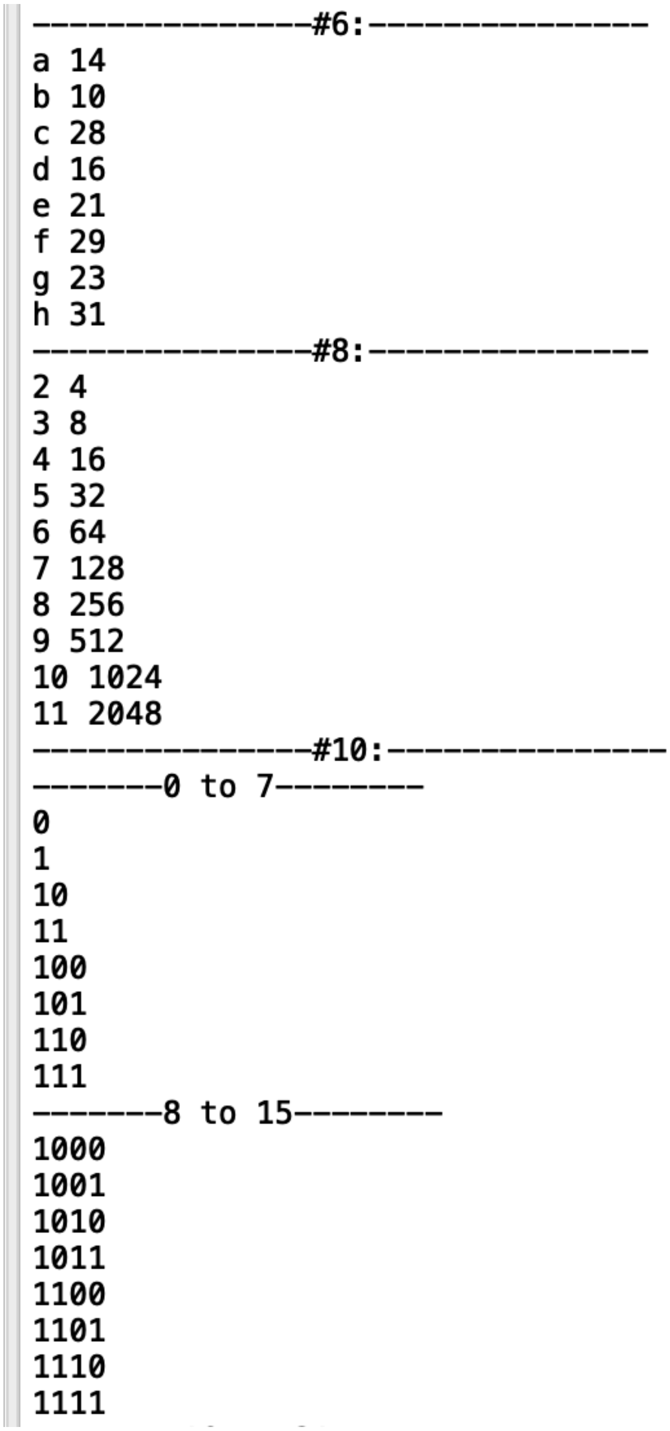
Veiva Piner

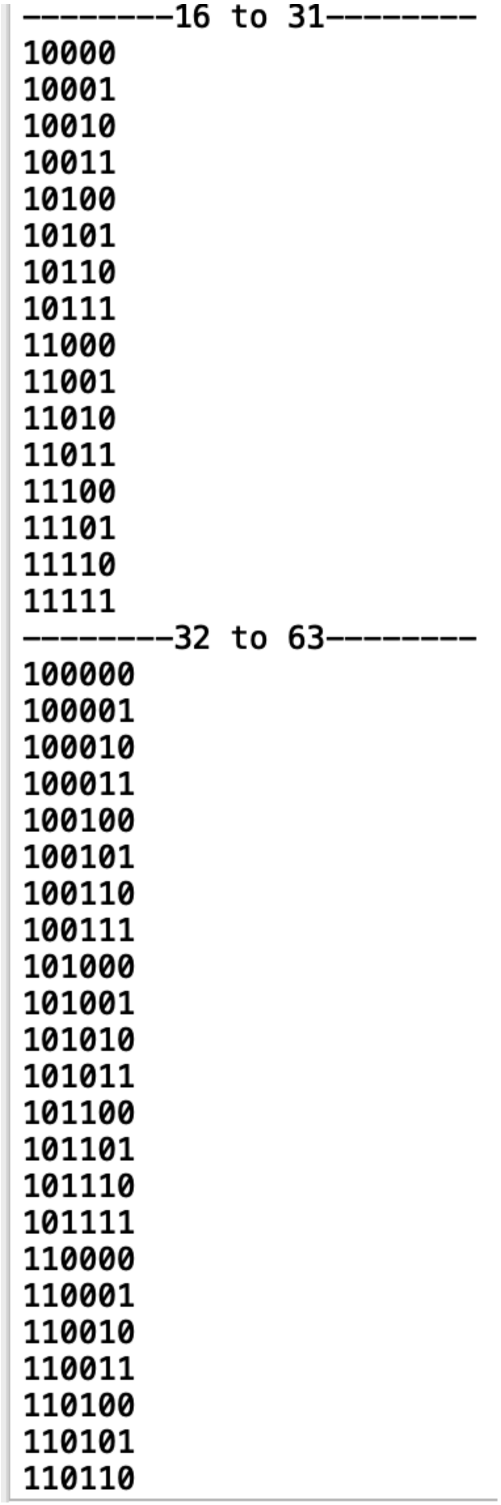
Code:

```
def Dec_to_Bin(num):
    # bin(num) returns binary version of given number
    # .replace("0b","") removes the 0b in the beginning
    # and replaces it with ""
    return bin(num).replace("0b","")

def Bin_to_Dec(num):
    # returns integer version (in base 10) of string input
    # 2 indicates which base the given number is in
    return int(num,2)

def Dec_to_Bin_Div(num):
    binary = ""
    while num !=0:
        r = num % 2
        num = num//2
        binary = str(r)+binary
    return(binary)

def Twos_Comp(n):
    # 1's comp
    if list(n)[0] == '1':
        n = list(n)
        ones_comp = ''
        for i in range (len(n)):
            if n[i] == '0':
                n[i] = '1'
            else:
                n[i] = '0'

        for i in range(len(n)):
            ones_comp+=n[i]

        twos_comp = (int(ones_comp, 2)+int('1', 2))
        twos_comp = -twos_comp
    else:
       twos_comp = Bin_to_Dec(n)

    return twos_comp

#==============================================================================

print("---------------#6:---------------")
print("a", Bin_to_Dec('1110'))
print("b", Bin_to_Dec('1010'))
