13) Write a C 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?

PROGRAM:-

```
from Crypto.Cipher import DES3
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
import base64

def encrypt_3des_cbc(plaintext, key, iv=None):
    """
    Encrypts plaintext using 3DES in CBC mode.

    Args:
        plaintext (str or bytes): The data to encrypt
        key (bytes): The encryption key (16 or 24 bytes)
        iv (bytes, optional): Initialization vector (8 bytes)

    Returns:
        dict: {'ciphertext': base64 encoded ciphertext, 'iv': base64 encoded iv}
    """
    if isinstance(plaintext, str):
        plaintext = plaintext.encode('utf-8')

    # Generate random IV if not provided
    if iv is None:
        iv = get_random_bytes(8)

    # Create cipher object
```

```
    cipher = DES3.new(key, DES3.MODE_CBC, iv)

    # Pad and encrypt
    padded_data = pad(plaintext, DES3.block_size)
    ciphertext = cipher.encrypt(padded_data)

    return {
        'ciphertext': base64.b64encode(ciphertext).decode('utf-8'),
        'iv': base64.b64encode(iv).decode('utf-8')
    }

def decrypt_3des_cbc(ciphertext, key, iv):
    """
    Decrypts ciphertext using 3DES in CBC mode.

    Args:
        ciphertext (str or bytes): The encrypted data (base64 encoded)
        key (bytes): The decryption key
        iv (bytes): Initialization vector (base64 encoded)

    Returns:
        bytes: The decrypted plaintext
    """
    if isinstance(ciphertext, str):
        ciphertext = base64.b64decode(ciphertext.encode('utf-8'))
    if isinstance(iv, str):
        iv = base64.b64decode(iv.encode('utf-8'))

    # Create cipher object
    cipher = DES3.new(key, DES3.MODE_CBC, iv)
```

```
    # Decrypt and unpad
    decrypted_data = cipher.decrypt(ciphertext)
    plaintext = unpad(decrypted_data, DES3.block_size)

    return plaintext

# Example usage
if __name__ == "__main__":
    # 3DES key must be either 16 or 24 bytes long
    # For 3-key 3DES (most secure), use 24 bytes
    key = get_random_bytes(24)
    plaintext = "This is a secret message to be encrypted with 3DES in CBC mode!"

    # Encrypt
    encrypted = encrypt_3des_cbc(plaintext, key)
    print("Encrypted:")
    print(f"IV: {encrypted['iv']}")
    print(f"Ciphertext: {encrypted['ciphertext']}")

    # Decrypt
    decrypted = decrypt_3des_cbc(encrypted['ciphertext'], key, encrypted['iv'])
    print("\nDecrypted:")
    print(decrypted.decode('utf-8'))
```

OUTPUT:-

```
Encrypted:
IV: Q3Qui+80pQM=
Ciphertext: pArwsz5m1oOtmVw+qvzXd4zfk9t52LWyubM3EbbaPBf3WhTWGypT6H7IsJ0K0LFixAzWXPs1ZXZL2CLdCkNfKA==

Decrypted:
This is a secret message to be encrypted with 3DES in CBC mode!
```