1. a. We are asked to find a generator of the group Z_P^* , there are p-1 elements in this group. We know that primitive elements are generators of the group, so if the primitive element of a group is x then x^{p-1} should be 1 and no other x^n should be 1. So in the function, I keep track of every element's power mod(p). I have tried all the elements of the group such that if any power of the element is equal to 1 and power is not p-1, the function continues with next element and when we find an element with x^{p-1} = 1 I double checked if it generated all the elements, if it generated then this is one of the primitive elements and quit the function. **The number sent to me was p = 541 and one of the generators of the group is 2.**

   b. t that was sent to me was 27, to find the subgroup with order 27 and the generator of the subgroup, for all the elements of the group, I have find the powers of each element by incrementing the power by 1 each time and add them to a subgroup list until it generated an element which was already in the subgroup list. Then I have checked the number of elements in the subgroup, if the length is 27 then I have found the subgroup and it's generator element. **The subgroup with order 27 was [28, 243, 312, 80, 76, 505, 74, 449, 129, 366, 510, 214, 41, 66, 225, 349, 34, 411, 147, 329, 15, 420, 399, 352, 118, 58, 1] and generator of that subgroup is 28.**

2. For the second question, I first found phi(n), which is equal to (p-1)*(q-1) since n = p * q. Then using the modinv function I have found d because d was equal to modular inverse of e mod phi(n). Then I have implemented Binary Right to Left function from the slides to find m = c^d mod n because both c, d and n are big numbers. Then found the m which was 561395015603525619642719231840467292642446911062474466341687 and when I decode it into Unicode string the message was 'Your secret number is 937'.

3. To solve the congruences of the form ax $\equiv$ b (mod n) I have created a list of solutions and find the gcd of a and n, gcd(a,n) = d. If d is 1 then there is only one solution and x is equal to $(a^{-1}$ mod n * b) mod n. If gcd is not 1 but divides b then there are d solutions, find a/d, b/d and n/d and this time x' = ($\frac{a}{d}^{-1}$ mod (n/d) * b/d) mod n/d . Then all the solutions to the congruence is x = [x', x' + n/d, x' + 2n/d, … , x + (d-1)*n/d]. If d is not 1 nor does not divide b then there are no solutions.

   n = 97289040915427312142046186233204893375
   a = 61459853434867593821323745103091100940
   b = 22119567361435062372463814709890918083
   NO SOLUTIONS.

   n = 97289040915427312142046186233204893375
   a = 87467942514366097632147785951765210855
   b = 32916824542066686459328799486938256 40

There are 5 solutions to that equation:

[113687137494180043727898218252158652 18,
308265219325034668011990590718568438 93, 502843301155889292296082963184978225 68,
697421382986743916580175335651388012 43,
891999464817598540864267708117797799 18]

n = 97289040915427312142046186233204893375
a = 74945727802091171826938590498744274413
b = 54949907590247169540755431623509626593
There is one solution:

[75940790615126559855606958795348491611]

4. I have used the functions in lfsr.py, generated a keystream and checked if the first period is equal to $2^L - 1$.

   $P_1(x) = x^7 + x^3 + x^2 + 1$

   First period of the keystream generated with this polynomial is 62 while $2^L - 1 = 127$ since L = 7. So **this polynomial does not generate maximum period sequences.**

   $P_2(x) = x^7 + x + 1$

   First period of keystream generated with this polynomial is 127 which is equal to $2^L - 1 = 127$ since L = 7. So **this polynomial generates maximum period sequences.**

5. I have used the functions in lsfr.py to check if the sequences are predictable or not. To do that I have used BM function, this function returns the linear complexity of the polynomial that generates the sequence.

   For all sequences linear complexity is 37 and the polynomial that generated these sequences are the same. Linear complexity of the sequences is the shortest LFSR that can generate this sequence and BM function found it so these sequences are predictable. Also the expected linear complexity of the random sequences of length n is n/2. All three sequences are length 150 and 37 is far more less than 150/2 = 75. So these sequences are not random, **predictable sequences.**

6. Bonus

   To find the plaintext of this message first we know that it starts with ‘Dear Student’. I first

converted it into binary using ASCII2bin function. Then when we XOR the binary representation of 'Dear Student' with the first 84 bits of the key we get the first 84 bits of the cipher text. Then to find the first 84 bits of the key, I XORed the first 84 bits of cipher text with the binary representation of the 'Dear Student'.

Now I know some part of the key, I have sent it to Berlekamp Massey algorithm to find the linear complexity and the connection polynomial. L = 27 and connection polynomial = $1 + x + x^2 + x^5 + x^{27}$.

Then to find the rest of the key stream I need to know the initial state of that LFSR. To do so I have got the first 27 elements of the key that we know and reversed it because the last element of the initial state left first and first element of the initial state left last, so the initial state is the reversed of the first 27 bits of the key. Since we know the initial state and connection polynomial we can use LFSR function from lfsr.py to find the rest of the key.

After I found the key, I XORed the key with cipher text and found the binary representation of all the plaintext. Then I put the plaintext to bin2ASCII function and got the string version of the plaintext which says:

Dear Student,
You have worked hard, I know taht; but it paid off:)
You have just earned 20 bonus points. Congrats!
Best,
Erkay Savas