NUMBER REPRESENTATION

- You can represent everything in bits. Given N bits, you can represent 2^N different things.
- There are five number representations we learned about in 61C. Assume you have N bits.
    - Unsigned
        - Smallest number: 0
        - Largest number: $2^N - 1$
        - You can only represent 0 and positive numbers, no negative numbers
    - Sign and magnitude
        - Smallest number: $-2^{N-1} + 1$
        - Largest number: $2^{N-1} - 1$
        - The most significant bit is the sign bit and the remaining bits are used to represent the magnitude.
            - Positive numbers have a sign bit = 0
            - Negative numbers have a sign bit = 1
        - There is a positive 0 (sign bit = 0, remaining bits are all 0) and negative 0 (sign bit = 1, remaining bits are all 0). Because of this redundancy, we can represent 1 less distinct number i.e. we can only represent $2^N - 1$ distinct numbers instead of the usual 2^N. Which number are we unable to represent now?
            - Because of the "negative" zero, we are able to represent one less negative number i.e. we can only represent up to $-2^{N-1} + 1$ instead of -2^{N-1}.
            - Example: Suppose N = 3.

BIT REPRESENTATION	VALUE
000	+0
001	1
010	2
011	3
100	-0
101	-1
110	-2
111	-3

As can be seen above, we can only represent up to $-2^{N-1} + 1 = -2^{3-1} + 1 = -3$ due to the redundancy of the 0 values.

- One's complement
    - Smallest number: $-2^{N-1} + 1$
    - Largest number: $2^{N-1} - 1$
    - When you negate a number in one's complement, you simply flip the bits.
        - Suppose N = 4. Negating 0b 0000 results in 0b 1111. The prior is the positive 0 and the latter is the negative 0.
    - Remember that positive numbers have leading 0s and negative numbers have leading 1s in one's complement.

- Two's complement
  - Smallest number: -2^{N-1}
  - Largest number: $2^{N-1} - 1$
  - When negating a number in two's complement, you flip the bits and add 1.
    - Examples: Suppose N = 4.
      - 0b 0000 = 0_{10}.
        - Flip → 0b 1111
        - Add 1 → 0b 1111 + 0b 0001 = 0b 1 0000 = 0b 0000
        - Because we have 4-bit numbers, we only look at the four least significant bits and consider the leading 1 to be overflow. Negating 0 does indeed give us 0.
      - 0b 0100 = 4_{10}
        - Flip → 0b 1011
        - Add one → 0b 1011 + 0b 0001 = 0b 1100
        - Using the formula for two's complement, 0b 1100 = $-(1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (0 * 2^0) = -8 + 4 + 0 + 0 = -4$.
      - 0b 1100 = -4_{10}
        - Flip → 0b 0011
        - Add one → 0b 0011 + 0b 0001 = 0b 0100
        - Using the formula for two's complement, 0b 0100 = $-(0 * 2^3) + (1 * 2^2) + (0 * 2^1) + (0 * 2^0) = 0 + 4 + 0 + 0 = 4$.
      - Try out what happens when you try to negate the smallest (most negative) number? When N=4, the smallest number is 0b 1000.
        - You should get back the same number. Why? Because we can represent one more negative number than positive number in two's complement. That means the most negative number won't have a positive counterpart i.e. we can represent [-8, 7] when N = 4.
  - If you want to practice adding numbers in two's complement, here are some practice problems: http://sandbox.mc.edu/~bennet/cs110/tc/add.html. It also outlines the overflow rules and how to detect if it occurred.
- Biased (bias = B)
  - Smallest number: 0 + B
  - Largest number: $2^N - 1 + B$
  - Note that we get the smallest number in biased representation by taking the smallest number in the unsigned representation and adding bias B. Similarly, to get the largest number, we take the largest number in unsigned representation and add bias B.

- Each of the five number representations can represent 2^N things with N bits. Even for sign and magnitude and one's complement, you are technically representing 2^N things. It just so happens that the positive 0 and negative 0 evaluate to the same thing, so in the end you are only representing 2^N-1 distinct values.
- <u>Note: If you add 1 to the largest number in ANY number representation, this results in overflow. This addition will give you the smallest number in that number representation.</u>