Challenge 4

José Costa, 92996

Sabendo e1,e2,c1,c2,N e :

C1 ≡ M^(e1) mod N

C2 ≡ M^(e2) mod N

Como gcd(e1,e2) = 1 podemos aplicar o extended Euclid's algorithm e1*s1 + e2*s2 = 1

Visto que:
C1^(s1) * C2^(s2) ⇔ (M^(e1))^s1 * (M^(e2))^s2 ⇔
M^(e1*s1+e1*s2) ⇔ M^1 ⇔ M

Logo:

M ≡ C1^(s1) * C2^(s2) mod N

Para calcular s1 e s2:
s1 = e1^(-1) mod e2
s2 = (gcd(e1,e2) - e1*s1) / e2

Como o s2 é negativo, é necessário calcular o inverso de C2:
inv_C2 = C2^(-1) mod N

Para obter a mensagem final M:

M = C1^(s1) mod N * (inv_C2^(-s2) mod N)

Por fim foi necessário converter a mensagem para hexadecimal e depois para ascii.

M = "Let's shaft Mallory! He's a lazy and ugly SOAB."

Source Code

```
import sys

def gcdExtended(a, b):
    if a == 0 :
        return b, 0, 1
    gcd, x1, y1 = gcdExtended(b%a, a)
    x = y1 - (b//a) * x1
    y = x1

    return gcd, x, y

def find_a_b(e1,e2):
    a = modinv(e1,e2)
    b = (1-e1*a)/e2
    return a,b

def euclidean_gcd(x,y):
    while(y):
        x, y = y, x %y
    return x

def modinv(a, m):
    g, x, y = gcdExtended(a, m)
    if g != 1:
        return -1
    else:
        return x % m

def findResult(c1,a,b,N,inv):
    return (pow(c1,a,N) * pow(inv,-int(b),N)) % N

def main():
    sys.setrecursionlimit(15000)
    # Parsing data.txt file
    datafile = open("data.txt","r")
    lineList = datafile.readlines()
    for line in lineList:
```

```
        if("N=" in line):
            N = int(line.replace("N=",""))
        elif("e1=" in line):
            e1 = int(line.replace("e1=",""))
        elif("e2=" in line):
            e2 = int(line.replace("e2=",""))
        elif("C1=" in line):
            c1 = int(line.replace("C1=",""))
        elif("C2=" in line):
            c2 = int(line.replace("C2=",""))

    #Start

    a,b = find_a_b(e1,e2)

    inv = modinv(c2,N)

    result = findResult(c1,a,b,N,inv)

    plain_text = str(result)

    hex_array = hex(int(plain_text)).replace("0x","")

    text_inverted = bytes.fromhex(hex_array).decode('UTF-8')
    # Final Result
    print(''.join(reversed(text_inverted)))


if __name__ == "__main__":
    main()
```