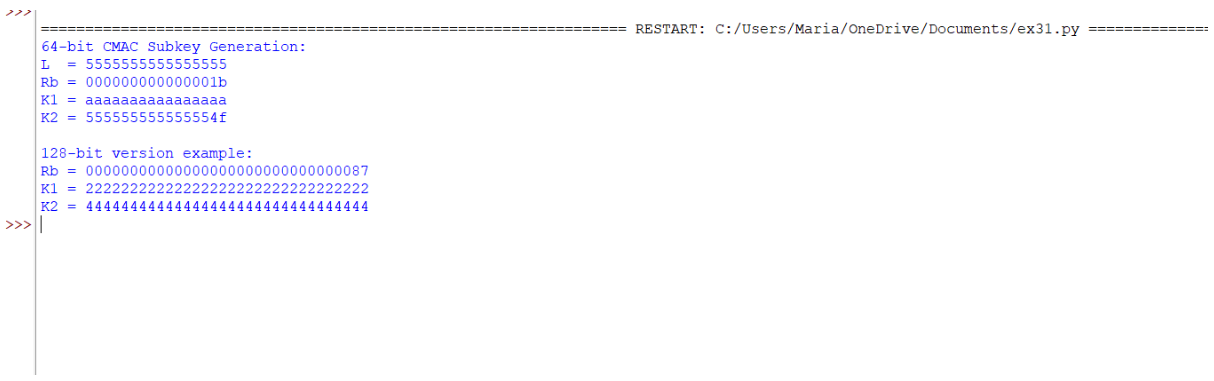
31. Write a python program for subkey generation in CMAC, it states that the block cipher is applied to the block that consists entirely of 0 bits. The first subkey is derived from the resulting string by a left shift of one bit and, conditionally, by XORing a constant that depends on the block size. The second subkey is derived in the same manner from the first subkey. a. What constants are needed for block sizes of 64 and 128 bits? b. How the left shift and XOR accomplishes the desired result.

```
# ---------------------------------------------------------
# CMAC Subkey Generation (for 64-bit or 128-bit block size)
# No crypto libraries. A dummy block cipher is used.
# ---------------------------------------------------------
```

Code:

```
def left_shift_one_bit(block):
    """Shift an entire block (bytes) left by 1 bit"""
    out = bytearray(len(block))
    carry = 0
    for i in reversed(range(len(block))):
        new_carry = (block[i] & 0x80) >> 7
        out[i] = ((block[i] << 1) & 0xFF) | carry
        carry = new_carry
    return bytes(out), carry
def xor_bytes(a, b):
    return bytes(x ^ y for x, y in zip(a, b))
# Dummy block cipher: E(K, M) = M XOR K (just for demonstration)
def E(K, M):
    return xor_bytes(K, M)
def generate_cmac_subkeys(K, block_size_bits):
    block_size_bytes = block_size_bits // 8
    # CMAC constants
    if block_size_bits == 64:
        Rb = bytes([0]*7 + [0x1B])   # last byte = 0x1B
    elif block_size_bits == 128:
        Rb = bytes([0]*15 + [0x87])  # last byte = 0x87
    else:
        raise ValueError("Use 64 or 128 bits")
 # Step 1: L = E(K, 0^b)
    zero_block = bytes(block_size_bytes)
    L = E(K, zero_block)
# Step 2: K1 = L << 1; if MSB(L) = 1, K1 = (L << 1) XOR Rb
    K1, carry1 = left_shift_one_bit(L)
    if carry1:
        K1 = xor_bytes(K1, Rb)
 # Step 3: K2 = K1 << 1; if MSB(K1) = 1, K2 = (K1 << 1) XOR Rb
    K2, carry2 = left_shift_one_bit(K1)
    if carry2:
        K2 = xor_bytes(K2, Rb)
 return K1, K2, L, Rb
# ---------------------------
# Example usage:
# ---------------------------
K = b"\x55" * 8         # 64-bit key
K1, K2, L, Rb = generate_cmac_subkeys(K, 64)
print("64-bit CMAC Subkey Generation:")
print("L  =", L.hex())
print("Rb =", Rb.hex())
print("K1 =", K1.hex())
print("K2 =", K2.hex())

print("\n128-bit version example:")

K128 = b"\x11" * 16

K1_128, K2_128, L128, Rb128 = generate_cmac_subkeys(K128, 128)

print("Rb =", Rb128.hex())

print("K1 =", K1_128.hex())

print("K2 =", K2_128.hex())
```



```
>>>
==================================================================== RESTART: C:/Users/Maria/OneDrive/Documents/ex31.py =============
64-bit CMAC Subkey Generation:
L  = 5555555555555555
Rb = 000000000000001b
K1 = aaaaaaaaaaaaaaaa
K2 = 555555555555554f

128-bit version example:
Rb = 00000000000000000000000000000087
K1 = 22222222222222222222222222222222
K2 = 44444444444444444444444444444444
>>>
```