**Q1)** My n = 101 and t = 25

1. Number of elements in the group is equal to n -1, **which is 100**. (All numbers from 1 to 100)
2. To find a generator I used the following python program (Q1.py). Since 101 is a prime, Euler totient function says there should be phi(100) generator for this group. Phi(100) = 40. Let’s find all of them by following code:

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

All the generators can be listed as follows :

[2, 3, 7, 8, 11, 12, 15, 18, 26, 27, 28, 29, 34, 35, 38, 40, 42, 46, 48, 50, 51, 53, 55, 59, 61, 63, 66, 67, 72, 73, 74, 75, 83, 86, 89, 90, 93, 94, 98, 99]

**Since question wants only one of them, I selected 2 as the generator.**

1. Let’s find a subgroup in the order of 25 and its possible generators. That’s actually a very simple process since we have the previous piece of code. Only thing we need to modify is the if conditional at line 13. This time we will look for some residue class with t = 25 elements.

ekran görüntüsü, metin, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Above, I have a slightly modified version of the previous code, we pass the order of the subgroup as parameter and see the result on the terminal. For order t = 25 the following set of generators will give us subgroups with 25 elements:

[5, 16, 19, 24, 25, 31, 37, 52, 54, 56, 58, 68, 71, 78, 79, 80, 81, 88, 92, 97]

**Since the question asks for only one of them, I selected 97 as the generator of the subgroup with 25 elements.**

**Q2)** I used the following code (Q2.py to solve the problem

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

I assumed p and q as prime numbers. By knowing that, phi(n) = phi(p) \* phi(q) = (p-1) \* (q – 1) I calculated phiN. To decide multiplicative inverse of e in modulo phiN, I used egcd from hw1\_helper to get Bezout coefficients. At the end, I calculated the modular exponent by implementing binary left to right algorithm. When I converted the result into binary, I got the following binary:

“01001001001000000111010001101000011010010110111001101011001000000100100100100000011010000110000101110110011001010010000000110110001100010011011000100000011101010110111001110010011001010110000101100100001000000110010100101101011011010110000101101001011011000111001100101110001000000100100101110011001000000111010001101000011000010111010000100000011000010010000001101100011011110111010000111111”

I putted this into an online converter and got the result:

**“I think I have 616 unread e-mails. Is that a lot?”**

**Q3)** When I used Q3.py to decrypt messages I received error messages for first 2 messages but the third one decrypted without any error. As:

**‘An expert is a person who has made all the mistakes that can be made in a very narrow field.'**

Since the encoding being done by the same nonce and key, we can find the right nonce here. First 8 bytes are the nounce which are: **”\xccU\xe0N\x0e\xb6^1”**

Now, we can fix the nonces of first two accordingly,

To fix first one:

* Our nonce is 1U\xe0N\xb6
* We should add **^1** to end of this nonce
* We should delete **‘N’** from xeON
* We should remove ‘**1U’** part and add **”\xccU\xe0N”**
* After these changes we can decryt the message which is:

**‘The first principle is that you must not fool yourself and you are the easiest person to fool.'**

To fix the second one:

* Nonce of the message two = **“'N\x0e\xb6^\xccU\xe0”**
* The places of them mixed we should put \xccU\xe0 to beginning. **“'\xccU\xe0\x0e\xb6^” ->** Now we need to a 1 at the end of the nonce.
* After these two changes we can decode successfully and receiving the message: **'Somewhere, something incredible is waiting to be known.'**

**Q4)** Q4 Solved by the following algorithm which is an implementation of the algorithm in the lecture slides. Resulting output is as follows: **metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturulduAnswer for part 1 :**

There is only 1 solution =

1115636343148004398322135138661008357945126147114770093414826

**-----------------------------------------------------------------------------------**

**Answer for part 2 :**

No valid answer!

**-----------------------------------------------------------------------------------**

**Answer for part 3 :**

**There are 2 solutions:**

**1 - 1840451085636978827079830514312022149966941191143010614385900**

**2 - 4573017168579321153146925263568627765759266852067838063135011**

**-----------------------------------------------------------------------------------**

**Answer for part 4 :**

**There are 4 solutions:**

1. **120574576795431477647425259344685590574672051332591719355582**
2. **1692041454071987051397898895041599936536312796085130907016143**
3. **3263508331348542625148372530738514282497953540837670094676704**
4. **4834975208625098198898846166435428628459594285590209282337265**

**Q5)** To find the periods of the polynomials we can simulate corresponding LFSRs by using the given lfsr.py. I just manipulated the example usage part of the code according to the given polynomials **(Q5.py)**.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Results are as follows:**

**Period of polynomial 1:** 127 (Maximum Period)

**Period of polynomial 2:** 21

**Period of polynomial 3:** 31 (Maximum Period)

**Q6)** To decide this algorithm we need to apply BM algorithm. Luckly, we have it in lfsr.py. When we run the following small script we obtain the following numbers. (Q6.py) metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Smallest lfsr to produce the streams accordingly as follows: 36 43 31**

**Length of the streams accordingly: 75 80 90**

We know that expected length of lfsr must be n/2 + 2/9. When we imply the formula to lengths of stream we get the expected lengths as follows: 37.7 40.2 45.2. If these numbers are close to the actual length of the smallest lfsr, we can say that it is random, however if it is far small from it is not random.

* **We can deduce that stream 1 and 2 seemed to us random but third one is not. Since 37.7 and 40.2 are very close to 36 and 43 but 31 is far away from the 45.2**

**Q7)**

* First, we need to find the correct format of the instructor name. (After a few billion of hours and mistake I found the correct combination but here let’s assume we hit the correct combination at the beginning). Let’s say our message is from Erkay Savas and he signed the message by the string “Erkay Savas”.
* Since his name contains 11 character the last 77 byte of the plaintext must contain his name. (11 x 7 =77)
* When we XOR the ctext’s last 77 character by “Erkay Savas” in binary. We receive the last 77 character of the key stream.
* Now, we can use this part of the keystream to decide the LFSR. By using BM algorithm, we see that smallest LFSR corresponds to a 27th degree polynomial which is:
* Now, we know that every 27 bit is used to produce another bit in keystream. However, At this point we need to think reversely.
* Let’s say we have 27 bit part of the stream (Let’s call it calculation block, CB). We can set the following algebra:
  + The 27th bit (let’s call it ans) is a result of the combination of first 26 bits with an unknown bit (let’s say c)
  + Our LFSR, XOR the first 24th 25th and 27th bits of it to produce the ans
  + C corresponds to 27th bit, so we can set the following equation:
  + At this point we can calculate left hand side’s first 3 XOR. Let’s call this part as the mid (short hand for middle part)
  + The calculation becomes:
  + Since we know the mid and ans values all the time we can easly produce the C at every round. Which is basically:
  + We found the previous byte. We have to repeat this until the keystream’s length is equal to the ctext.
* After obtained the full keystream, we can xor it by ctext and can obtain the answer, which is:

**Dear Student,**

**Outstanding job on tackling this challenging problem!**

**Congratulations!**

**Best, Erkay Savas**

**The implementation can be found in Q7.py**

**metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**