Q1) Since the oracle does not give us decrypted version of the C, we need to give it something that we can recover M from the decryptions it gave. I did the following:

$$O(C^2) = (C^2)^d = (M^2)^{ed} = M^2 \; mod \; p$$

When we send C^2 result will be M^2. Then I send C^3 and received M^3. At the last step if I multiply M^3 by M^-2 mod p I will get M mod p which is the plain text.

Following Python code did it and the results are as follows:

Answer : Bravo! You found it. Your secret code is 22545

```
c, N, e = RSA_Oracle_Get()

c2 = (c * c) % N
c3 = (c2 * c) % N

n1 = RSA_Oracle_Query(c2)
n2 = RSA_Oracle_Query(c3)

nInv = modinv(n1,N)
result = (nInv * n2) % N

print(len(bin(result)) % 8)
RSA_Oracle_Checker(binaryToString(bin(result)))
```

Q1.py

Q2) Since the pin code has 10^4 possible value and since the salt can take at most 2^8 different value, key space is very small (around 2.5 million). So we can do a brute force attack to find the PIN number. Q2.py does that operation and results are as follows:

Answer:

PIN NUMBER iS: 1308

The Salt is: 206

```
from RSA_OAEP import *


c = 1556331743614519634596601287095135546751822311026466753718
e = 65537
N = 7342003289123690169505044765550086134382471360514182286688

maxSALT = 2 ** 8
done = False
for PIN in range(0,10000):
    for salt in range(0, maxSALT):
        if c == RSA_OAEP_Enc(PIN,e,N,salt):
            print("PIN NUMBER iS: ", PIN)
            print("The Salt is: ", salt)
            done = True
            break
    if done: break
print("DONE")
```

Q3) The problem with the implementation is that, at the encryption it does not select random number from a large range but only choose values from 0 to 2^16. That makes there are around 65.000 possible k value. By implementing a brute force attack on k, we can retrieve the k value because generator is a public variable and the space we need to try is very small. If we know the k we know that $t * \beta^{-k} = m$. So, I implemented that solution in the Q3.py and the results are as follows:

Answer: Be yourself, everyone else is already taken.

```
# Since the random number generated by using very small range, we can find k by implementing brute force
max_range = 2 ** 16 - 1
k = 0
for i in range(1, max_range):
    if binaryLeftRight(g,i,p) == r:
        # Find the k value
        k = i
        break
#Retrieve message by simple mathematical manipulations.
key = modinv(binaryLeftRight(h,k,p),p)
plaintext = (t * key) % p
print(binaryToString(bin(plaintext)))

```

Q4) Since the r1 and r2 are equal, that implies k1 and k2 are equal. By knowing that we can do the following math:

$$\frac{t_2}{t_1} = (m_2 * B^k * r^k)/(m_1 * B^k * r^k) = \frac{m_2}{m_1} \; mod \; p$$

Basically if we divide t2 by t1 in mod p we will get m1 * inverse of m2 mod p. If we multiply that by m1 we will obtain the m2. These calculations are done in q4.py and the answer is as follows:

Answer: m2 = “A person can change, at the moment when the person wishes to change.”

```python
q = 1445431254694174381649371259143791311198736690037
p = 1372481214340454362479807389530594124163672516191671729652250604396383263125
g = 1272236419218501099095442498814490099446486890402863495267121840789217026026
m1 = b'Believe in the heart of the cards.'
r1 = 98112636909089823473886804230734608783665151359820285384385184926586779318832
t1 = 76506200278870980622832162087706397184942731175881073072279653879125374026784

m2 = ""
r2 = 98112636909089823473886804230734608783665151359820285384385184926586779318832
t2 = 95801086901355834240081662719865802187550109851113545620170852280638597493801

binary_representationM1 = int.from_bytes(m1, byteorder='big')
invT1 = modinv(t1,p)
result = (t2 * invT1) % p
result = (result * binary_representationM1) % p
print(binaryToString(bin(result)))
```

Q5) If we know the relation between k1 and k2 we can exploit that relation to find a. We know the following 2 equalities by implementation of DSA.

$$s_1 = \frac{(h_1 + ar_1)}{k_1} \; mod \; q$$

$$s_2 = \frac{(h_2 + ar_2)}{3k_1} \; mod \; q$$

That means

$$\frac{s_1}{s_2} = 3 * \frac{h_1 + ar_1}{h_2 + ar_2} \; mod \; q$$

$$a = \frac{3s_2h_1 - s_1h_2}{s_1r_2 - 3s_2r_1} \; mod \; q$$

Since all the values are known we can obtain the a from this calculation. Q5.py will solve this issue.

Answer: 224768882479056124130979539634536705233906181169471385891036522645