```
############################################################
#           cs315 Week 10 - part 2
#
#   ->  Homework 4 solution
#
############################################################
```

*   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.

```
a)      29.125  FPS =       0 - 10000011 (1) 1101001   >-- rewrite exponent -->        0 - 10000011 (1) 1101001
    +   6.50    FPS =   +   0 - 10000001 (1) 1010000   >-- match exponent   -->    +   0 - 10000011 (0) 0110100
    -----------             -------------------------                                  ---------------------------
        35.635 (expected)
                                                                 1  1  1
        (1) 1101001    >-- sign extend to 10 bits -->          00 (1) 1101001
    +   (0) 0110100    >-- sign extend to 10 bits -->  +       00 (0) 0110100
    ---------------                                    ------------------
                                                           01 (0) 0011101 => 10.0011101 * 2^4 = 1.0001110 * 2^5 (biased exponent = 5 + 127 = 132)
                                                            ^
                                       sign of result is positive which was expected (but we need to normalize the result)

    result: 0 - 10000100 (1) 00011101 => 1.0001110 * 2^5 = 100011.10 <-- 35.5 (correct, close enough)


b)      27.749  FPS =       0 - 10000011 (1) 1011101    <--|
    -   25.69   FPS =   +   0 - 10000011 (1) 1001101    <--|
    -----------             -------------------------       |
        2.059 (expected)                                    |
                                    exponents are already matched, so no need to match exponents


        (1) 1011101     >-- sign extend to 10 bits -->      00 (1) 1011101
    -   (1) 1001101     >-- sign extend to 10 bits -->  +   00 (1) 1001101
    ---------------                                    ------------------
                                                           11 (0) 0110010
                                                       +                1
                                                       ------------------
                                                           11 (0) 0110011
                                                       +   00 (1) 1011101
                                                       ------------------
                                                           00 (0) 0010000 => 0.0010000 * 2^4 = 1.0000 * 2^1 (biased exponent = 1 + 127 = 128)
                                                           ^
                                       sign of result is positive which was expected (but we need to normalize the result)

    result: 0 - 10000000 (1) 0000000 => 1.0000 * 2^1 = 10.000 <-- 2.0 (correct, close enough)


c)      - 62.374    FPS =       1 - 10000100 (1) 1111001    >-- match exponent   -->        0 - 10000110 (0) 0111110
    -   147.69      FPS =   +   0 - 10000110 (1) 0010011    >-- rewrite exponent -->    +   0 - 10000110 (1) 0010011
    ------------            ---------------------------                                    ---------------------------
        -210.064 (expected)
```

```
     (0) 0111110        >-- sign extend to 10 bits -->      00 (0) 0111110
-    (1) 0010011        >-- sign extend to 10 bits -->  -   00 (1) 0010011
----------------                                        ------------------

* convert first number to 2's complement:
                                                            11 (1) 1000001
                                                        +               1
                                                        ------------------
                                                            11 (1) 1000010  <--|
* convert first number to 2's complement:                                      |
    (additive inverse as we have subtraction)                                  |
                                                            11 (0) 1101100     |
                                                        +               1      |
                                                        ------------------     |
                                                            11 (0) 1101101  <--|
                                                                               |
                                                            11  1              |
                                                            11 (1) 1000010  <--|
                                                        +   11 (0) 1101101  <--|
                                                        ------------------
                                                            11 (0) 0101111  <-----------------------------------------------|
                                                            ^                                                              |
                          sign of result is negative which was expected (but we need to convert result back to unsigned binary) |
                                                                                                                           |
                                                            11 (0) 0101111  <-----------------------------------------------|
                                                            00 (1) 1010000
                                                        +               1
                                                            --------------
                                                            00 (1) 1010001  <-- result is already in normal form

result: 1 - 10000110 (1) 1010001 => 1.1010001 * 2^7 = 11010001.0 <-- -209 (correct, close enough)



d)      5353.475    FPS =       0 - 10001011 (1) 0100111    >-- rewrite exponent -->        0 - 10001011 (1) 0100111
-   29.875      FPS =   -   0 - 10000011 (1) 1101111    >-- match exponent   -->    -   0 - 10001011 (0) 0000000
------------        ----------------------------                                    ----------------------------
    5323.6 (expected)

    (1) 0100111     >-- sign extend to 10 bits -->      00 (1) 0100111
-   (0) 0000000     >-- sign extend to 10 bits -->  -   00 (0) 0000000
---------------                                     ------------------
                                                        11 (1) 1111111
                                                    +               1
                                                    ------------------
                                                        00 (0) 0000000
                                                    +   00 (1) 0100111
                                                    ------------------
                                                        00 (1) 0100111  <-- result is already in normal form
                                                        ^
                                        sign of result is positive which was expected

result: 0 - 10001011 (1) 0100111 => 1.0100111 * 2^12 = 1010011100000.0 <-- 5344 (correct, close enough)
```