1. **Floating Point**

(a) Convert the decimal value -7.5 to 32-bit floating point. Report the result in hex. For full credit, show your work.

(b) Convert the 32-bit floating point value `0xece27100` to its decimal equivalent. For full credit, show your work.

2. **Floating Point (Solution)**

(a) Convert the decimal value -7.5 to 32-bit floating point. Report the result in hex. For full credit, show your work.

32-bit floating point bit breakdown:
Sign=1 bit, Exponent=8 bits, Fraction=23 bits
Exponent Bias=127

-7.5 is negative, so **sign bit=1**

Take the absolute value (7.5) and Divide or multiply by 2 until it is greater than one but less than two. This is getting the number into the binary `1.fraction` expected by the format.

- $7.5/2 = 3.75$
- $3.75/2 = 1.875$
- So 7.5 is 1.875×2^2

(This is equivelant to converting 7.5 to binary, which is 111.1, and noting you have to shift the radix point two to the left to get it to be in `1.fraction` format. The divide/multiply by two in decimal algorithm gives you the same result and can be easier to do especially for large numbers).

Since we are multiplying by 2^2 this means the exponent is 2, but remember we have to adjust by the bias. So the exponent field=2+127 **exponent=129**

Finally we need to calculate the fraction part, 0.875. The textbook descrbes an alogorithm where you repeatedly take the fractional part and muliply by 2 until you hit 1. Note that worse case you might never hit 1 (repeating decimal pattern) and in that case stop after 23 bits for single precision.

- .875*2 = 1.75, integer part is 1 (1)
- .75 *2 = 1.5, integer part is 1 (1)
- .5*2 = 1, integer part is 1 (1)

So fractional part is **111**.
You can verify this by remembering that 0.111 binary is
$2^{-1} + 2^{-2} + 2^{-3} = .5 + .25 + .125 = .875$
Now we need to put the bits in the proper positions

```
Sign          Exponent+Bias              Fraction
(negative)        (129)
1               1000 0001          1110 0000 0000 0000 0000 000

1100 0000 1111 0000 0000 0000 0000 0000
C    0    F    0    0    0    0    0
```

The end result is **0xc0f00000**

(b) Convert the 32-bit floating point value `0xece27100` to its decimal equivalent. For full credit, show your work.

First break things up into binary 32-bits

```
E    C    E    2    7    1    0    0
1110 1100 1110 0010 0111 0001 0000 0000
```

Remember that for 32-bit, Sign=1 bit, Exponent=8 bits, Fractional part is 23 bits.

```
Sign  Exponent      Fraction
1     1101 1001     1100 0100 1110 0010 0000 000
```

A sign bit of 1 means the result will be **negative**

The exponent is 0xd9, or 217 decimal. Remember to subtract off the bias (127), $217 - 127 = 90$ so the **exponent is 90**.

The rest is 1.fraction, so in our case = 1.1100010011100010
The fractional part is $\frac{1}{2} + \frac{1}{4} + \frac{1}{64} + \frac{1}{512} + \frac{1}{1024} + \frac{1}{2048} + \frac{1}{32768}$
= 0.76907349

So the final floating point value is:
$= -1^{signbit} \times 1.fraction \times 2^{exponent-bias}$
$= -1 \times 1.76907349 \times 2^{90}$
$= -2.190 \times 10^{27}$
So the result is approximately $\mathbf{-2.190 * 10^{27}}$