//Name:Shahroz Imtiaz
//Email ID:si6rf
//File Name:floatingpoint.pdf
//Date:9/25/2018

For userid 'si6rf':

```
Part A) Your magic (32 bit) floating point number is -9.78125
This is the number that needs to be converted to (little endian) binary, and expressed in
hexadecimal.

Part B) Your other magic floating point number is, in hex, 0x00c01f40
This is the number that needs to be converted to a (32 bit) floating point number.
Note that the hexadecimal printed above is in little-endian format!
```

Part A)

1. Sign bit: 1 (it's negative)
2. Exponent:
    a. 9.78125 / 2^3 = 1.22265625 (which falls between 1<=x<2)
    b. 3 + 127 = 130
    c. 130 => 10000010 (binary)
3. Mantissa:
    a. 1.22265625 – 1 = 0.22265625
    b. Encode using powers of ½:
        i. 0.22265625 - .5 = 0
        ii. 0.22265625 - .25 = 0
        iii. 0.22265625 - .125 = 1
        iv. 0.09765625 - .0625 = 1
        v. 0.03515625 - .03125 = 1
        vi. 0.00390625 - .015625 = 0
        vii. 0.00390625 - .0078125 = 0
        viii. 0.00390625 - .00390625 = 1
    c. => 00111001000000000000000 (binary)
4. Binary (big-endian) => 11000001 00011100 10000000 00000000
5. Hex (big-endian) => 0xc11c8000
6. Binary (little-endian) =>00000000 10000000 00011100 11000001
7. Hex (little-endian) => 0x00801cc1

Part B)

1. 0x00c01f40 (little-endian) => 0x401fc000 (big-endian)
2. 0x401fc000 (big-endian) = 0100 0000 0001 1111 1100 0000 0000 0000
    a. Sign bit: 0
    b. Exponent: 1000 0000 = 128
        i. Exponent offset = 127
        ii. Real exponent value = 128 – 127 = 1
        iii. Multiply mantissa by $2^1 = 2$
    c. Mantissa: 001 1111 1100 0000 0000 0000
        i. $1 + ((\frac{1}{2})^3 + (\frac{1}{2})^4 + (\frac{1}{2})^5 + (\frac{1}{2})^6 + (\frac{1}{2})^7 + (\frac{1}{2})^8 + (\frac{1}{2})^9) = 1.248046875$
3. 1.248046875 * 2 = 2.49609375
4. Thus, 0x00c01f40 (little-endian) = 2.49609375