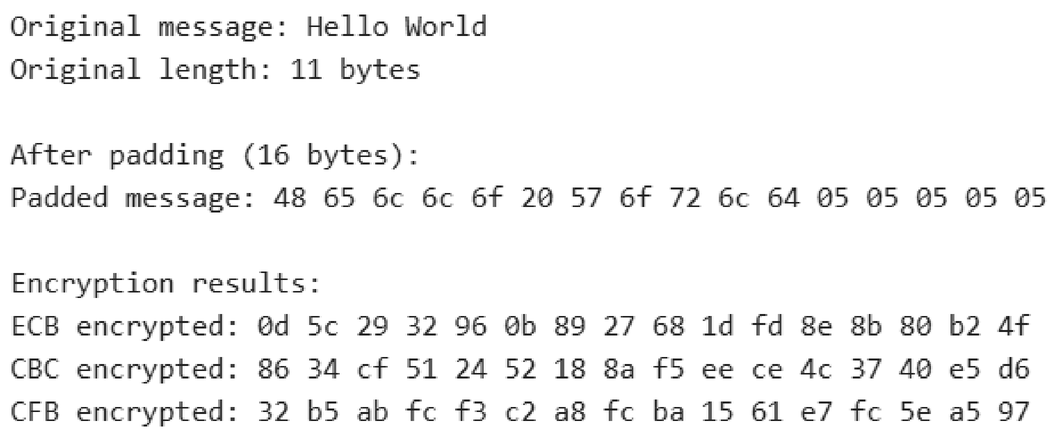
14) Write a C program for ECB, CBC, and CFB modes, the plaintext must be a sequence of one or more complete data blocks (or, for CFB mode, data segments). In other words, for these three modes, the total number of bits in the plaintext must be a positive multiple of the block (or segment) size. One common 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?

PRGRAM:-

```
from Crypto.Cipher import DES3

from Crypto.Util.Padding import pad

from Crypto.Random import get_random_bytes

import binascii

BLOCK_SIZE = 8  # DES block size is 8 bytes

def print_hex(data, label):

    hex_data = binascii.hexlify(data).decode('utf-8')

    formatted = ' '.join([hex_data[i:i+2] for i in range(0, len(hex_data), 2)])

    print(f"{label}: {formatted}")

key = get_random_bytes(24)  # 3DES requires 16 or 24 byte key

iv = get_random_bytes(BLOCK_SIZE)

message = b"Hello World"

print(f"Original message: {message.decode('utf-8')}")

print(f"Original length: {len(message)} bytes")

padded_msg = pad(message, BLOCK_SIZE)

print(f"\nAfter padding ({len(padded_msg)} bytes):")

print_hex(padded_msg, "Padded message")

ecb_cipher = DES3.new(key, DES3.MODE_ECB)

cbc_cipher = DES3.new(key, DES3.MODE_CBC, iv)

cfb_cipher = DES3.new(key, DES3.MODE_CFB, iv, segment_size=8)

ecb_encrypted = ecb_cipher.encrypt(padded_msg)

cbc_encrypted = cbc_cipher.encrypt(padded_msg)

cfb_encrypted = cfb_cipher.encrypt(padded_msg)

print("\nEncryption results:")

print_hex(ecb_encrypted, "ECB encrypted")
print_hex(cbc_encrypted, "CBC encrypted")

print_hex(cfb_encrypted, "CFB encrypted")
```

OUTPUT:-



```
Original message: Hello World
Original length: 11 bytes

After padding (16 bytes):
Padded message: 48 65 6c 6c 6f 20 57 6f 72 6c 64 05 05 05 05 05

Encryption results:
ECB encrypted: 0d 5c 29 32 96 0b 89 27 68 1d fd 8e 8b 80 b2 4f
CBC encrypted: 86 34 cf 51 24 52 18 8a f5 ee ce 4c 37 40 e5 d6
CFB encrypted: 32 b5 ab fc f3 c2 a8 fc ba 15 61 e7 fc 5e a5 97
```