Exercise 6

Question 1.

Pseudocode:

```
Function AES_Encrypt(text,key):
    state = ConvertToStateMatrix(plaintext)
    roundKeys = KeyExpansion(key)

    state = AddRoundKey(state,roundKeys)

    for round from  1 to 9:
        state = SubBytes(state)
        state = ShiftRows(state)
        state = MixColumns(state)
        state = AddRoundKey(state, roundKeys[round])

    state = SubBytes(state)
    state = ShiftRows(state)
    state = AddRoundKey(state,roundKeys)

    result_encrypted= ConvertStateMatrixToOutput(state)
    return result_encrypted
```

Functions Explanation:

ConvertToStateMatrix() = takes the 16 bytes of the text and arranges them into a 4 x 4 matrix
SubBytes() = substitutes every byte in the data with another byte using a predefined table called "S-Box"
ShiftRows() = this shifts the bytes in each row of the data matrix to the left by a predefined number of positions
MixColumns() = mixes the bytes within each column of the data matrix by using a special kind of multiplication and addition
AddRoundKey() = combines the current data with some of the secret key by using a simple XOR operation
KeyExpansion() = the original key is being expended to multiple ones, one for each encryption round
ConvertStateMatrixToOutput() = takes the final 4 x 4 matrix and converts it back into the 16 byte text as an encrypted massage now

Question 2.

Pseudocode:

```
Function AES_Decrypt(encrypted_text,key):
    state = ConvertToStateMatrix(encrypted_text)
    roundkeys = KeyExpansion(key)

    state = AddRoundKey(state, roundKeys[10])
```

```
for round from 9 to 1:
    state = InvShiftRows(state)
    state = InvSubBytes(state)
    state = AddRoundKey(state, roundKeys)
    state = InvMixColumns(state)

state = InvShiftRows(state)
state = InvSubBytes(state)
state = AddRoundKey(state, roundKeys)

decrypted_text=ConvertStateMatrixToOutput(state)
return decrypted_text
```

Function Explanation:

ConvertToStateMatrix() = converts de 16 byte encrypted text into a 4 x 4 matrix
KeyExpansion() = expands the original key into all rounds KeyExpansion
InvShiftRows() = shifts each row of the matrix to the right
InvSubBytes() = replaces each byte in the matrix using the inverse S-Box
InvShiftRows() = shifts each row of the matrix to the right by the row index(number)
InvMixColumns() = performs a revers multiplication on each column
AddRoundKey() = combines the state with the round key
ConvertStateMatrixToOutput() = converts the final matrix back to the 16 bytes decrypted text

Question 3.

CBC = Cipher Block Chaining

Before encrypting a block we apply the operation XOR with a previous encrypted block only then we encrypt the block.
The first block will be mixed with the initialization vector.

To decrypt each block of the original massage we decrypt the current ciphered block and then apply XOR with the previous ciphered block.

We will assume that the ciphertext, CIP[err], gets an error it will cause the plaintext block which has the same index , PB[err], and the next one , PB[err + 1], to be broken.
Starting from PB[err + 2] there will be no problem caused to the decryption.

Question 4.

OFB = Output Feedback Mode

OFB turns a block cipher into a stream cipher, instead of encrypting the message block directly, it encrypts an initialization vector , IV, to create a keystream block.
Then encrypts the result again to get the next keystream block.
So each keystream block has the operation XOR applied with the plaintext.

We will say that the ciphertext block with the index n, CIP[n] got corrupted.
Then the plaintext block with index n, P[n], will be wrong because the operation XOR has been applied on the corrupted CIP[n].
After this, the next plaintext boxes with indexes of n+1,n+2, etc will not be affected because the keystream is generated independently of the ciphertext and the XOR operation can be used correctly.