print("c", Bin_to_Dec('11100'))
print("d", Bin_to_Dec('10000'))
print("e", Bin_to_Dec('10101'))
print("f", Bin_to_Dec('11101'))
print("g", Bin_to_Dec('10111'))
print("h", Bin_to_Dec('11111'))
print("---------------#8:---------------")
for i in range (2, 12) :
    print(i, 2**i)

print("---------------#10:---------------")
for i in range(0, 76) :
    if i == 0:
        print("-------0 to 7--------")
    elif i == 8:
        print("-------8 to 15--------")
    elif i == 16:
        print("--------16 to 31--------")
    elif i == 32:
        print("--------32 to 63--------")
    elif i == 64:
        print("-------64 to 75--------")
    print(Dec_to_Bin(i))

print("---------------#13:---------------")
print("15", Dec_to_Bin_Div(15))
print("21", Dec_to_Bin_Div(21))
print("28", Dec_to_Bin_Div(28))
print("34", Dec_to_Bin_Div(34))
print("40", Dec_to_Bin_Div(40))
print("59", Dec_to_Bin_Div(59))
print("65", Dec_to_Bin_Div(65))
print("73", Dec_to_Bin_Div(73))


print("---------------#15:---------------")
s = int('11',2) + int('01', 2)
print("a DECIMAL:", s , end = '  ')
print("BINARY: ", Dec_to_Bin(s))

s = int('10',2) + int('10', 2)
print("b DECIMAL:", s , end = '  ')
print("BINARY: ", Dec_to_Bin(s))

s = int('101',2) + int('11', 2)
