Problem 1

(a) Yes，x^8 + x^4 + x^3 + x^2 + 1在F(2)上沒有根，因為該多項式本身不可約分
(b) 2^8 - 1 = 255
(c) No, 有反例

Examples [edit]

Over $\mathrm{GF}(3)$ the polynomial x^2+1 is irreducible but not primitive because it divides x^4-1: its roots generate a cyclic group of order 4, while the multiplicative group of $\mathrm{GF}(3^2)$ is a cyclic group of order 8. The polynomial x^2+2x+2, on the other hand, is primitive. Denote one of its roots by α. Then, because the natural numbers less than and relatively prime to $3^2-1=8$ are 1, 3, 5, and 7, the four primitive roots in $\mathrm{GF}(3^2)$ are α, $\alpha^3=2\alpha+1$, $\alpha^5=2\alpha$, and $\alpha^7=\alpha+2$. The primitive roots α and α^3 are algebraically conjugate. Indeed $x^2+2x+2=(x-\alpha)(x-(2\alpha+1))$. The remaining primitive roots α^5 and $\alpha^7=(\alpha^5)^3$ are also algebraically conjugate and produce the second primitive polynomial: $x^2+x+2=(x-2\alpha)(x-(\alpha+2))$.

Problem 2

```python
def LFSR(state, poly):
    fb = 0
    for i in range(len(state)):
        fb ^= state[i] & poly[i+1]
    state.pop(0)
    state.append(fb)
    return fb


def transfer(text, poly, key):
    state = [int(bit) for bit in key]
    result = ''
    btext = ''.join(format(ord(char), '08b') for char in text)
    for idx in range(0, len(btext), 8):
        b = ''
        for i in range(8):
            fb = LFSR(state, poly)
            b += str(int(btext[idx+i]) ^ fb)
        r_char = ''
        r_char += chr(int(b, 2))
        result += r_char
    return result


plaintext = 'ATNYCUWEARESTRIVINGTOBEAGREATUNIVERSITYTHATTRANSCENDSDISC
poly = [1, 0, 0, 0, 1, 1, 1, 0, 1]
key = '00000001'
e = transfer(plaintext, poly, key)
print('Encrypted:\n',e)
d = transfer(e, poly, key)
print('Decrypted:\n',d)
```

```
Encrypted:
  Ø½ÂÂvn];ðJ
ëmñË2:@õ§®p/
            ¢È¡ÁÞcjZ;ðRéwúÚ:.Qù¶¶l/¦Ò¾ÜÃrA<íJä|÷ÇSã¯´b-·Ë¤ÐßgdL#ç

Þ©ÁÂe
dI*öYäxðÞ0<\ù¨¾l7
                 °×¤ÌÆq
n\.îHønÿÚ
         ;8[ô£»q) ß¼ÚÓh\*ãU
ãñÜ
&3[æ£¨v( Ï ÐÁ~xZ?î_
Decrypted:
 ATNYCUWEARESTRIVINGTOBEAGREATUNIVERSITYTHATTRANSCENDSDISCIPLINARYDIVIDESTOSOLVETHEINC
REASINGLYCOMPLEXPROBLEMSTHATTHEWORLDFACESWEWILLCONTINUETOBEGUIDEDBYTHEIDEATHATWECANACH
IEVESOMETHINGMUCHGREATERTOGETHERTHANWECANINDIVIDUALLYAFTERALLTHATWASTHEIDEATHATLEDTOTH
ECREATIONOFOURUNIVERSITYINTHEFIRSTPLACE
```

(a) 使用x^8 + x^4 + x^3 + x^2 + 1作為characteristic polynomial，並把initial key 設為00000001,然後與plaintext丟到transfer函式裡先進行string to binary的處理，接著再以8 bit一組的概念1bit 1bit的丟給LSFR函式做運算再經過 binary to string的處理拼出encrypted text。decrypted則是把encrypted text與相同的characteristic polynomial與initial key再丟到transfer函式即可得到plaintext

