ECE 361
Phil Nevins

Question 1

A.

a) Unsigned number range is (0, 2^n –1) so $2^{24} - 1$ = **16777215**
16777215 is the largest positive 24-bit binary unsigned number

b) Two's complement number range is (-2^{n-1}, 2^{n-1} -1) so 2^{24-1} -1 = **8388607**
8388607 is the largest positive 24-bit binary twos complement number

c) Sign/Mag number range is (-2^{n-1} -1, 2^{n-1} -1), so 2^{24-1} -1 = **8388607**
8388607 is the largest positive 24-bit binary sign/mag number

B. How many 10-bit two's complement numbers are:

a) Greater than 0?
For two's complement 10-bit numbers greater than zero there are 2^{N-1} -1 numbers so 2^{10-1} -1 = 511 so there are **511 numbers greater than zero**.

b) Less than 0?
For two's complement 10-bit numbers less than zero there are -2^{N-1} numbers so -2^{10-1} = -512 thus there are **512 numbers less than zero.**

c) Equal to 0?
There is only one 10-bit two's complement number that equals zero.

C. Estimate the value of 2^{20} without using a calculator. Show your work.

$2^{20} = 2^{10} \times 2^{10}$
$2^{10} \approx 1000$
$1000 \times 1000 \approx 1000000$
$2^{20} \approx 1000000$
Using a calculator to check work:
2^{20} = 1,048,576

Question 2

a. Convert the following hexadecimal numbers to unsigned binary. Show your work.

a) 0xB5
B => 1011
5 => 0101
0xB5 => 1011 0101

b) 0x5B
5 => 0101
B => 1011
0x5B => 0101 1011

c) 0xFFFF
F => 1111
0xFFFF => 1111 1111 1111 1111

d) 0xD0000000
D => 1101
0 => 0000

0xD0000000 => 1101 0000 0000 0000 0000 0000 0000 0000

b. Convert the following decimal numbers to 8-bit two's complement numbers or indicate that the decimal number is out of range (i.e., it won't fit in 8 bits). Show your work.

8-bit range is $(-2^{n-1}, 2^{n-1} -1)$, so (-128, 127)

a) 48
48 => 0011 0000
Invert => 1100 1111
Add 1 => **1101 0000**

b) -59
-59 => 1100 0101
Invert => 0011 1010
Add 1 => **0011 1011**

c) 128
OUT OF RANGE

d) -150
OUT OF RANGE

c. Add the following two's complement numbers. Indicate whether the sum overflows an 8-bit two's complement result.

a) 1001 1001
+ 0100 0100
= **1101 1101**
No overflow

c) 1101 0010
+ 1011 0110
=**1** **1000 1000**
According to the rules in the week2_1 pdf, this is carry-out but **no overflow**