```
##########################################################
#           cs315 Week 2 - part 1
#
#   -> Number systems, unsigned binary, octal, decimal, hex
#
##########################################################

Number systems:
    (base => 2) Unsigned binary or binary representation of positive integers (so no fractional digits)
        * unsigned binary of n-bits has a range of 0 to 2^(n) - 1 inclusive
            Ex: 0000, 1111

    (base => 10) Decimal number system {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
        * Ex: (18, 999

    (base => 8) Octal number system {0, 1, 2, 3, 4, 5, 6, 7}
        * Ex: 12, 77

    (base => 16) Hexadecimal number system {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
        * Ex: 1EF, FFF                                                    ^  ^  ^  ^  ^  ^
                                                                        10  11 12 13 14 15

Conversions between number systems:
    1) decimal to unsigned binary:
        recursively divide the number (or quotient) by 2, and keep the remainder. Do the process until number (or quotient) is zero. Then, rewrite the remainders from bottom to up to get unsigned binary.

        Ex: 19 >-- convert from decimal to unsigned binary -->

            19  |
            9   |   1       --> (9 * 2) + 1 = 19
            4   |   1       --> (4 * 2) + 1 = 9
            2   |   0       --> (2 * 2) + 0 = 4
            1   |   0       --> (1 * 2) + 0 = 2
            0   |   1       --> (0 * 2) + 1 = 1
            ^
        we stop as soon as we get zero. Now, rewrite the remainders from bottom to up to get unsigned binary.

        19 (in base 10) is equal to 10011 (in base 2)


    2) unsigned binary to decimal:
        we follow polynomial expansion procedure to get decimal from unsigned binary.

        Ex. 11101 >-- convert from unsigned binary to decimal -->

            11101 = (1 * 2^0) + (0 * 2^1) + (1 * 2^2) + (1 * 2^3) + (1 * 2^4)
                 => (1 * 1) + (0 * 2) + (1 * 4) + (1 * 8) + (1 * 16)
                 => 1 + 0 + 4 + 8 + 16
                 => 29

        11101 (in base 2) is equal to 29 (in base 10)


    3) decimal to octal:
        recursively divide the number (or quotient) by 8, and keep the remainder. Do the process until number (or quotient) is zero. Then, rewrite the remainders from bottom to up to get octal number.

        Ex. 23 >-- convert from decimal to octal -->

            23  |
            2   |   7       --> (2 * 8) + 7 = 23
            0   |   2       --> (0 * 8) + 2 = 2
            ^
        we stop as soon as we get zero. Now, rewrite the remainders from bottom to up to get octal number.


        23 (in base 10) is equal to 27 (in base 8)


    4) octal to decimal:
        we follow polynomial expansion procedure to get decimal from octal.

        Ex: 130 >-- convert from octal to decimal -->

            130 = (0 * 8^0) + (3 * 8^1) + (1 * 8^2)
              => (0 * 1) + (3 * 8) + (1 * 64)
              => 0 + 24 + 64
              => 88

        130 (in base 8) is equal to 88 (in base 10)


    5) decimal to hex:
        recursively divide the number (or quotient) by 16, and keep the remainder. Do the process until number (or quotient) is zero. Then, rewrite the remainders from bottom to up to get hex number.

        Ex. 58 >-- convert from decimal to hex -->

            58  |
```

```
        3   |   A       --> (3 * 16) + A = 58
        0   |   3       --> (0 * 16) + 3 = 3
        ^
```

we stop as soon as we get zero. Now, rewrite the remainders from bottom to up to get hex number.

58 (in base 10) is equal to 3A (in base 16)

6) hex to decimal:
we follow polynomial expansion procedure to get decimal from hex.

Ex. 12F >-- convert from hex to decimal -->

```
    12F = (F * 16^0) + (2 * 16^1) + (1 * 16^2)
       => (15 * 1) + (2 * 16) + (1 * 256)
       => 15 + 32 + 256
       => 303
```

12F (in base 16) is equal to 303 (in base 10)

7) unsigned binary to octal:
group the number to 3 bits (because 2^3 = 8) starting from right. Find the value of each group in octal. Rewrite the result.

Ex. 1101010 >-- convert from unsigned binary octal -->

```
    (1) (101) (010)
     ^    ^     ^
     1    5     2   => 152
```

1101010 (in base 2) is equal to 152 (in base 8)

8) octal to unsigned binary:
convert each digit to 3 bits unsigned binary representation (because 2^3 = 8) . Rewrite the result.

Ex. 1072 >-- convert from octal to unsigned binary -->

(001) (000) (111) (010) => 001000111010

1072 (in base 8) is equal to 001000111010 (in base 2)

9) unsigned binary to hex
group the number to 4 bits (because 2^4 = 16) starting from right. Find the value of each group in octal. Rewrite the result.

Ex. 111101010 >-- convert from unsigned binary hex -->

```
    (1)  (1110)   (1010)
     ^     ^        ^
     1  14 or E   10 or A   => 1EA
```

111101010 (in base 2) is equal to 1EA (in base 16)

10) hex to unsigned binary:
convert each digit to 4 bits unsigned binary representation (because 2^4 = 16) . Rewrite the result.

Ex. FE01 >-- convert from hex to unsigned binary -->

(1111) (1110) (0000) (0001) => 1111111000000001

FE01 (in base 16) is equal to 1111111000000001 (in base 2)

Note: There is no direct way to convert from octal to hex or vice versa:
We use unsigned binary as an indirect step. Thus, to convert to convert from octal to hex or vice versa:
convert octal (or hex) to unsigned binary first, and then convert unsigned binary to hex (or octal)