1. For both of the questions I have used BitVector class and implemented solutions with BitVector.
{'a': '11110100', 'b': '10000100'}
a = $x^7 + x^6 + x^5 + x^4 + x^2$
b = $x^7 + x^2$

a. c(x) = a(x)×b(x) and I found c(x) as $x^6 + x^5 + x^3 + x^2 + x + 1$ and bit representation of c is 01101111.
b. Multiplicative inverse of a in GF(2^8) is $x^6 + x^5 + x^3$ and the bit representation of a inverse is 01101000.

2. I've implemented the answer in 411_hw3_question2 using the correlation attack and as a result I have found the initial states as:

LFSR1 = [1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1]
LFSR2 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0]
LFSR3 = [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]

3.

a. Linear Complexity = 79*85 + 79*97 + 85*97 + 79*85*97 = 673978
Period of output sequence =
$(2^{79} - 1) * (2^{85} - 1) + (2^{79} - 1) * (2^{97} - 1) + (2^{85} - 1) * (2^{97} - 1) + (2^{79} - 1) * (2^{85} - 1) * (2^{95} - 1)$
b. Nonlinearity degree is 3 since we are and'ing x1, x2, x3.
To find the balance and correlation I have constructed the truth table

x1	x2	x3	Output sequence
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

According to the table the output of the function is not balanced because there are five 0's and three 1's. There are more 0's in the output sequence according to the truth table, then the

output sequence is going to tend to have more 0's than 1's and it will be biased and it loses good statistical properties of LFSR sequences.
The correlation between x1 and output sequence is 5/8, the correlation between x2 and output sequence is 5/8 , the correlation between x3 and output sequence is 5/8. Then x1, x2, x3 are all correlated to output sequence because the number of same bits between x1 and output sequence, x2 and output sequence and x3 and output sequence are not ½ but it's greater than ½ so these will give advantage to the attacker.

If we consider the function in terms of nonlinearity degree, balance and correlation, this function is not a good combiner function because output sequence is not balanced and there are correlation between x1, x2, x3 and output sequence.

4. In AES if we don't have ShiftRow and MixColumn layers then we will only have a Key Addition layer at the beginning. Then in round 1 to $n_r - 1$ we will apply Byte Substitution layer and Key Addition layer. In round n we will apply only Byte Substitution layer and we will get the ciphertext.

In the key addition layer before we start the rounds we XOR the plaintext, which is 128-bit, with the key which is 128-bit too. Then the resulting 128 bit which is 16 bytes will be arranged into a 4x4 matrix, so each element of the matrix will consist of 8 bits which is 1 byte. Then we will take each byte and change it to another byte in Byte Substitution layer using a table with 256 entries. Then at each round we will add the round key and continue with the other round. Since we don't have ShiftRow and MixColumn layers then every byte in the plaintext will affect only one, corresponding byte in the ciphertext. So without these 2 layers we won't have diffusion, each bit in ciphertext will not depend on each bit in plaintext anymore, instead each 8-bit in ciphertext will depend on corresponding 8-bit in plaintext. Then the encryption becomes the substitution of each 8-bit plaintext block with a 8-bit ciphertext block. At each plaintext we will have 16 bytes and each byte will be substituted by a byte in ciphertext.

Since this becomes substituting a byte with another byte, for instance for two different plaintexts which have 0 at their first bytes, the first byte of their ciphertexts will be the same. Then if we can learn the corresponding ciphertext bytes, for all possible bytes (0,1,...,255) at every possible position on the plaintext then we can solve the ciphertext easily. So with chosen plaintext attack, if we choose [0, 0, 0,...,0], [1,1,1,...,1], ... , [255,255,255,... ,255] and get the ciphertext bytes for each of these plaintexts then when we get a ciphertext, with table lookup we can solve each 8-bit (byte) one by one easily and construct the plaintext.

So with moderate effort we can break the modified AES.

5. If the ciphertext block C_i is corrupted during transmission, then the block P_i will be decrypted incorrectly because $P_i = D_k(C_i) \oplus C_{i-1}$ and the decryption function will return an incorrect value. Also the block $P_{i+1} = D_k(C_{i+1}) \oplus C_i$ will be decrypted incorrectly since we are XOR'ing with corrupted C_i. Then in total 2 of the blocks will be decrypted incorrectly and then it will resynchronize itself.