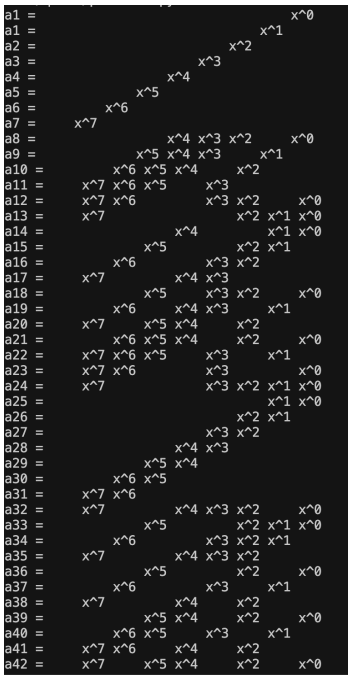
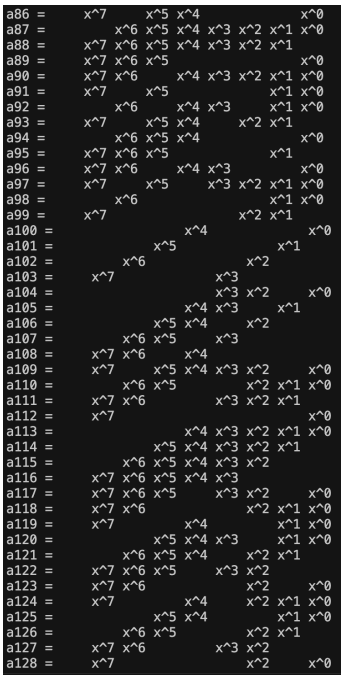
Quiz 4

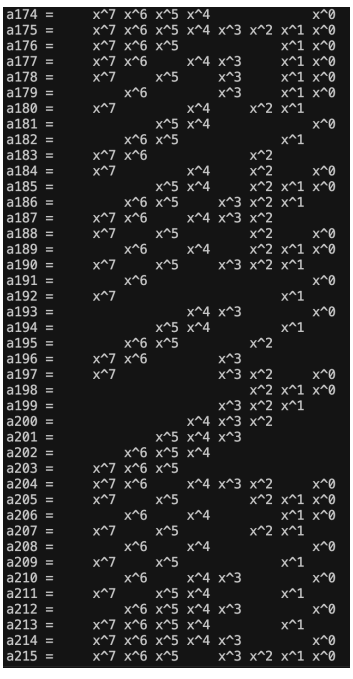
111550057 資工15 莊婷馨

Problem 1

a) Yes, it is a primitive polynomial. I wrote a code to do mod $(X^8 + X^4 + X^3 + X^2 + 1)$ and the result looks like this. (screenshot of the code is attached at the end of the report)







b) The maximum cycle length is 255.

c) No, not all irreducible polynomials are primitive. For instance,

$P(X) = X^8 + X^4 + X^3 + X + 1$ is irreducible. However, it is not primitive.

Problem 2

a) The method of encrypting is as follows. Generate key stream by the way LFSR works (xor the bits of the polynomial). Encrypt 8 bits at a time because that represents a character. Encrypt until the end of the string then output the cipher text. Decryption works the same. Generate key stream the same way and decrypt the cipher text back to plain text.

How to run the code:

Run " python problem2.py " in the terminal.

