20. Write a python program for ECB mode, if there is an error in a block of the transmitted ciphertext, only the corresponding plaintext block is affected. However, in the CBC mode, this error propagates. For example, an error in the transmitted C1 obviously corrupts P1 and P2. a. Are any blocks beyond P2 affected? b. Suppose that there is a bit error in the source version of P1. Through how many ciphertext blocks is this error propagated? What is the effect at the receiver?

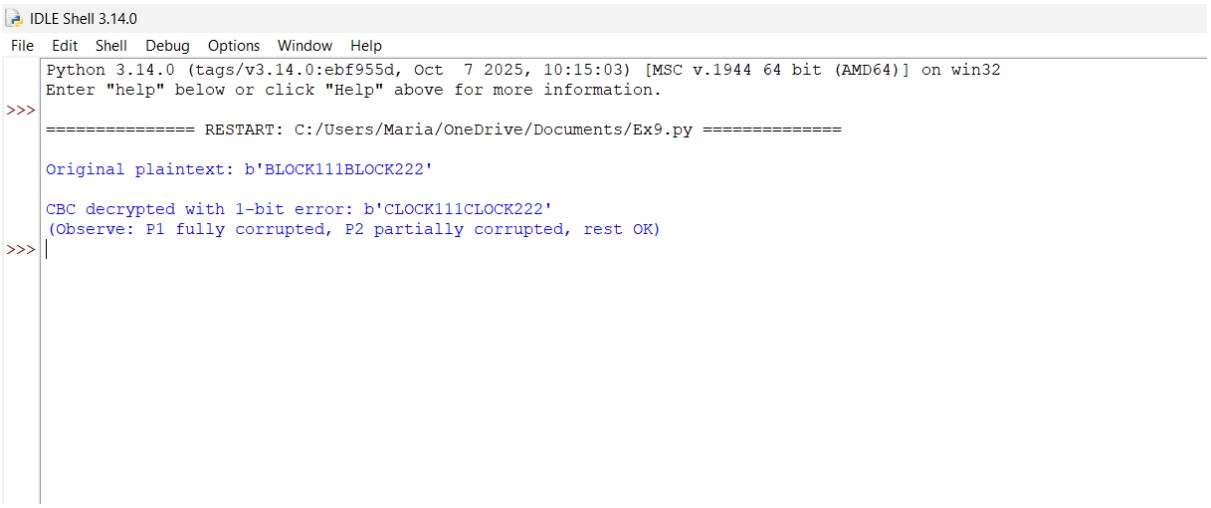
```
# ------------------------------------------------------
# Simple ECB and CBC implementation (no crypto module)
# Toy block cipher: XOR with key (8-byte blocks)
# ------------------------------------------------------

BLOCK = 8
```

Code:

```
def xor_block(a, b):
    """XOR two byte strings of equal length."""
    return bytes([x ^ y for x, y in zip(a, b)])
def pad(data):
    """Simple padding to multiple of 8 bytes."""
    while len(data) % BLOCK != 0:
        data += b"_"
    return data
# ------------------------------------------------------
# ECB Mode
# ------------------------------------------------------
def ecb_encrypt(plaintext, key):
    plaintext = pad(plaintext)
    blocks = [plaintext[i:i+BLOCK] for i in range(0, len(plaintext), BLOCK)]
    return b"".join([xor_block(b, key) for b in blocks])
```

```
def ecb_decrypt(ciphertext, key):
    blocks = [ciphertext[i:i+BLOCK] for i in range(0, len(ciphertext), BLOCK)]
    return b"".join([xor_block(b, key) for b in blocks])
# ------------------------------------------------------
# CBC Mode
# ------------------------------------------------------
def cbc_encrypt(plaintext, key, iv):
    plaintext = pad(plaintext)
    blocks = [plaintext[i:i+BLOCK] for i in range(0, len(plaintext), BLOCK)]
    ciphertext = b""
    prev = iv
for b in blocks:
        x = xor_block(b, prev)
        c = xor_block(x, key)  # encrypt
        ciphertext += c
        prev = c
 return ciphertext
def cbc_decrypt(ciphertext, key, iv):
    blocks = [ciphertext[i:i+BLOCK] for i in range(0, len(ciphertext), BLOCK)]

    plaintext = b""
    prev = iv

    for c in blocks:
        x = xor_block(c, key)  # decrypt
        p = xor_block(x, prev)
        plaintext += p
```

```
        prev = c

    return plaintext

# ----------------------------------------------------
# DEMONSTRATION OF ERROR PROPAGATION
# ----------------------------------------------------
key = b"ABCDEFGH"  # 8 bytes
iv  = b"12345678"

plaintext = b"BLOCK111BLOCK222"

print("\nOriginal plaintext:", plaintext)

# Encrypt normally
cbc_ct = cbc_encrypt(plaintext, key, iv)

# Introduce 1-byte error in first ciphertext block
cbc_ct_error = bytearray(cbc_ct)
cbc_ct_error[0] ^= 0x01   # flip one bit
cbc_ct_error = bytes(cbc_ct_error)

# Decrypt corrupted ciphertext
dec_with_error = cbc_decrypt(cbc_ct_error, key, iv)

print("\nCBC decrypted with 1-bit error:", dec_with_error)
print("(Observe: P1 fully corrupted, P2 partially corrupted, rest OK)")
```


IDLE Shell 3.14.0
File Edit Shell Debug Options Window Help
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/Ex9.py ==========
Original plaintext: b'BLOCK111BLOCK222'
CBC decrypted with 1-bit error: b'CLOCK111CLOCK222'
(Observe: P1 fully corrupted, P2 partially corrupted, rest OK)
>>>