Question 1: Show that $S1(x1) \oplus S1(x2) \neq S1(x1 \oplus x2)$

1. x1 = 000000, x2 = 000001

    s1(000000) = 14

    s1(000001) = 00

    $x1 \oplus x2$ = 000001

    $s1(x1 \oplus x2)$ = 00

    $14 \oplus 00 \Rightarrow 14 \mathrel{!}= 00$

2. x1 = 111111, x2 = 100000

    s1(111111) = 13

    s1(100000) = 04

    $x1 \oplus x2$ = 011111

    $s1(x1 \oplus x2)$ = 08

    $13 \oplus 04 \Rightarrow 09 \mathrel{!}= 08$

3. x1 = 101010, x2 = 010101

    s1(101010) = 06

    s1(010101) = 12

    $x1 \oplus x2$ = 111111

    $s1(x1 \oplus x2)$ = 13

    $06 \oplus 12 \Rightarrow 10 \mathrel{!}= 13$

Question 2: We want to verify that IP(·) and IP−1(·) are truly inverse operations. We consider a vector x = (x1,x2,...,x64) of 64 bit. Show that IP−1(IP(x)) = x for the first five bits of x, i.e. for xi, i = 1,2,3,4,5.

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}

40	8	48	16	56	24	64	32

39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

We can easily see that IP(·) and IP−1(·) simply by observing that the IP table's first row is decreasing by 8 (going across) while 1 to 5 on IP–1 goes up the table (each row contains 8 numbers, so the position is changing by 8).

Question 3: What is the output of the first round of the DES algorithm when the plaintext and the key are both all zeros?

R1 = L0 ⊕ f(R0, K1)
f(R0, K1) = Expansion of all zeros = zeros
Permutation of K for keys zeros also = zeros
0 ⊕ 0 = 0
S-Box[1-8](zeros) = 1110 1111 1010 0111 0010 1100 0100 1101
L1 = R0 = 0000 0000 0000 0000 0000 0000 0000 0000
R1 = 1110 1111 1010 0111 0010 1100 0100 1101 ⊕ L0 (all zeros, so no change)

Question 4: What is the output of the first round of the DES algorithm when the plaintext and the key are both all ones?

R1 = L0 ⊕ f(R0, K1)
f(R0, K1) = Expansion of all ones = ones (R0)
Permutations of all ones for key also = ones

1 ⊕ 1 = 0s
S-Box[1-8](zeros) = 1110 1111 1010 0111 0010 1100 0100 1101
L1 = R0 = 1111 1111 1111 1111 1111 1111 1111 1111
R1 = 1110 1111 1010 0111 0010 1100 0100 1101 ⊕ L0 (all ones, so flip each bit) =
0001 0000 0101 1000 1101 0011 1011 0010

Question 5: Remember that it is desirable for good block ciphers that a change in one input bit affects many output bits, a property that is called diffusion or the avalanche effect. We try now to get a feeling for the avalanche property of DES. We apply an input word that has a "1" at bit position 57 and all other bits as well as the key are zero. (Note that the input word has to run through the initial permutation.)

1. How many S-boxes get different inputs compared to the case when an all-zero plaintext is provided?

- The 57th position is 25th on R0. There are two 25 on the expansion table, so we end up with two 1s (in positions 36, 38).
- 36 and 38 belong to two different buckets, so 2 S-boxes will be affected.

2. What is the minimum number of output bits of the S-boxes that will change according to the S-box design criteria?
    - By design, at least two bits will differ (when one bit is changed).

3. What is the output after the first round?
    - L1 = R0 = 0000 0000 0000 0000 0000 0000 1000 0000
    - R1 = 1110 1111 1010 0111 0010 **1010 1101** 1101 ⊕ L0 (all zeros, so no change)
    - **Identical to Question 3 except for the bolded parts

4. How many output bit after the first round have actually changed compared to the case when the plaintext is all zero? (Observe that we only consider a single round here. There will be more and more output differences after every new round. Hence the term avalanche effect.)
    - 1100 0100
    - 1**01**0 **1**10**1**
    - A total of 4 bits were different.

Question 6: Assume we perform a known-plaintext attack against DES with one pair of plaintext and ciphertext. How many keys do we have to test in a worst-case scenario if we apply an exhaustive key search in a straightforward way? How many on average?

Worst Case: Key size is 56, there are two values for each key (0 or 1), so **2^56** for every combination.

Average: On average, we are assuming that around half the keys would need to be tested before finding the correct key (since each key is equally likely): 2^56 /2 ⇒ **2^55**

Question 7:

Plaintext	0000 0000 0000 0000
Round Key	BBBB 5555 5555 EEEE
State after KeyAdd	BBBB 5555 5555 EEEE
State after S-Layer	8888 0000 0000 1111
State after P-Layer	F000 0000 0000 000F

Key	BBBB 5555 5555 EEEE FFFF

Key State after Rotation	DFFF F777 6AAA AAAA BDDD
Key State after S-Box	7222 2DDD AFFF FFFF 8777
Key State after CounterAdd	7222 2DDD AFFF FFFE 8776
Round Key for Round 2	7222 2DDD AFFF FFFE

Part 2: LFSR

Test Cases

```
n = 5  # Number of bits
seed = 0b01111  # Initial seed
tap = 2  # Tap position

n = 5  # Number of bits
seed = 0b01111  # Initial seed
tap = 4  # Tap position

n = 4  # Number of bits
seed = 0b01001  # Initial seed
tap = 1  # Tap position
```

Results

```
(base) Macs-MacBook-Air:submission_assignment2 reaper$ python a2.py
[0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0]
(base) Macs-MacBook-Air:submission_assignment2 reaper$ python a2.py
[0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
(base) Macs-MacBook-Air:submission_assignment2 reaper$ python a2.py
[1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0]
(base) Macs-MacBook-Air:submission_assignment2 reaper$
```

For the implementation, refer to a2.py.