?

   0xC00000000, 0x3, 0x6, 0xCCCCCCCC, 0x80000000, 0x00000000, 0x1, 0xFFFFFFFF, 0x0

4. Function ***printDashOct*** takes an integer as a parameter and prints the integer as octal digits. Each octal digit in the printout should be separated by a single dash with a newline at the end. The printout must be a full 32-bit number (i.e., print leading zeros). Note that the high order digit is only 2 bits.

   For example: printDashOct(0x12345678); //00 010 010 001 101 000 101 011 001 111 000
   should print: 0-2-2-1-5-0-5-3-1-7-0

   What would the following arguments print?
   0xC00000000, 0x12345678, 0x3, 0xF, 0x0

5. Create a file named main.cc with a main method inside. Use this main method to test your bits.cc file. DO NOT PUT A MAIN FUNCTION IN BITS.CC. Make sure to test your functions thoroughly using values other than the ones above. Make sure to check boundary values. That is check your functions for start position and end position the same and check for start or end being 0 and 31.
   There is a main.cc attached to this document.

   DO NOT SUBMIT MAIN.CC TO WEB-CAT.
   DO NOT SUBMIT AN BITS.CC THAT HAS A MAIN MEHOD IN IT.

6. Compile and run the program.
   a. Compile with the following command. Cutting and pasting from the PDF may not work. If you cut and paste the following from the PDF and get errors, try typing the command dirctly.

      ```
      g++ -Werror  -Wall -o bits main.cc bits.cc
      ```

   b. Run by typing the following.

      ```
      ./bits
      ```

7. When you have completed your project, upload bits.cc (DO NOT SUBMIT MAIN.CC. DO NOT SUMBIT BITS.CC WITH A MAIN IN IT) to Web-CAT for grading. You will lose points for excessive uploads so you must thoroughly test your work **BEFORE** uploading.

   Here is a link to Web-CAT for this course. Note that it is NOT the same server you used in CS1440 or CS2440.

SAMPLE MAIN FOLLOWS

```
//The following is a sample main.cc.  Modify it to test your program.
//Make sure to test thoroughly.  All the pieces below must be included.

#include <stdio.h>

//These are called prototypes and they must be here.
//Any function in bits.cc that you also call from main must be listed here.
//For now, make sure you include the following 3 lines in main.cc

int highLow (int, int);
int countPairs (int);
void printDashOct (int);

//This function will run when you run your program from the command line.
int main(void)
{
        int p1 = 0x045F3E47;
        int p2 = 0xAB001F23;
        printf ("highLow(0x%08x, 0x%08x) returned 0x%08X\n", p1, p2, highLow(0x045F3E47, 0xAB001F23));

        p1 = 0xD8000000;
        printf ("countPairs(0x%08x) returned %d\n", p1, countPairs(p1));

        //p1 = 0x12345678;
        p1 = 0xC0000000;
        printf ("printDashOct(0x%08X) printed \n", p1);
        printDashOct (p1);

        return 0;
}
```