23. Write a python program for Encrypt and decrypt in counter mode using one of the following ciphers: affine modulo 256, Hill modulo 256, S-DES. Test data for S-DES using a counter starting at 0000 0000. A binary plaintext of 0000 0001 0000 0010 0000 0100 encrypted with a binary key of 01111 11101 should give a binary plaintext of 0011 1000 0100 1111 0011 0010. Decryption should work correspondingly.

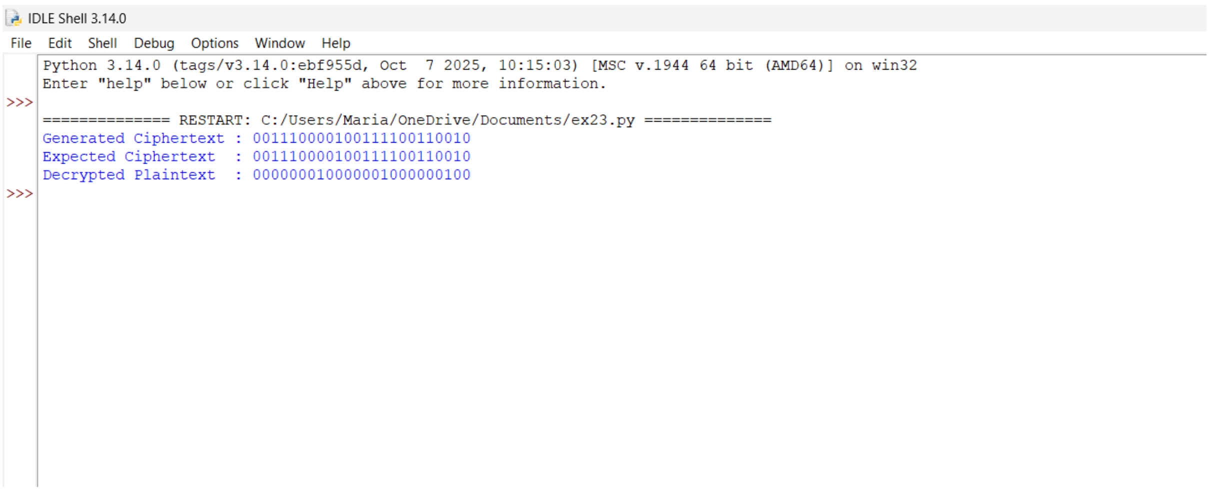
```
# ---------------------------

# S-DES IMPLEMENTATION

# ---------------------------


P10  = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]

P8   = [6, 3, 7, 4, 8, 5, 10, 9]

P4   = [2, 4, 3, 1]

IP   = [2, 6, 3, 1, 4, 8, 5, 7]

IPinv= [4, 1, 3, 5, 7, 2, 8, 6]

EP   = [4, 1, 2, 3, 2, 3, 4, 1]


S0 = [

    [1, 0, 3, 2],

    [3, 2, 1, 0],

    [0, 2, 1, 3],

    [3, 1, 3, 2]

]


S1 = [

    [0, 1, 2, 3],

    [2, 0, 1, 3],

    [3, 0, 1, 0],

    [2, 1, 0, 3]
```

]

Code:

```
def permute(bits, table):
    return ''.join(bits[i-1] for i in table)
def left_shift(bits, n):
    return bits[n:] + bits[:n]
def key_generation(key10):
    k = permute(key10, P10)
    L, R = k[:5], k[5:]
    L1, R1 = left_shift(L, 1), left_shift(R, 1)
    K1 = permute(L1 + R1, P8)
    L2, R2 = left_shift(L1, 2), left_shift(R1, 2)
    K2 = permute(L2 + R2, P8)
    return K1, K2
def f_function(bits, key):
    L, R = bits[:4], bits[4:]
    ER = permute(R, EP)
    x = ''.join('1' if ER[i] != key[i] else '0' for i in range(8))
    L0, R0 = x[:4], x[4:]
    row, col = int(L0[0]+L0[3], 2), int(L0[1]+L0[2], 2)
    s0 = format(S0[row][col], '02b')
    row, col = int(R0[0]+R0[3], 2), int(R0[1]+R0[2], 2)
    s1 = format(S1[row][col], '02b')
    out = permute(s0 + s1, P4)
    return ''.join('1' if L[i] != out[i] else '0' for i in range(4)) + R
def sdes_encrypt(block8, K1, K2):
    x = permute(block8, IP)
    y = f_function(x, K1)
    y = y[4:] + y[:4]       # swap
    z = f_function(y, K2)
    return permute(z, IPinv)
# CTR mode uses the same function for decrypt
sdes_decrypt = sdes_encrypt
# --------------------------
# CTR MODE IMPLEMENTATION
# --------------------------
def xor_bits(a, b):
    return ''.join('1' if a[i] != b[i] else '0' for i in range(len(a)))
def increment_counter(cnt):
    return format((int(cnt, 2) + 1) % 256, "08b")
def CTR_encrypt(plaintext, K1, K2, counter):
    blocks = [plaintext[i:i+8] for i in range(0, len(plaintext), 8)]
    cipher = ""
 cnt = counter
 for blk in blocks:
        keystream = sdes_encrypt(cnt, K1, K2)
        ciphertext_block = xor_bits(blk, keystream)
        cipher += ciphertext_block
        cnt = increment_counter(cnt)
 return cipher
def CTR_decrypt(ciphertext, K1, K2, counter):
    # CTR decryption is identical to encryption
    return CTR_encrypt(ciphertext, K1, K2, counter)
# --------------------------
# TEST VECTOR FROM QUESTION
# --------------------------
```

```
key10      = "0111111101"

plaintext  = "00000001" "00000010" "00000100"

expected   = "00111000" "01001111" "00110010"

counter    = "00000000"

K1, K2 = key_generation(key10)

cipher = CTR_encrypt(plaintext, K1, K2, counter)

decrypt = CTR_decrypt(cipher, K1, K2, counter)

print("Generated Ciphertext :", cipher)

print("Expected Ciphertext  :", expected)

print("Decrypted Plaintext  :", decrypt)
```



```
Python 3.14.0 (tags/v3.14.0:ebf955d, Oct  7 2025, 10:15:03) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
============== RESTART: C:/Users/Maria/OneDrive/Documents/ex23.py ==============
Generated Ciphertext : 001110000100111100110010
Expected Ciphertext  : 001110000100111100110010
Decrypted Plaintext  : 000000010000001000000100
>>>
```