```
# Task 1: 2's complement via "fast operations"

from math import *

def int2int8(n):
    a = (n & 255)
    return a

def add(n1,n2):
    a = int2int8(n2 + n1)
    return a

def neg(n):
    a = add(~(n), 1)
    return a

def sub(n1,n2):
    a = (add(n1,neg(n2)))
    return a

def divBy2(n):
    a = (n>>1)
    return a
```

Task 1 of problem 4. These were challenging but fun. A lot of mistakes were made but a lot was learned.

```
In [2]: int2int8(298)
Out[2]: 42

In [3]: int2int8(512)
Out[3]: 0

In [4]: add(5,37)
Out[4]: 42

In [5]: add(1,255)
Out[5]: 0

In [6]: add(77,88)
Out[6]: 165

In [7]: add(121,165)
Out[7]: 30

In [8]: neg(5)
Out[8]: 251

In [9]: neg(42)
Out[9]: 214

In [10]: neg(91)
Out[10]: 165

In [11]: add(121,neg(91))
Out[11]: 30

In [12]: neg( add(neg(42),neg(42)) )
Out[12]: 84

In [13]: sub(42,42)
Out[13]: 0

In [14]: sub(neg(1),100)
Out[14]: 155

In [15]: neg( sub(neg(1),100) )
Out[15]: 101
```

Following along with the instructions and demonstrating that all my outputs match with the HW.

```
# TASK 2:  Extracting binary-digits!

def mod2(n):
    a = sub((divBy2(n+1)), (divBy2(n)))
    return a

def bit2string(b):
    a = str(b)
    return a

def int8ToString(n):
    a = str((mod2(divBy2(divBy2(divBy2(divBy2(divBy2(divBy2(divBy2(n))))))))) + str(((mod2(di
    return a

def displayInt8(n):
    a = bit2string(-neg(n))
    b = int8ToString(n)
    print("val:",(a))
    print("bin:",(b))
    None
```

Task 2. It runs off the page, but it works! Also, my mod function is lame. ((N>>1)<<1) ^N is much better and more fun to work with.

```
In [19]: mod2(42)
Out[19]: 0

In [20]: mod2(21)
Out[20]: 1

In [21]: mod2(255)
Out[21]: 1

In [22]: int8ToString(42)
Out[22]: '00101010'

In [23]: int8ToString(7)
Out[23]: '00000111'

In [24]: int8ToString(neg(128))
Out[24]: '10000000'

In [25]: displayInt8(neg(42))
val: -42
bin: 11010110
```

Some example outputs from the HW