```
############################################################
#           cs315 Week 3
#
#   -> Solution to Homework 1
#
############################################################
```

1) How many clock cycles does the following code take.

```
    li  $t2, -128       <-- (Macro instruction; Appendix D).        <<<     -128 is a negative number. Thus, load immediate takes 2 clock cycles.   # $t2 <-- -128
    lw  $t3, 0($t9)     <-- (True assembly instruction; Appendix A) <<<     load word takes 1 clock cycles.                                         # $t3 <-- memory[$t9 + 0]
    mul $t3, $t3, $t2   <-- (Macro instruction; Appendix D).        <<<     multiply takes 33 clock cycles                                          # $t3 <-- $t3 * $t2
    sw  $t3, 0($t9)     <-- (True assembly instruction; Appendix A) <<<     store word takes 1 clock cycles                                         # memory[$t9 + 0] <-- $t3
    ---------------------------------------------
    2 + 1 + 33 + 1 = 37 clock cycles total
```

2) Rewrite the code in question 1 to make it significantly faster.

Note: we can see that multiplication instruction takes the most clock cycles. We can also see that we want to multiply memory[$t9 + 0] with -128 which is -(2^7).
      -(2^7) is a multiple of 2. Therefore, we can use property of shifting (shift left logical 7 times) to achieve the same result with less clock cycles.
      after shift left logical, we should also negate the result to compensate for -(2^7)

```
    li  $t2, -128       <-- (Macro instruction; Appendix D).        <<<     -128 is a negative number. Thus, load immediate takes 2 clock cycles.   # $t2 <-- -128
    lw  $t3, 0($t9)     <-- (True assembly instruction; Appendix A) <<<     load word takes 1 clock cycles.                                         # $t3 <-- memory[$t9 + 0]
    sll $t3, $t3, 7     <-- (True assembly instruction; Appendix A) <<<     shift left logical takes 1 clock cycles                                 # $t3 <-- $t3 * 2^7
    neg $t3, $t3        <-- (Macro instruction; Appendix D).        <<<     negate value takes 1 clock cycles                                       # $t3 <-- $t3 * -1
    sw  $t3, 0($t9)     <-- (True assembly instruction; Appendix A) <<<     store word takes 1 clock cycles                                         # memory[$t9 + 0] <-- $t3
    ---------------------------------------------
    2 + 1 + 1 + 1 + 1 = 6 clock cycles total

    6 clock cycles vs 37. Improvement.
```

3) Convert the following numbers (note these are unsigned binary numbers).

-- Unsigned Binary --	-- Octal --	-- Decimal --	-- Hexadecimal --
0110 0111	0147	103	0x67
1010 0011	0243	163	0xA3
1 0111 1110	0576	382	0x17E
0011 1110	076	62	0x3E
1 0101 1111	0537	351	0x15F
0111 1011	0173	123	0x7B
1011 0011	0263	179	0XB3
100 0111 1011	02173	1147	0X47B

4) Give the unsigned binary representation for the first 5 letters (base 10) and the decimal values of the second 5 letters (unsigned binary):

```
    a. 179          --> 1011 0011
    b. 55           --> 0011 0111
    c. 117          --> 0111 0101
    d. 98           --> 0110 0010
    e. 73           --> 0100 1001
    f. 1001 1101    --> 157
    g. 1101 0101    --> 213
    h. 0111 0111    --> 119
    i. 1011 1101    --> 189
    j. 1001 1101    --> 157
```