program for encryption in the cipher block chaining (CBC) mode using an algorithm stronger than DES. 3DES is a good candidate. Both of which follow from the definition of CBC. Which of the two would you choose: a. For security? b. For performance?

# Simple CBC Encryption Demo (3DES-like, no external libraries)

def xor\_bytes(a, b):

"""XOR two byte strings of equal length"""

return bytes([x ^ y for x, y in zip(a, b)])

def simple\_des\_block(data, key):

"""A toy 'DES-like' block cipher (not real DES, for demo only)"""

return bytes([(b + key[i % len(key)]) % 256 for i, b in enumerate(data)])

def pad(data):

pad\_len = 8 - (len(data) % 8)

return data + bytes([pad\_len]) \* pad\_len

def unpad(data):

pad\_len = data[-1]

return data[:-pad\_len]

def cbc\_encrypt(plaintext, key, iv):

plaintext = pad(plaintext)

ciphertext = b""

prev\_block = iv

for i in range(0, len(plaintext), 8):

block = plaintext[i:i+8]

block = xor\_bytes(block, prev\_block)

enc\_block = simple\_des\_block(block, key)

ciphertext += enc\_block

prev\_block = enc\_block

return ciphertext

def cbc\_decrypt(ciphertext, key, iv):

plaintext = b""

prev\_block = iv

for i in range(0, len(ciphertext), 8):

block = ciphertext[i:i+8]

dec\_block = simple\_des\_block(block, key) # same function (symmetric)

dec\_block = xor\_bytes(dec\_block, prev\_block)

plaintext += dec\_block

prev\_block = block

return unpad(plaintext)

# --- Main Program ---

key = b"simpleky" # 8 bytes

iv = b"initvect" # 8 bytes

plaintext = input("Enter plaintext: ").encode()

ciphertext = cbc\_encrypt(plaintext, key, iv)

print("\nCiphertext (hex):", ciphertext.hex().upper())

decrypted = cbc\_decrypt(ciphertext, key, iv)

print("Decrypted text:", decrypted.decode())

