**Homework #2**

Due date: 20/10/2019

Notes:

* If you used Python codes for questions, compress them along with an answer sheet (a docx or pdf file).
* Name your winzip file as “SEC501\_hw02\_yourname.zip”
* Attached are “myntl.py”, “lfsr.py”, and “bonus\_helper.py” that you can use for the homework questions.

1. (**20 pts**) Consider the group .
   1. (**10 pts**) How many generators are there in ? Find at least two generators in .

Powers of 2 and 6 generate all elements of . Therefore, they are generators.

* 1. (**10 pts**) Find a subgroup of , whose order is 5. Find also its generator.

H = {1, 9, 20, 58, 34} is a subgroup of . 9 generates it.

1. (**20 pts**) Consider the following numbers:

p = 5527064775949971276700546474393760569152256071781455112554572252477078482817219303694924773296938028736900310336193124455858291501008953781025760084204617

q = 10263715010889663011237581846368625560721472564413404816414563312121711111200512062338133630430698017953660509772503749526855821347556114413740814256720609

n=p×q

c= c: 6016447327565519594114000681088119027251827426178525282923410539353402986195693266732998286093991346837854685945523958512316911535914428131590560765331415513592957120361162167163223354643895818554169459592360948153053758601978139504376688619229954806515113869761029037549195053798048200751500102605855423415

e = 67

Compute m = cd mod n (where d = e-1 mod φ(n)).

n = 56728217707077232293617084210299931257452691958908569907702847569944125011032128859439275442928907413529981280135882031249581123964245041413231153352601670244385793199015989262875852446426828022908038971753969311869202290420343060705458569514255672332644055782565668458166300776544926703650553643138556851753

φ(n) = 56728217707077232293617084210299931257452691958908569907702847569944125011032128859439275442928907413529981280135882031249581123964245041413231153352601654453606006359381701324747531684040698149179402776894040342733637691630749042974092536455851944696597365221745559761292318062432078138582358876564215926528

Using EEA we can find

d = 17780486146994356390536698036064157558306067628911641314354623865206367540771264269376489317932941129613874729594828696361809009003718595069818719707531861843667554232045010862980569632311263598996529228578729062647856589914115371678446914411535684158634995069502339626673711631508561804630291588176843797867

m = cd mod n =

30256242323116471143377579036851734956599566784957808777900084159344311403348077243305468893900944356797865493742870723640786398026030923177760147308711037452403389422258824881485692843121042180280211955250327896263032672055294111900141070981523217773015658586930923177732713966172972763510074059270940820416

1. (**30 pts**) Solve the following equations of the form ax ≡ b mod n and find all solutions for x if a solution exists. In case there is no solution, your answer must be “NO SOLUTION”, and explain why there is no solution.
   1. n = 333837116253674643166082492900

a = 57063337401967433471889139534

b = 397555361861029295385484594412

d= gcd(a, n) = 2 and 2 divides b then there are two solutions.

One solution can be obtained as follows:

d divides b

Now we have an equation:

≡ mod

≡

* 1. n = 333837116253674643166082492900

a = 176622984297114106732586191098

b = 84172329859897226978948124629

d=gcd(a, n) = 2 and 2 does not divide b. There is no solution.

* 1. n = 333837116253674643166082492900

a = 320736651991764172584335713727

b = 30472957776104045808802882504

d = gcd(a, n) = 1. There is a solution.

solution : 327252728639173874206458501252

1. (**15 pts**) Consider the following binary connections polynomials for LFSR:

p1(x) = x5 + x2 + 1

p2(x) = x5 + x3 + x2 + 1

Do they generate maximum period sequences? (**Hint:** You can use the functions in lfsr.py)

p2(x) = x5 + x2 + 1: when we start with 00001 and run the LFSR, we obtain the following sequence of LFSR states: 00001🡪 10000 🡪 01000 🡪 10100 🡪 01010 🡪 10101 🡪 11010 🡪 11101 🡪 01110 🡪 10111 🡪 11011 🡪 01101 🡪 00110 –> 00011 🡪 10001 🡪 11000 🡪 11100 🡪 11110 🡪 11111 🡪 01111 🡪 00111 🡪 10011 🡪 11001 🡪 01100 🡪 10110 🡪 01011 🡪 00101 🡪 10010 🡪 01001 🡪 00100 🡪 00010 🡪 00001

The period is 16, which is maximum. Then, the answer is YES.

p1(x) = x5 + x3 + x2 + 1: when we start with 00001 and run the LFSR, we obtain the following sequence of LFSR states: 00001 🡪 10000 🡪 01000 🡪 10100 🡪 11010 🡪 11101 🡪 11110 🡪 01111 🡪 10111 🡪 01011 🡪 00101 🡪 00010 🡪 00001.

The period is 12, which is less than the maximum period of 31. Then the answer is NO.

1. (**15 pts**) Consider a random number generator that generates the following sequences. Are they unpredictable? (**Hint:** You can use the functions in lfsr.py)

x1 = [1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0]

x2 = [0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

x3 = [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]

No, they are predictable. If we apply the Berlekamp-Massey algorithm to them as follows

print ("L1 and C1(x): ", BM(x1))

print ("L2 and C2(x): ", BM(x2))

print ("L3 and C3(x): ", BM(x3))

You will see their linear complexity are 31 whereas their lengths are 100.

**Bonus Question**

1. (**20 pts**) Consider the following ciphertext bit stream encrypted using a stream cipher. And you strongly suspect that an LFRS is used to generate the key stream:

ctext = [0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0]

Also, encrypted in the ciphertext you also know that there is a message to you from the instructor; and therefore the message starts with “Dear Student”. Find the plaintext. For this you need to find the connection polynomial of the LFSR first. Note that the ASCII encoding (seven bits for each ASCII character) is used.

**(Hint:** You can use the ASCII2bin(msg) and bin2ASCII(msg) functions (in bonus\_helper.py) to make conversion between ASCII and binary)

The connection polynomial: x17 + x3 + 1.

Dear Student,

You have worked hard and that paid off:)

You have just earned 20 bonus points. Congrats!

Best,

Erkay Savas