HW 1

This assignment checks your understanding of the number representation basics. It will also give you practice compiling and executing C programs. You can make **hw1.txt** or **hw1.doc** to put your answers and attach it your submission. Also attach **bitcount.c** once you have completed problems 4 and 5.

Background reading

- K&R: Chapters 1-4

Problem 1

What are the decimal values of each of the following binary numbers if you interpret them as **2's complement** integers?

a. 1111 1111 0000 1010

b. 1111 1111 1101 1111

c. 0111 1111 1101 1111

d. 0101 0101 0101 0101

Add your answers in hw1.txt.

Problem 2

Take the same bit patterns from problem 1 and interpret them as the following representations. Provide their value as decimal numbers.

a. Unsigned

b. Sign Magnitude

c. 1's complement

d. Biased

Add your answers to hw1.txt.

Problem 3

For each of the bit lengths and number representations below, list the binary encodings and values of the possible numbers closest to +infinity and to -infinity. (For each of the three parts to this question, you will provide two binary strings and two corresponding decimal numbers.)

a. A nibble (4 bits) using two's compliment.

b. A byte using sign-magnitude.

c. A byte using one's compliment.

Add your answers to hw1.txt

Problem 4

Write a function named `bitCount()` in `bitcount.c` that returns the number of 1-bits in the binary representation of its unsigned integer argument. Remember to fill in the identification information and run the completed program to verify correctness.

```
  /*
    Name:
    Lab section time:
  */

  #include <stdio.h>

  int bitCount (unsigned int n);

  int main ( ) {
    printf ("# 1-bits in base 2 representation of %u = %d, should be 0\n",
      0, bitCount (0));
    printf ("# 1-bits in base 2 representation of %u = %d, should be 1\n",
      1, bitCount (1));
    printf ("# 1-bits in base 2 representation of %u = %d, should be 16\n",
      2863311530u, bitCount (2863311530u));
    printf ("# 1-bits in base 2 representation of %u = %d, should be 1\n",
      536870912, bitCount (536870912));
    printf ("# 1-bits in base 2 representation of %u = %d, should be 32\n",
      4294967295u, bitCount (4294967295u));
    return 0;
  }

  int bitCount (unsigned int n) {
    /* your code here */
  }
```

Problem 5

You have decided that you want your bitcount program above to work from the command-line (see K&R Sec. 5.10), as follows:

```
  # ./bitcount 17
  2
  # ./bitcount 255
  8
  # ./bitcount 10 20
too many arguments!
# ./bitcount
[the same result as from problem 4]
```

You may assume that the single argument will always be an integer in the range from 0 to 2^31-1. You will find the function `atoi` helpful.

Extra for experts: Implement this exercise without using the library function `atoi` (or something comparable).