CS 315 Lab 8

First convert the following numbers as shown to IEEE Floating Point Standard (FPS) modified (16 bits) by changing the 23 bit fractional part to a 7 bit fractional part (still using the hidden bit) and showing how they would be stored in a register. Then do the specified operation showing the result as the modified FPS. If you choose to show the hidden bit you **must** put it in parentheses.

```
      29.125     FPS:
+     9.50       FPS:
------------
      +38.625 (expected)
```

```
      29.877     FPS:
-     23.62      FPS:
-----------
      +6.256 (expected)
```

```
############################################################
#           cs315 Week 9
#
#   ->  Solution to lab exercise
#
############################################################
Exercise #1:

        29.125
    +   9.50
    ----------
        +38.625 (expected)

    29|
    14| 1   ^
    7 | 0   ^
    3 | 1   ^   <<<--<<< rewrite from bottom to top
    1 | 1   ^
    0 | 1   ^

    0.125|
    0.250| 0
    0.500| 0    <<<--<<< rewrite from top to bottom
    1.000| 1

    29.125 => 11101.001 = 1.1101001 * 2^4 (131 biased)


    9|
    4| 1    ^
    2| 0    ^
    1| 0    ^   <<<--<<< rewrite from bottom to top
    0| 1    ^

    0.5|
    1.0| 1      <<<--<<< rewrite from top to bottom

    9.50 => 1001.1 = 1.0011 * 2^3 (130 biased)


        0 10000011 (1) 1101001      --> rewrite exponents                                                                  -->      0 10000011 (1) 1101001
    +   0 10000010 (1) 0011000      --> match exponents (match one with smaller exponent to one with larger exponent)  --> +    0 10000011 (0) 1001100
    --------------------------                                                                                          ------------------------

    Now, add mantissas (or hidden bit + fraction) together (sign extend to 10 bits, always):

                                 1  1    1
        (1) 1101001     -->     00 (1) 1101001
    +   (0) 1001100     --> +   00 (0) 1001100
    ---------------         ------------------
                                01 (0) 0110101 * 2^4 = (1) 0011010 * 2^5    <-- normalized result
                                ^
                sign of result is positive but we need to normalize the result by shift right once

    result:     0 10000100 (1) 0011010  => 1.0011010 * 2^5 = 100110.10 = +38.5
                                                              (correct, close enough)

Exercise #2:
        29.877
    -   23.62
    ----------
        +6.256 (expected)
```

```
29|
14| 1   ^
7 | 0   ^
3 | 1   ^   <<<--<<< rewrite from bottom to top
1 | 1   ^
0 | 1   ^

 .877|
1.754| 1
1.504| 1
1.016| 1

29.877 => 11101.111 = 1.1101111 * 2^4 (131 biased)


23|
11| 1
5 | 1
2 | 1
1 | 0
0 | 1

 .620|
1.240| 1
0.480| 0
0.960| 0

23.620 => 10111.100 = 1.0111100 * 2^4 (131)


    0 10000011 (1) 1101111
-   0 10000011 (1) 0111100
---------------------------

Now, add mantissas (or hidden bit + fraction) together (sign extend to 10 bits, always):
    * we are subtracting two numbers, thus, we need to find additive inverse of second one and add it together with the first one.

    00 (1) 0111100
    11 (0) 1000011
+                1
------------------
    11 (0) 1000100
+   00 (1) 1101111
------------------
    00 (0) 0110011  --> 0.0110011 * 2^4 = 1.10011 * 2^2 <-- normalized result
    ^
sign of result is positive but we need to normalize the result


result: 0 - 10000001 (1) 1001100 => 110.011 <--> 6.375 (correct, close enough)
```