3a) j2tham

```
def bitject(l):
    if l =="_":
        return 0
    return ord(l)-64

def bitunject(n):
    if n ==0:
        return "_"
    return chr(64+n)

def enc(msg,k="_"):
    encrypted = ""
    for i in range(len(msg)):
        m = i%len(k)
        sum = (bitject(msg[i]) + bitject(k[m]))%27
        letter = bitunject(sum)
        encrypted += letter
    return encrypted

def dec(encrypted,k="_"):
    msg = ""
    for i in range(len(encrypted)):
        m = i%len(k)
        sum = (bitject(encrypted[i]) - bitject(k[m]))%27
        letter = bitunject(sum)
        msg += letter
    return msg
```

3b)

```
#import files
f1 = open("q3_c1.txt")
f2 = open("q3_c2.txt")
c1 = f1.read()
c2 = f2.read()
f1.close()
f2.close()

c1longer=0
xored = ""
if len(c1)>len(c2):
    c1longer = 1
if c1longer:
    xored = dec(c1,c2)
else:
    xored = dec(c2,c1)

k = "LOSE_OUR_HEADS"
for i in range(len(k)):
    msg = dec(xored,k)
    k = "_" + k
    print(msg)
```

Methodology:

1. xored c1 and c2 together to get m1 xor m2

2. decrypted m1 xor m2 with variations of "LOSE_OUR_HEADS" to try to find a portion of m2

Since strings do not have any 14 character streams that make sense, "LOSE_OUR_HEADS" is not inside c1

3c)

```
msg = "HELO_WORLD"
omsg = "HELO_WORLD"
k ="AH"
for i in range(50):
    msg= enc(msg,k)
    if msg == omsg:
        print(i+1)


c = enc(msg,k)
print(0,c)
for i in range(28):
    c = enc(c,c)
    print(i+1,c)
```