(b) Yes，可以利用LFSR的輸出位元建立線性方程組，進而推導出特徵多項式。

Problem 3

(a)

```python
import random

def Naive(cards):
    count = {}
    for i in range(1000000):
        s_cards = cards[:]
        for j in range(len(cards)):
            n = random.randint(0, len(cards) - 1)
            s_cards[j], s_cards[n] = s_cards[n], s_cards[j]
        if tuple(s_cards) in count.keys():
            count[tuple(s_cards)] += 1
        else:
            count[tuple(s_cards)] = 1
    return count

def Fisher_Yates(cards):
    count = {}
    for i in range(1000000):
        s_cards = cards[:]
        for j in range(len(cards)-1, 0, -1):
            n = random.randint(0, j)
            s_cards[j], s_cards[n] = s_cards[n], s_cards[j]
        if tuple(s_cards) in count.keys():
            count[tuple(s_cards)] += 1
        else:
            count[tuple(s_cards)] = 1
    return count

```

```python
cards = [1, 2, 3, 4]
print('Naive:')
naive_count = Naive(cards)
for a, b in naive_count.items():
    print(f'{a} : {b}')
print('-------------------------------')
print('Fisher-Yates:')
FY_count = Fisher_Yates(cards)
for a, b in FY_count.items():
    print(f'{a} : {b}')
```

```
Naive:
(2, 4, 3, 1) : 43267
(2, 1, 4, 3) : 59023
(4, 3, 2, 1) : 38863
(4, 2, 1, 3) : 35274
(2, 4, 1, 3) : 42792
(1, 3, 4, 2) : 54797
(4, 1, 3, 2) : 35278
(3, 1, 4, 2) : 43056
(1, 2, 3, 4) : 38567
(1, 4, 2, 3) : 43199
(3, 2, 4, 1) : 42894
(2, 3, 1, 4) : 54699
(2, 3, 4, 1) : 54456
(4, 1, 2, 3) : 31210
(4, 2, 3, 1) : 31644
(3, 1, 2, 4) : 42756
(2, 1, 3, 4) : 38951
(4, 3, 1, 2) : 39246
(3, 4, 2, 1) : 38861
(3, 2, 1, 4) : 35070
(3, 4, 1, 2) : 42452
(1, 4, 3, 2) : 35463
(1, 2, 4, 3) : 39381
(1, 3, 2, 4) : 38801
```

```
Fisher-Yates:
(2, 3, 1, 4) : 41856
(4, 2, 1, 3) : 41765
(1, 3, 4, 2) : 41821
(2, 1, 3, 4) : 41577
(4, 1, 3, 2) : 41363
(4, 3, 1, 2) : 41732
(2, 4, 3, 1) : 41804
(3, 1, 2, 4) : 42045
(1, 2, 3, 4) : 41875
(1, 4, 3, 2) : 41971
(1, 2, 4, 3) : 41534
(4, 3, 2, 1) : 41801
(4, 2, 3, 1) : 41521
(2, 1, 4, 3) : 41561
(2, 4, 1, 3) : 41263
(4, 1, 2, 3) : 41720
(3, 2, 4, 1) : 41513
(3, 1, 4, 2) : 41734
(1, 3, 2, 4) : 41498
(3, 4, 2, 1) : 41454
(3, 2, 1, 4) : 41455
(1, 4, 2, 3) : 41708
(2, 3, 4, 1) : 41672
(3, 4, 1, 2) : 41757
```

(b) Fisher-Yates更好，因為他各種組合的分布更均勻，更符合現實中的機率

(c) naive的缺點體現在每個組合出現次數不平衡，與Fisher-Yates差別在於每次取n的範圍不同，naive每次n的範圍都一樣，可能導致各張卡牌被選到的次數不均等，進而造成組合分布不均衡