1   Problem 1

1. Yes. The polynomial is irreducible over $GF(2)$, and it generates a maximal length LFSR sequence of length 255.

2. The maximum cycle length generated by $x^8 + x^4 + x^3 + x^2 + 1$ is $2^8 - 1 = 255$. We can use the following program to verify the answer:

```python
key = [0, 0, 0, 0, 0, 0, 0, 1]
for i in range(1, 1000):
    if key == [0, 0, 0, 0, 0, 0, 0, 1]:
        print(f'{i}: {key}')
    tmp = key[0]
    for j in range(7):
        key[j] = key[j + 1]
    key[7] = 0
    if tmp:
        key[3] = 1 ^ key[3]
        key[4] = 1 ^ key[4]
        key[5] = 1 ^ key[5]
        key[7] = 1 ^ key[7]
```

Output:

```
1: [0, 0, 0, 0, 0, 0, 0, 1]
256: [0, 0, 0, 0, 0, 0, 0, 1]
511: [0, 0, 0, 0, 0, 0, 0, 1]
766: [0, 0, 0, 0, 0, 0, 0, 1]
```

3. Not all irreducible polynomials are primitive polynomials. For example, $x^4 + x^3 + x^2 + x + 1$ is irreducible over $GF(2)$, but it is not primitive because it does not generate a maximal length LFSR sequence of length 15.

```python
key = [0, 0, 0, 1]
for i in range(1,20):
    if key == [0, 0, 0, 1]:
        print(f'{i}: {key}')
    tmp = key[0]
    for j in range(3):
        key[j] = key[j + 1]
    key[3] = 0
    if tmp:
        key[0] = 1 ^ key[0]
        key[1] = 1 ^ key[1]
        key[2] = 1 ^ key[2]
        key[3] = 1 ^ key[3]
```

Output:

```
1: [0, 0, 0, 1]
6: [0, 0, 0, 1]
11: [0, 0, 0, 1]
16: [0, 0, 0, 1]
```

2   Problem 2

1. output:

```
Cipher text: 010000000101011001001010010100010101001101110101000101111100010
1010111000110100000110001101110111001100111010101010110100111000000000101110
1011001101010000011101111101100110111101011111000100011001000010100010100001
1010011010100110001100101001011101000100111001011011000100001110011001111011
0110000011110110011010110000100100010100101011110000100100011101111001000000
0110100010111000000001011100100101001011010011001000111011011000001100111
1001100011110111100110011100110011110111100000100001100111111111100110000011
0111110100010100010010100110100000110101100011011111011110011000110000101110
1101001000000010100100110101000111101101001001011010110100010100110101111010
1001010001000010011110100001111101010110110101000100100011110010000010111111
0111100110010101011011001010011111111000011111100001101001001011100100001011
0110000111010011110101011001110001001001011000001111010010010111110001111110
1010101001011011101100011100110101111101000101001010111011111101111111100011000
0101011011000011011010010011000000011000110001001110100100101111001100010000
1000011110110010001010010100111111010000011011010000011101000000011111101000
0010110100110100000010001111011110001100011010011011011010001011111100000000
1011111110011000111011101101001111000011101110100000100100001111011000100111
1101011101001000011101011001101001000001110100100111110001100001011110101101
00110100111111111101000111110101000001011010010101100101011001001011011101111
0000111111101111001111000010110010000011110110000110001000101101100101001111
0001000000101111010110000000110101100101110001100000011000010000111111000101
0010000100101101001010100011111010010001010110100100101011110011000011101111
1000100010100111000010001100011111100001100001011110010111101101001101001010
1110100100110111100000011111011010000100111000110010100000101001101100010000
1110011010101011011110011101110111000101110011010001101110110111101110101011111
1100111001100101001011000010000100110010000011101111110010011100110111111101
0100010011011110010110101101001100010001000001001110111101001000100111100000
1010101010100101001000000101001011010000100110010100000001110101000101010100011
1010100111010101001111000101111001000010001100111010000011001110101100110010
0000011001111000100100111101110011000000011011011010110100100111101000010010
0110000111100000100101100001101101010001110101000111011100010000111010100000011
0110101010111011000101111
```

Decrypted text: ATNYCUWEARESTRIVINGTOBEAGREATUNIVERSITYTHATTRANSCENDSDISCIPL
INARYDIVIDESTOSOLVETHEINCREASINGLYCOMPLEXPROBLEMSTHATTHEWORLDFACESWEWILLCONT
INUETOBEGUIDEDBYTHEIDEATHATWECANACHIEVESOMETHINGMUCHGREATERTOGETHERTHANWECAN
INDIVIDUALLYAFTERALLTHATWASTHEIDEATHATLEDTOTHECREATIONOFOURUNIVERSITYINTHEFI
RSTPLACE

