34. Write a python program for ECB, CBC, and CFB modes, the plaintext must be a sequence of one or more complete data blocks (or, for CFB mode, data segments). In other words, for these three modes, the total number of bits in the plaintext must be a positive multiple of the block (or segment) size. One common method of padding, if needed, consists of a 1 bit followed by as few zero bits, possibly none, as are necessary to complete the final block. It is considered good practice for the sender to pad every message, including messages in which the final message block is already complete. What is the motivation for including a padding block when padding is not needed?

```
# ------------------------------
# ECB, CBC, and CFB modes demo
# Block size = 8 bytes (64 bits) for simplicity
# ------------------------------
```

Code:

```
def pad(plaintext, block_size):
    """ Pad plaintext to a multiple of block_size using 1-bit + zeros
    """
    n = len(plaintext)
    pad_len = block_size - (n % block_size)
    if pad_len == 0:
        pad_len = block_size  # always add padding even if exact multiple
    return plaintext + bytes([0x80] + [0]*(pad_len-1))
def split_blocks(data, block_size):
    return [data[i:i+block_size] for i in range(0, len(data), block_size)]
# --- Simple XOR block cipher for demonstration ---
def simple_encrypt_block(block, key):
    return bytes([b ^ k for b, k in zip(block, key)])
def simple_decrypt_block(block, key):
    return bytes([b ^ k for b, k in zip(block, key)])
# --- ECB Mode ---
def ecb_encrypt(plaintext, key):
```

```
    blocks = split_blocks(plaintext, len(key))
    return b''.join([simple_encrypt_block(b, key) for b in blocks])
def ecb_decrypt(ciphertext, key):
    blocks = split_blocks(ciphertext, len(key))
    return b''.join([simple_decrypt_block(b, key) for b in blocks])
# --- CBC Mode ---
def cbc_encrypt(plaintext, key, iv):
    blocks = split_blocks(plaintext, len(key))
    ciphertext = []
    prev = iv
    for b in blocks:
        x = bytes([a ^ b for a, b in zip(b, prev)])
        c = simple_encrypt_block(x, key)
        ciphertext.append(c)
        prev = c
    return b''.join(ciphertext)
def cbc_decrypt(ciphertext, key, iv):
    blocks = split_blocks(ciphertext, len(key))
    plaintext = []
    prev = iv
    for c in blocks:
        x = simple_decrypt_block(c, key)
        p = bytes([a ^ b for a, b in zip(x, prev)])
        plaintext.append(p)
        prev = c
    return b''.join(plaintext)
# --- CFB Mode ---
def cfb_encrypt(plaintext, key, iv):
```

```
    blocks = split_blocks(plaintext, len(key))
    ciphertext = []
    prev = iv
    for b in blocks:
        s = simple_encrypt_block(prev, key)
        c = bytes([a ^ b for a, b in zip(s, b)])
        ciphertext.append(c)
        prev = c
    return b''.join(ciphertext)
def cfb_decrypt(ciphertext, key, iv):
    blocks = split_blocks(ciphertext, len(key))
    plaintext = []
    prev = iv
    for c in blocks:
        s = simple_encrypt_block(prev, key)
        p = bytes([a ^ b for a, b in zip(s, c)])
        plaintext.append(p)
        prev = c
    return b''.join(plaintext)
# ----------------------------
# Example Usage
# ----------------------------
plaintext = b"HELLO ECB CBC CFB MODES"
block_size = 8  # bytes
key = b"KEYBLOCK"  # 8 bytes key
iv = b"\x00"*8     # simple IV
# Pad plaintext
padded = pad(plaintext, block_size)
```

```
# Encrypt & Decrypt
ecb_ct = ecb_encrypt(padded, key)
ecb_pt = ecb_decrypt(ecb_ct, key)
cbc_ct = cbc_encrypt(padded, key, iv)
cbc_pt = cbc_decrypt(cbc_ct, key, iv)
cfb_ct = cfb_encrypt(padded, key, iv)
cfb_pt = cfb_decrypt(cfb_ct, key, iv)
print("Original plaintext:", plaintext)
print("Padded plaintext  :", padded)
print("\n--- ECB ---")
print("Ciphertext:", ecb_ct)
print("Decrypted :", ecb_pt)
print("\n--- CBC ---")
print("Ciphertext:", cbc_ct)
print("Decrypted :", cbc_pt)
print("\n--- CFB ---")
print("Ciphertext:", cfb_ct)
print("Decrypted :", cfb_pt)
```


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/ex34.py ==========
Original plaintext: b'HELLO ECB CBC CFB MODES'
Padded plaintext : b'HELLO ECB CBC CFB MODES\x80'
--- ECB ---
Ciphertext: b'\x03\x00\x15\x0e\x03o\x06\x08\te\x1a\x00\x0fo\x00\r\te\x14\r\x08\n\x10\xcb'
Decrypted : b'HELLO ECB CBC CFB MODES\x80'
--- CBC ---
Ciphertext: b'\x03\x00\x15\x0e\x03o\x06\x08\ne\x0f\x0e\x0c\x00\x06\x05\x03\x00\x1b\x03\x04\n\x16\xce'
Decrypted : b'HELLO ECB CBC CFB MODES\x80'
--- CFB ---
Ciphertext: b'\x03\x00\x15\x0e\x03o\x06\x08\ne\x0f\x0e\x0c\x00\x06\x05\x03\x00\x1b\x03\x04\n\x16\xce'
Decrypted : b'HELLO ECB CBC CFB MODES\x80'
>>>