15) Write a C 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.

PROGRAM:-

```
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]
]

def generate_subkeys(key):
    # Simplified key scheduling: extract 8-bit keys k1 and k2 from the 10-bit input
    k1 = key[:8]
    k2 = key[2:10]
    return k1, k2

def sdes_encrypt(plaintext_byte, key):
    # Simplified S-DES: uses 4 bits to look up S-boxes and return a byte
    row = ((plaintext_byte & 0b1000) >> 2) | (plaintext_byte & 0b0001)
    col = (plaintext_byte & 0b0110) >> 1
    s0 = S0[row][col]
```

```
    s1 = S1[row][col]
    return (s0 << 2) | s1

def ctr_encrypt(plaintext, key_bits, counter_start):
    k1, k2 = generate_subkeys(key_bits)
    counter = counter_start
    ciphertext = []
    for byte in plaintext:
        encrypted_counter = sdes_encrypt(counter, k1)
        ciphertext.append(byte ^ encrypted_counter)
        counter = (counter + 1) % 256  # Simulate 8-bit counter
    return ciphertext

def ctr_decrypt(ciphertext, key_bits, counter_start):
    # Decryption is the same as encryption in CTR mode
    return ctr_encrypt(ciphertext, key_bits, counter_start)

def print_binary(data):
    print(' '.join(format(byte, '08b') for byte in data))

# === Test ===
plaintext = [0b00000001, 0b00000010, 0b00000100]
key = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1]  # 10-bit key
counter_start = 0

print("Original plaintext:")
print_binary(plaintext)

ciphertext = ctr_encrypt(plaintext, key, counter_start)
print("\nEncrypted ciphertext:")
print_binary(ciphertext)
```

```
decrypted = ctr_decrypt(ciphertext, key, counter_start)

print("\nDecrypted plaintext:")

print_binary(decrypted)
```

OUTPUT:-

```
Original plaintext:
00000001 00000010 00000100

Encrypted ciphertext:
00000101 00001100 00000101

Decrypted plaintext:
00000001 00000010 00000100
```
