1. It is possible to get an overflow error in Two's Complement when adding numbers of opposite signs.

False. Overflow errors only occur when the correct result of the addition falls outside the range of $[-(2_{n-1}); 2_{n-1} - 1]$. Adding numbers of opposite signs will not result in numbers outside of this range

2. If you interpret a N bit Two's complement number as an unsigned number, negative numbers would be smaller than positive numbers.

False. In Two's Complement, the MSB is always 1 for a negative number. This means ALL Two's Complement negative numbers will be larger than the positive numbers.

3. If you interpret an N bit Bias notation number as an unsigned number (assume there are negative numbers for the given bias), negative numbers would be smallerthan positive numbers.

True. In bias notation, we add a bias to the unsigned interpretation to create the value. This means that negative numbers will stay smaller than positive numbers. This is unlike Two's Complement (see description above)

Unsigned Integers

4. If we have an n-digit unsigned numeral $d_{n-1}d_{n-2}\ldots d_0$ in *radix* (or *base*) r, then the value of that numeral is $\sum_{i=0}^{n-1} r^i d_i$, which is just fancy notation to say that instead of a 10's or 100's place we have an r's or r^2's place. For the three radices binary, decimal, and hex, we just let r be 2, 10, and 16, respectively.

Let's try this by hand. Recall that our preferred tool for writing large numbers is the IEC prefixing system（IEC前缀系统）:

Ki (Kibi) = 2^{10}	Gi (Gibi) = 2^{30}	Pi (Pebi) = 2^{50}	Zi (Zebi) = 2^{70}
Mi (Mebi) = 2^{20}	Ti (Tebi) = 2^{40}	Ei (Exbi) = 2^{60}	Yi (Yobi) = 2^{80}

(a) Convert the following numbers from their initial radix into the other two common radices（16进制、10进制和二进制其中一种进制的表示向其他两种进制转换之间转换）:

比如 0b10010011

0b10010011= 147 = 0x93

2. 63 = 0b0011 1111 = 0x3F

3. 0b00100100 = 36 = 0x24

4. 0 = 0b0 = 0x0

5. 39 = 0b0010 0111 = 0x27

6. 437 = 0b0001 1011 0101 = 0x1B5

7. 0x0123 = 0b0000 0001 0010 0011 = 291

(b) Convert the following numbers from hex to

binary:

1. 0xD3AD = 0b1101 0011 1010 1101 = 54189
2. 0xB33F = 0b1011 0011 0011 1111 = 45887
3. 0x7EC4 = 0b0111 1110 1100 0100 = 32452

(b) Write the following numbers using IEC prefixes:

- 举例：
- 2^{16} = 64 Ki
- 2^{34} = 16 Gi
- 2^{27} = 128 Mi
- 2^{61} = 2 Ei
- 2^{36} = 64 Gi
- 2^{59} = 512 Pi

(c) Write the following numbers as powers of 2:

- 2 Ki = 2^{11}
- 256 Pi = 2^{58}
- 512 Ki = 2^{19}
- 64 Gi = 2^{36}
- 16 Mi = 2^{24}
- 128 Ei = 2^{67}

Signed Integers

5 Unsigned binary numbers work for natural numbers, but many calculations use negative numbers as well. To deal with this, a number of different schemes have been used to represent signed numbers, but we will focus on two's complement, as it is the standard solution for representing signed integers.

- Most significant bit has a negative value, all others are positive. So the value of an n-digit two's complement number can be written as $\sum_{i=0}^{n-2} 2^i d_i - 2^{n-1} d_{n-1}$.
- Otherwise exactly the same as unsigned integers.
- A neat trick for flipping the sign of a two's complement number: flip all the bits and add 1.
- Addition is exactly the same as with an unsigned number.
- Only one 0, and it's located at 0b0.

For questions (a) through (c), assume an 8-bit integer and answer each one for the case of an unsigned number, biased number with a bias of -127, and two's complement number. Indicate if it cannot be answered with a specific representation.

(a) What is the largest integer? What is the result of adding one to that number?

Unsigned? 255, 0
Two's Complement? 127, -128

(b) How would you represent the numbers 0, 1, and -1?

1. Unsigned? 0b0000 0000, 0b0000 0001
2. Two's Complement? 0b0000 0000, 0b0000 0001, 0b1111 1111

(c) How would you represent 17 and -17?

1. Unsigned? 0b0001 0001, N/A
2. Two's Complement? 0b0001 0001, 0b1110 1111

A straightforward hand calculation shows that 0b1 . . . 1 + 0b1 = 0.

Arithmetic and Counting

6 Addition and subtraction of binary/hex numbers can be done in a similar fashion as with decimal digits by working right to left and carrying over extra digits to the next place. However, sometimes this may result in an overflow if the number of bits can no longer represent the true sum. Overflow occurs if and only if two numbers with the same sign are added and the result has the opposite sign.

(a) Compute the decimal result of the following arithmetic expressions involving 6-bit Two's Complement numbers as they would be calculated on a computer. Do any of these result in an overflow? Are all these operations possible?

1. 0b011001 − 0b000111

   0b010010 = 18, No overflow.

2. 0b100011 + 0b111010

Adding together we get 0b1011101, however since we are working with 6-bit numbers we truncate the first digit to get 0b011101 = 29. Since we added two negative numbers and ended up with a positive number, this results in an overflow.

3. 0x3B + 0x06

   Converting to binary, we get 0b111011 + 0b000110 = (after truncating) 0b000001 = 1. Despite the extra truncated bit, this is not an overflow as -5 + 6 indeed equals 1!

3. 0xFF − 0xAA

   Trick question! This is not possible, as these hex numbers would need 8 bits to represent and we are working with 6 bit numbers.