print("c DECIMAL:", s , end = '  ')
print("BINARY: ", Dec_to_Bin(s))

s = int('111',2) + int('110', 2)
print("d DECIMAL:", s , end = ' ')
print("BINARY: ", Dec_to_Bin(s))

s = int('1001',2) + int('101', 2)
print("e DECIMAL:", s , end = ' ')
print("BINARY: ", Dec_to_Bin(s))

s = int('1101',2) + int('1011', 2)
print("f DECIMAL:", s , end = ' ')
print("BINARY: ", Dec_to_Bin(s))

print("---------------#19:---------------")
print("0 in 1's complement form can be represented as all 0's or all 1's:")
print("00000000 or 11111111")
print("---------------#28:---------------")
print(Twos_Comp('10011001'))
print(Twos_Comp('01110100'))
print(Twos_Comp('10111111'))
print("---------------#29:---------------")
print("a: -1.011000 times 2 to the -79th power")
print("b: 1.0101011 times 2 to the 121st power")
```

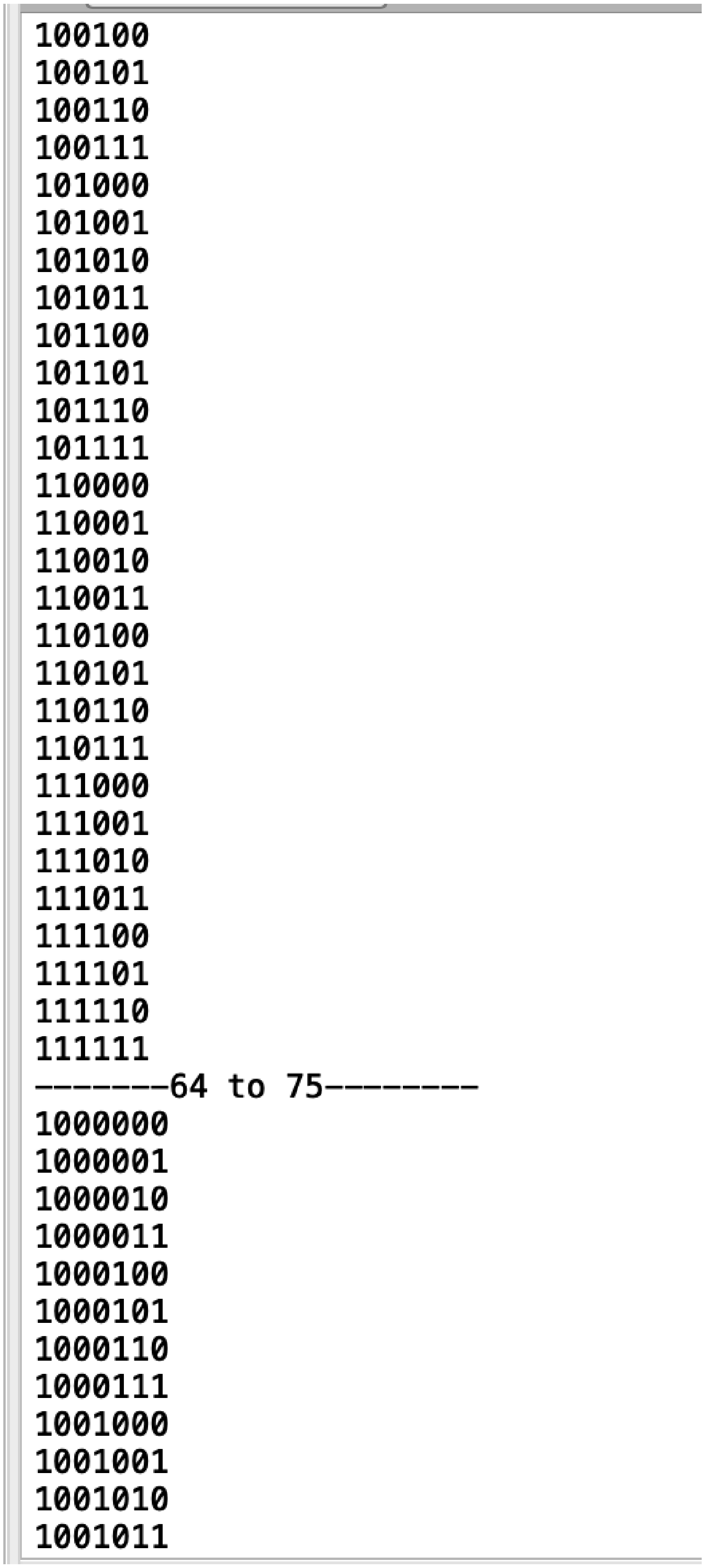
Results:



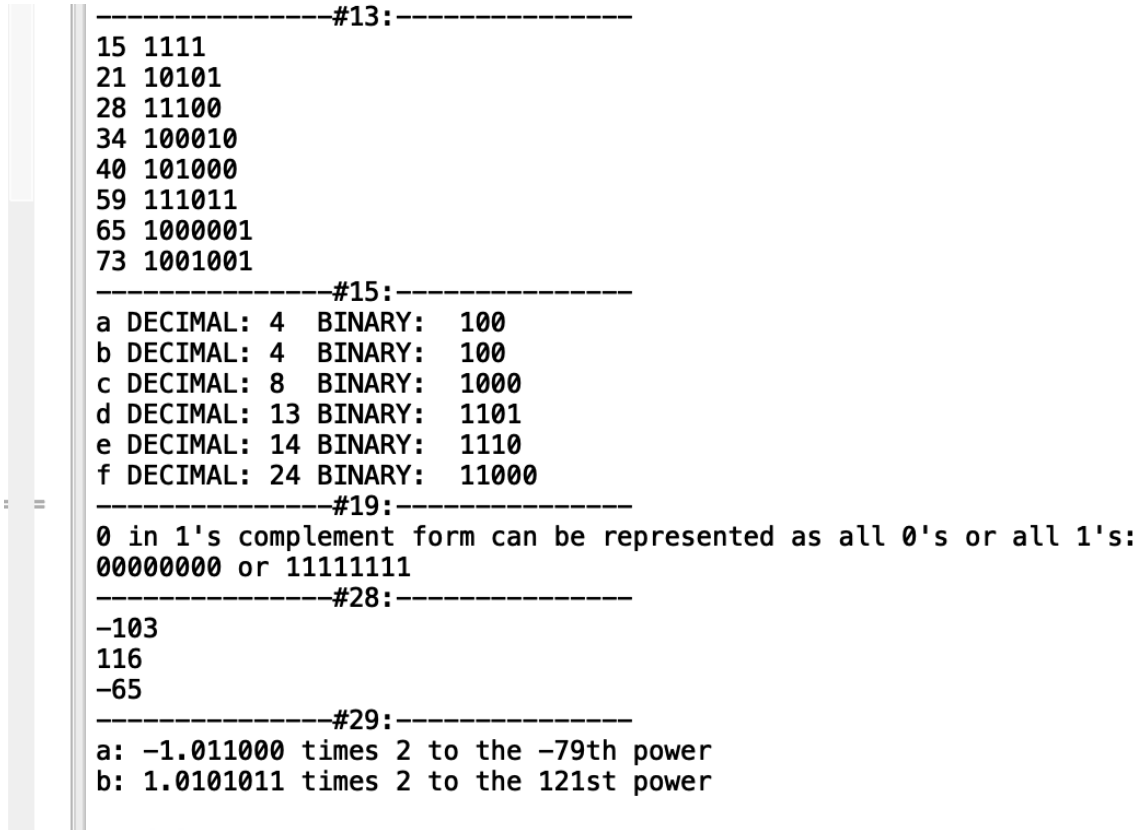
```
---------------#6:---------------
a 14
b 10
c 28
d 16
e 21
f 29
g 23
h 31
---------------#8:---------------
2 4
3 8
4 16
5 32
6 64
7 128
8 256
9 512
10 1024
11 2048
---------------#10:---------------
-------0 to 7--------
0
1
10
11
100
101
110
111
-------8 to 15--------
1000
1001
1010
1011
1100
1101
1110
1111
```



```
