# Homework 1

Liane Yanglian  Net ID: xy48

February 2, 2017

## 1 Problem 1

128|64|32|16|8|4|2|1
0| 0| 1| 0| 0 | 1|1|1

So the answer is 00100111.

## 2 Problem 2

First, convert 27 to binary:

128|64|32|16|8|4|2|1
0 | 0| 0 | 1 | 1|0|1|1
I flip the digits:

11100100

Then I add 1, to obtain the final answer:

11100101

## 3 Problem 3

As noted in Problem 1, 39 in binary is 00100111. Then, I normalize it to obtain:

$$1.00111 * 2^5.$$

Since the number is positive, the sign bit is 0. The exponent is 127+5 = 132. I convert 132 into binary:
128|64|32|16|8|4|2|1
1 | 0 |0 | 0 |0|1|0|0
That is 10000100.
The 23-bit mantissa is 00111000000000000000000.

So the answer is 01000010000111000000000000000000.

## 4 Problem 4

I first convert -1.125 to regular binary:

For the integer part, -1, it's just equal to -1 in binary.
For the fractional part, -.125, I multiply it by 2 in the following way:
-.125*2 = -0.250 I get 0 at the first digit to the left of the decimal point, so I keep going;

-.250*2 = -0.500 I get 0 at the first digit to the left of the decimal point, so I keep going;
-0.5*2 = -1 I get (-)1 and there's nothing left, so I stop.
Therefore, reading from top to bottom, I know 1.125 in regular binary is 1.001.
I normalize it to be:

$$1.001 * 2^0.$$

Sign bit is 1 since the number is negative.
Exponent is 0+127 = 127. I convert it to binary:
128|64|32|16|8|4|2|1
0| 1 | 1 | 1 | 1|1|1|1
23-bit Mantissa: 00100000000000000000000.

So the answer is 10111111100100000000000000000000.

# 5 Problem 5

2017:K>Roy in ASCII using hexadecimal format is 323031373a4b3e526f79.

# 6 Problem 6

In a 32-bit computer with 2s complement, the first bit represents the sign, leving me with 31 bits. When all 31 bits have 1 as their digits, I have

$$2^0 + 2^1 + ... + 2^{30} = 2^{31} - 1$$

As a result, any number that's greater than this value cannot be represented, an example would be

$$2^{31}, \quad \text{which is 2147483648.}$$

# 7 Problem 7

a. stack
b. stack
c. heap
d. stack
e. heap

# 8 Problem 8

The value returned by main() is 0. This is because

$$floatc = foo(eptr, bptr, bptr[1]),$$

which is equal to

$$foo(a, [7, 4], 4).$$

Since the pointer is dereferenced, I pick the first value of the second argument, which is 7, so now I have, with a being 21.34:

$$foo(21.34, 7, 4).$$

Since the value of the first argument is greater than that of the other two combined, the foo function returns c = 21.34. Since 21.34 >10.5, the main function then returns 0.

# 9 Problem 9

When I run my program after I type g++ -O0 -o myProgramUnopt prog.c, the run time I got is:
real: 3.546s, user: 3.528s, sys: .004s
When I run my program after I type g++ -O3 -o myProgramOpt prog.c, the run time I got is:
real: 8.557s, user: 8.536s, sys: .003s

So the run time is 3.528s for the unoptimized code and 8.536s for the highly optimized code. I observe that the run time is a lot faster (it takes more than two times the run time of the highly optimized code to run the unoptimized code) with the highly optimized code. This run time difference makes sense because the compiler makes more effort (-O3, not -O0 with my input) into trying to create code that runs faster in the case of the highly optimized code so it naturally optimizes more.