CMSC 216 Quiz 5 Worksheet

The fifth quiz for the course will be on Thu, Jul 9 (in lab). The following list provides additional information about the quiz:

- The quiz will be a written quiz (no computer).
- Closed book, closed notes quiz.
- Answers must be neat and legible.
- Quiz instructions can be found at http://www.cs.umd.edu/~nelson/classes/utilities/examRules.html
- Make sure you know your section number and your TA’s name.

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TA or instructor during office hours. It is recommended that you try this exercises on paper first (without using the computer).

Exercises

1. Convert $45_{10}$ to binary.
2. Convert $01101101_2$ to decimal.
3. Convert the previous binary number to hexadecimal.
4. Assuming the variables $a$ and $b$ have the following values:
   
   $a \rightarrow 01101101_2$ and $b \rightarrow 10101001_2$

   What is the result of the following operations (provide your answers using hexadecimal numbers).

   $a \& b \rightarrow$
   $a | b \rightarrow$
   $a ^ b \rightarrow$
   $\neg b \rightarrow$
   $a \ll 4 \rightarrow$
   $b \gg 7 \rightarrow$

5. Write the 2’s complement representation of $-17$.

6. Write a function that has as prototype void printBits(unsigned int val). The function prints the parameter value in bit format. For example, the function call printBits(22) will print (for 4-byte unsigned integer):

   0000000000000000000000010110

   For this problem:

   - You cannot assume all unsigned integers are 4-bytes. Use sizeof() to determine the number of bytes associated with an unsigned integer.
   - Your solution must be efficient.
   - You may not use arrays.

7. Implement a bit_and function that returns the bitwise AND of the $a$ and $b$ parameters ($a \& b$). For this function the only allowed operators are $+$ $|$ $\sim$ $^$ $\ll$ $\gg$ and the only allowed constants are $0$ $1$ $2$ $4$. Feel free to use parentheses.

   unsigned int bit_and(unsigned int a, unsigned int b){
8. Implement a **zeros_count** function that returns the number of **zeros** that are present in the first **n** bits (counting from the left) of the **a** parameter. For example, calling **zeros_count(0x86000000, 7)** will return 4. For this function the only allowed operators are 

```
<<  >>  -  ^  |  --  ++  ==  sizeof
```

and any assignment operator. You can use at most one iteration statement and at most one conditional statement. The only allowed constants are 0 1 2 3 4. Feel free to use parentheses.

```c
int zeros_count(unsigned int a, unsigned int first_leftmost_n_bits)
```

9. For this problem, you will write a function called **get_values** that has the prototype below. The function computes the least significant byte and the most significant byte of the **value** parameter. In addition, both values are combined. For this problem:

a. You may not use any loop constructs.

b. You may not use the multiplication operator (* or *=).

c. Your code must work for an unsigned int of any size (4 bytes, 8 bytes, etc.).

d. To combine the values, append the least significant byte to the most significant one.

e. Your implementation should be efficient.

The following driver (and associated output) provides an example of using the function you are expected to write. Notice that in this example an unsigned int is 4 bytes, but your function needs to work with an unsigned int of any size.

**Driver**

```c
int main() {
    unsigned int value = 0xabcdfaec, lsb, msb, combined;
    get_values(value, &lsb, &msb, &combined);
    printf("Value: %x, lsb: %x, msb: %x, combined: %x
\n", value, lsb, msb, combined);
    return 0;
}
```

**Output**

Value: abcdfaec, lsb: ec, msb: ab, combined: abec

```c
void get_values(unsigned int value, unsigned int *p_lsb, unsigned int *p_msb, unsigned int *p_combined)
```