2. Yes. Given a 8-stage LSFR, we know:

$$
\begin{cases}
a_n = (a_{n+1}C_7 + a_{n+2}C_6 + a_{n+3}C_5 + ... + a_{n+6}C_2 + a_{n+7}C_1 + a_{n+8}C_0) \mod 2 \\
a_{n+1} = (a_{n+2}C_7 + a_{n+3}C_6 + a_{n+4}C_5 + ... + a_{n+7}C_2 + a_{n+8}C_1 + a_{n+9}C_0) \mod 2 \\
a_{n+2} = (a_{n+3}C_7 + a_{n+4}C_6 + a_{n+5}C_5 + ... + a_{n+8}C_2 + a_{n+9}C_1 + a_{n+10}C_0) \mod 2 \\
a_{n+3} = (a_{n+4}C_7 + a_{n+5}C_6 + a_{n+6}C_5 + ... + a_{n+9}C_2 + a_{n+10}C_1 + a_{n+11}C_0) \mod 2 \\
a_{n+4} = (a_{n+5}C_7 + a_{n+6}C_6 + a_{n+7}C_5 + ... + a_{n+10}C_2 + a_{n+11}C_1 + a_{n+12}C_0) \mod 2 \\
a_{n+5} = (a_{n+6}C_7 + a_{n+7}C_6 + a_{n+8}C_5 + ... + a_{n+11}C_2 + a_{n+12}C_1 + a_{n+13}C_0) \mod 2 \\
a_{n+6} = (a_{n+7}C_7 + a_{n+8}C_6 + a_{n+9}C_5 + ... + a_{n+12}C_2 + a_{n+13}C_1 + a_{n+14}C_0) \mod 2 \\
a_{n+7} = (a_{n+8}C_7 + a_{n+9}C_6 + a_{n+10}C_5 + ... + a_{n+13}C_2 + a_{n+14}C_1 + a_{n+15}C_0) \mod 2
\end{cases}
$$

Knowing a_0, a_1, ..., a_{15}, can compute C_0, C_1, ..., C_7, thus can solve a 8-stage LFSR.

3. $C_0 = 1$, $C_1 = 0$, $C_2 = 0$, $C_3 = 0$, $C_4 = 1$, $C_5 = 1$, $C_6 = 1$, $C_7 = 0$

```
f16keyoutput = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0]
solution = [0, 0, 0, 0, 0, 0, 0, 0]

for i in range(255):
    flg = 1
    for j in range(8):
        c = 0
        for k in range(8):
            c += solution[k] * f16keyoutput[j+1+k]
        if(f16keyoutput[j] != c%2):
            flg = 0
            break
    if flg:
        print(solution)
        break

    cnt = 0
    while(solution[cnt]):
        solution[cnt] = 0
        cnt += 1
    solution[cnt] = 1
```

Output:

```
[0, 1, 1, 1, 0, 0, 0, 1]
```

3   Problem 3

1. output:

```
Naive algorithm:
(1, 2, 3, 4): 38688 (2, 3, 1, 4): 54314 (3, 4, 1, 2): 42889
(1, 2, 4, 3): 39161 (2, 3, 4, 1): 54895 (3, 4, 2, 1): 38954
(1, 3, 2, 4): 39184 (2, 4, 1, 3): 43461 (4, 1, 2, 3): 31338
(1, 3, 4, 2): 54739 (2, 4, 3, 1): 42571 (4, 1, 3, 2): 35229
(1, 4, 2, 3): 43078 (3, 1, 2, 4): 43135 (4, 2, 1, 3): 35416
(1, 4, 3, 2): 35168 (3, 1, 4, 2): 42929 (4, 2, 3, 1): 31271
(2, 1, 3, 4): 39189 (3, 2, 1, 4): 35047 (4, 3, 1, 2): 39013
(2, 1, 4, 3): 58516 (3, 2, 4, 1): 42893 (4, 3, 2, 1): 38922
Average: 41666.666666666664, Standard Deviation: 7168.251295740747

Fisher – Yates shuffle:
(1, 2, 3, 4): 41951 (2, 3, 1, 4): 41694 (3, 4, 1, 2): 41600
(1, 2, 4, 3): 41832 (2, 3, 4, 1): 41521 (3, 4, 2, 1): 41528
(1, 3, 2, 4): 41968 (2, 4, 1, 3): 41997 (4, 1, 2, 3): 41651
(1, 3, 4, 2): 42073 (2, 4, 3, 1): 41199 (4, 1, 3, 2): 41263
(1, 4, 2, 3): 41594 (3, 1, 2, 4): 41737 (4, 2, 1, 3): 41348
(1, 4, 3, 2): 41466 (3, 1, 4, 2): 41953 (4, 2, 3, 1): 41777
(2, 1, 3, 4): 41668 (3, 2, 1, 4): 41553 (4, 3, 1, 2): 41643
(2, 1, 4, 3): 41730 (3, 2, 4, 1): 41299 (4, 3, 2, 1): 41955
Average: 41666.666666666664, Standard Deviation: 240.0315372797685
```

2. Fisher-Yates shuffle is better because it demonstrates a more uniform distribution, as indicated by a lower standard deviation compared to the Naïve algorithm.
3. The Naïve algorithm is known to be biased because the probability of each element in each position is not equal, primarily due to the fact that it always selects from the entire range of indices for swapping. In contrast, the Fisher-Yates shuffle ensures each position can only swap with positions before it (or itself), leading to a more uniform and unbiased distribution.

4   Appendix

Package used in the program: numpy

```
pip install numpy
```