# 16.216: ECE Application Programming

Fall 2011

Programming Assignment #3: Practicing Bit Manipulation

Due **Monday, 9/26/11**, 11:59:59 PM

## 1. Introduction

This assignment will give you practice with the C bitwise operators, as well as some basic output formatting. You will enter input values in hexadecimal, as well as the bit positions at which these values are to be modified, and then display the results of these operations in tabular form.

## 2. Deliverables

Submit your source file using our course site in Blackboard; you should be able to access the site through https://continuinged.uml.edu/login/login.cfm.

Ensure your source file name is ***prog3_bits.c***. You should submit only the .c file. Failure to meet this specification will reduce your grade, as described in the program grading guidelines.

## 3. Specifications

**Input:** Your program should prompt the user to enter the following:

- Two unsigned integers, in hexadecimal
  - Remember, hexadecimal values are read using format specifier `%x`.
  - Note that inputs will be interpreted as hexadecimal values regardless of whether you include a leading "`0x`"—`0x15` and `15` will be handled the same.
  - You can specify up to 32 bits, using eight hexadecimal digits.
- A single bit position to be modified, which should be between 0 and 31 (including both 0 and 31).
- The lowest and highest bit positions in a range of bits to be modified; each of these values should be between 0 and 31 (including those two values)

A sample run of the program might produce the following first three lines (user inputs are underlined):

```
Enter two values in hexadecimal: 0xdeadbeef 0x12345678
Enter single bit position to be changed: 5
Enter lowest and highest bits to be changed: 0 15
```

**Output:** Your program should perform the following operations on the two unsigned integer inputs:

- Use the single bit position that was entered on the second input line to set, clear, and flip that single bit in both inputs.
- Use the range of bits that were entered on the third input line to set, clear, and flip all bits in that range in both inputs.

You should then print those outputs using exactly the formatting shown below, including the number of spaces shown in the example output.

The first and last lines shown below are "rulers" that **should not be printed**—their purpose is to show you exactly how many characters are being printed. Each dot represents a single character, each slash indicates that 5 characters have been used, and each number indicates that 10 characters have been used. You can therefore see that each output line should use at most 55 characters.

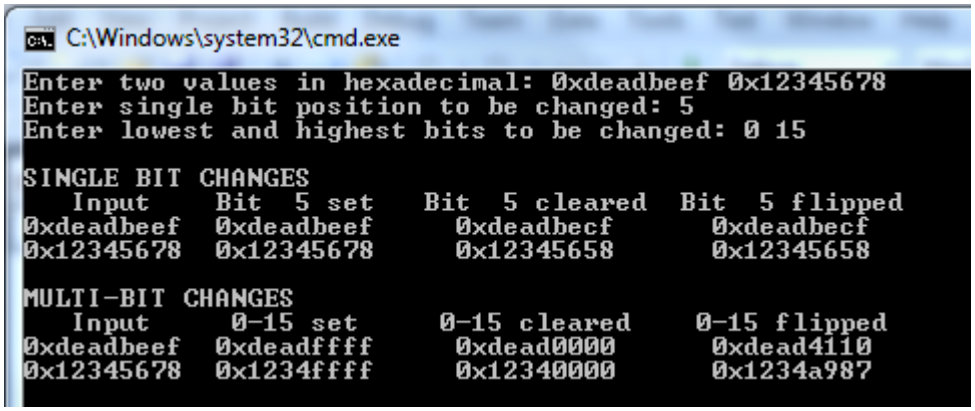
```
..../....1..../....2..../....3..../....4..../....5..../
SINGLE BIT CHANGES
   Input    Bit  5 set   Bit  5 cleared  Bit  5 flipped
0xdeadbeef  0xdeadbeef     0xdeadbecf      0xdeadbecf
0x12345678  0x12345678     0x12345658      0x12345658

MULTI-BIT CHANGES
   Input     0-15 set     0-15 cleared    0-15 flipped
0xdeadbeef  0xdeadffff     0xdead0000      0xdead4110
0x12345678  0x1234ffff     0x12340000      0x1234a987
..../....1..../....2..../....3..../....4..../....5..../
```

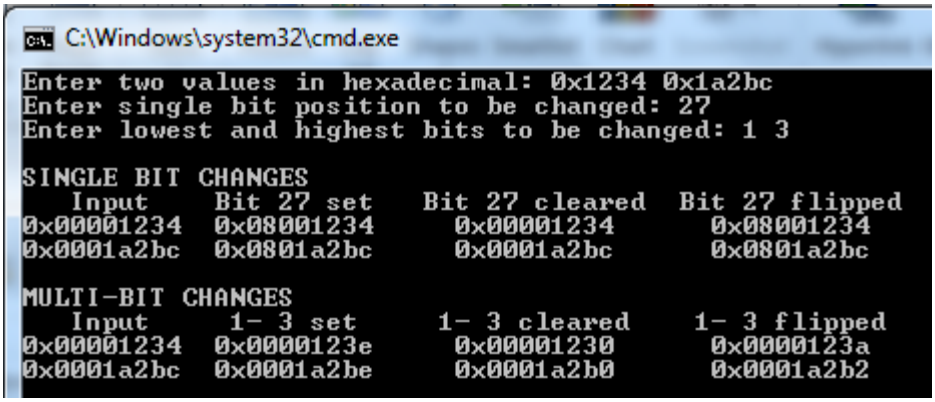
Note: the spacing should be the same regardless of what bit positions are entered. If you don't do any formatting on your output, a one-digit bit position (for example, 5) and a two-digit bit position (for example, 15) will produce differently spaced output.

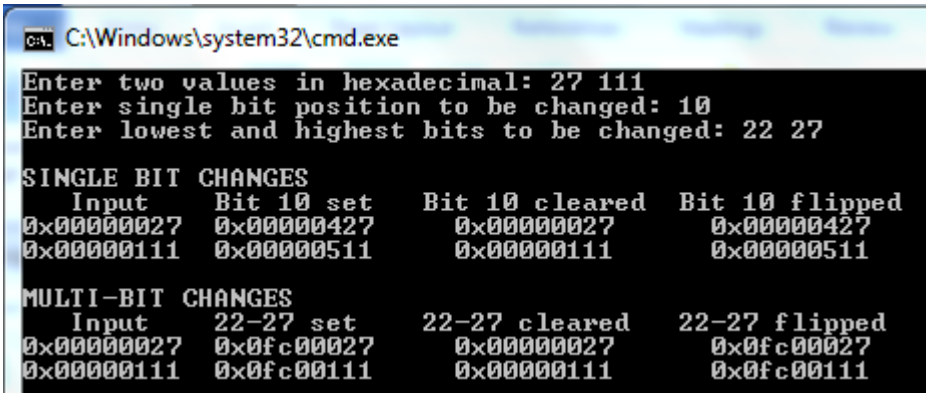
See Section 4: Test Cases for more sample program runs.

## 4. Test Cases

Your output should match these test cases exactly for the given input values. I will use these test cases in grading of your assignment, but will also generate additional cases that will not be publicly available. Note that these test cases may not cover all possible program outcomes. You should create your own tests to help debug your code and ensure proper operation for all possible inputs.




Remember, if you are using Visual Studio, to get your program to terminate with a message saying, “Press any key to continue …”, use the **Start Without Debugging** command (press Ctrl + F5) to run your code.