Result of encryption and decryption:

```
Encrypted message:
 @j7Ì1¯P^ëgS³Ç`ñ6þæ,C{8Ò ¿FOÿ{_¶Özî+éõ"Uj+Ô<©D^äqE¤Újî/üè"Sg=Ü$³C^ùaY³Ü+øä–Bl<Ô!³I\ælU¯Þeâ'àó!M{4Æ&²FOþ
}S·Üeã9ñâ0V{.Ü>¶DTäa_®Æ}è=õæ*E{=×+®O^ãqSiÇhó(õâ}1Ü7¬BHåxS´Ûgà2åâ$S{8Á7¨STípB¨Ö}ï>þö @p0Û6³QRî`W¬ßhá+õój
1Ô& FHþ}S©×hó7ñõ&Ej6Á:¿DIïtB©Üfá0åó–Hh<Ç!³SBã{B¨Ö`õ,äñ"B{
Decrypted message:
 ATNYCUWEARESTRIVINGTOBEAGREATUNIVERSITYTHATTRANSCENDSDISCIPLINARYDIVIDESTOSOLVETHEINCREASINGLYCOMPLEXP
ROBLEMSTHATTHEWORLDFACESWEWILLCONTINUETOBEGUIDEDBYTHEIDEATHATWECANACHIEVESOMETHINGMUCHGREATERTOGETHERTH
ANWECANINDIVIDUALLYAFTERALLTHATWASTHEIDEATHATLEDTOTHECREATIONOFOURUNIVERSITYINTHEFIRSTPLACE
```

b) Yes, every 8 bits will reveal 1 bit of keystream.
Yes, it is possible to find out the characteristic polynomial by solving linear equations.
By the set of equations :

$$z_8 = z_7c_7 + z_6c_6 + z_5c_5 + z_4c_4 + z_3c_3 + z_2c_2 + z_1c_1 + z_0c_0 \mod 2$$
$$z_9 = z_8c_7 + z_7c_6 + z_6c_5 + z_5c_4 + z_4c_3 + z_3c_2 + z_2c_1 + z_1c_0 \mod 2$$
$$\cdots$$
$$z_{15} = z_{14}c_7 + z_{13}c_6 + z_{12}c_5 + z_{11}c_4 + z_{10}c_3 + z_9c_2 + z_8c_1 + z_7c_0 \mod 2$$

Knowing z_0 to z_{15} , we can compute c_0 to c_7. This solves the characteristic polynomial.

Problem 3

a) How to run the program:

Run "python problem3.py" in the terminal.

```
Naive algorithm:
1234 : 61959
1243 : 31563
1324 : 78294
1342 : 30959
1423 : 15696
1432 : 31011
2134 : 78135
2143 : 31081
2314 : 77895
2341 : 78428
2413 : 31121
2431 : 31344
3124 : 62216
3142 : 46936
3214 : 62808
3241 : 31326
3412 : 31181
3421 : 46806
4123 : 15761
4132 : 15739
4213 : 15788
4231 : 31116
4312 : 31423
4321 : 31414
```

```
Fisher–Yates shuffle:
1234 : 41838
1243 : 41977
1324 : 41612
1342 : 41699
1423 : 41650
1432 : 41546
2134 : 41940
2143 : 41752
2314 : 41448
2341 : 41515
2413 : 41776
2431 : 42192
3124 : 41580
3142 : 41456
3214 : 41812
3241 : 41674
3412 : 41580
3421 : 41204
4123 : 41823
4132 : 41540
4213 : 41275
4231 : 41460
4312 : 41915
4321 : 41736
```

b) Fisher-Yates shuffle is better, since the result is less biased than the result of Naive algorithm.
c) The drawback of Naive algorithm is that the permutation is not evenly distributed. It is caused by the uneven probability of each permutation in the algorithm. Some positions are switched more often.

Program of Problem1

```
def xor(a,b):
    if a==0 and b==0:
        return 0
    elif a==1 and b==1:
        return 0
    else:
        return 1

def printpoly(num,poly):
    print('a'+str(num),'=',end=' ')
    for i in range(7,-1,-1):
        if poly[i]==1:
            print('x^'+str(i),end=' ')
        else:
            print('   ',end=' ')
    print('')


primpoly = [1,0,0,0,1,1,1,0,1]
testpoly = [1,0,0,0,0,0,0,0]

printpoly(1,testpoly)

for i in range(255):
    temp=[0]
    for j in range(8):
        temp.append(testpoly[j])
    testpoly = temp

    if testpoly[8]==1:
        for j in range(9):
            testpoly[j]=xor(testpoly[j],primpoly[j])
    printpoly(i+1,testpoly)
```