--------16 to 31--------
10000
10001
10010
10011
10100
10101
10110
10111
11000
11001
11010
11011
11100
11101
11110
11111
--------32 to 63--------
100000
100001
100010
100011
100100
100101
100110
100111
101000
101001
101010
101011
101100
101101
101110
101111
110000
110001
110010
110011
110100
110101
110110
```



```
100100
100101
100110
100111
101000
101001
101010
101011
101100
101101
101110
101111
110000
110001
110010
110011
110100
110101
110110
110111
111000
111001
111010
111011
111100
111101
111110
111111
-------64 to 75--------
1000000
1000001
1000010
1000011
1000100
1000101
1000110
1000111
1001000
1001001
1001010
1001011
```



```
---------------#13:---------------
15 1111
21 10101
28 11100
34 100010
40 101000
59 111011
65 1000001
73 1001001
---------------#15:---------------
a DECIMAL: 4  BINARY:  100
b DECIMAL: 4  BINARY:  100
c DECIMAL: 8  BINARY:  1000
d DECIMAL: 13 BINARY:  1101
e DECIMAL: 14 BINARY:  1110
f DECIMAL: 24 BINARY:  11000
---------------#19:---------------
0 in 1's complement form can be represented as all 0's or all 1's:
00000000 or 11111111
---------------#28:---------------
-103
116
-65
---------------#29:---------------
a: -1.011000 times 2 to the -79th power
b: 1.0101011 times 2 to the 121st power
```