Security Lab
Lab Assignment No. 4

Aim: Design and Implement a Play Fair cipher.

The **Playfair cipher** is a manual symmetric encryption technique and was the first literal diagram substitution cipher. The technique encrypts pairs of letters, instead of single letters as in the simple substitution cipher and rather more complex Vigenère cipher systems then in use. The Playfair is thus significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it. The frequency analysis of bigrams is possible, but considerably more difficult. With 600 possible bigrams rather than the 26 possible monograms (single symbols, usually letters in this context), a considerably larger cipher text is required in order to be useful.

First, a plaintext message is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter. Let us say we want to encrypt the message "hide money". It will be written as:

HI DE MO NE YZ

Rules of encryption:

If both the letters are in the same column, take the letter below each one (going back to the top if at the bottom)

T	U	O	R	I
A	L	S	B	C
D	E	F	G	H
K	M	N	P	Q
V	W	X	Y	Z

('H' and 'I' are in the same column, hence take the letter below them to replace. HI → QC)
If both letters are in the same row, take the letter to the right of each one (going back to the left if at the farthest right)

T	U	O	R	I
A	L	S	B	C
D	E	F	G	H
K	M	N	P	Q
V	W	X	Y	Z

('D' and 'E' are in the same row, hence take the letter to the right of them to replace. DE → EF)
If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

Using these rules, the result of the encryption of 'hide money' with the key of 'tutorials' would be:

QC EF NU MF ZV

Decrypting the Playfair cipher is as simple as doing the same process in reverse. Receiver has the same key and can create the same key table, and then decrypt any messages made using that key. The Playfair cipher was used mainly to protect important, yet non-critical secrets, as it is quick to use and requires no special equipment.

Algorithm:

STEP 1: Read the plain text from the user.
STEP 2: Read the keyword from the user.
STEP 3: Arrange the keyword without duplicates in a 5*5 matrix in the row order and fill the remaining cells with missed out letters in alphabetical order. Note that ' i' and ' j' take the same cell.
STEP 4: Group the plain text in pairs and match the corresponding corner letters by forming a rectangular grid.
STEP 5: Display the obtained cipher text.

Code:

```
# Function to initialize the matrix
def find_position(key_matrix,letter):
    x, y = 0, 0

    # Making the matrix
    for i in range(5):
        for j in range(5):
            if key_matrix[i][j] == letter:
                x, y = i, j

    return x,y

# Function to make a 5x5 matrix
```

```
def getKeywordMatrix(plaintext, keyword):
    matrix = []
    alpha = 'abcdefghijklmnopqrstuvwxyz'
    alpha = alpha.upper()

    # Add alphabets to the matrix
    for e in keyword.upper():
        if e not in matrix:
            matrix.append(e)

    # If the alphabet is J, then we don't need to append it to matrix
    for e in alpha:
        if e not in matrix:
            if e == 'J':
                continue
            else:
                matrix.append(e)

    keyword_matrix=[]

    for e in range(5):
        keyword_matrix.append('')

    # Make a final keyword matrix
    for i in range(0, 5):
        keyword_matrix[i] = matrix[i*5:i*5+5]

    return keyword_matrix

# Function to encrypt a plain text using Play Fair Cipher
def encrypt(plaintext, keyword_matrix):
    message=[]

    # Remove all spaces in the plain text
    for  plain in plaintext:
        message.append(plain)
    for unused in message:
        if unused == ' ':
```

```
        message.remove(' ')

    # If the plain text is of even length
    print('Plain Text:',''.join(message))

    i = 0
    l = int(len(message)/2)

    for both in range(l):
       if message[i] == message[i+1]:
          message.insert(i+1, 'Z')
       i=i+2

    # If the plain text is of odd length, adding Z in the end
    if len(message)%2 == 1:
       message.append("Z")

    i, new_message = 0, []

    for x in range(int(len(message)/2)):
       new_message.append(message[i:i+2])
       i = i+2

    print("\nDigraphs: ", new_message)

    # To print the encrypted text using the keyword matrix
    cipher_matrix = []

    for e in new_message:
       p1,q1 = find_position(keyword_matrix,e[0])
       p2,q2 = find_position(keyword_matrix,e[1])

       if p1 == p2:
          cipher_matrix.append(keyword_matrix[p1][(q1+1)%5])
          cipher_matrix.append(keyword_matrix[p1][(q2+1)%5])
       elif q1 == q2:
          cipher_matrix.append(keyword_matrix[(p1+1)%5][q1])
          cipher_matrix.append(keyword_matrix[(p2+1)%5][q2])
```

```
        else:
            cipher_matrix.append(keyword_matrix[p1][q2])
            cipher_matrix.append(keyword_matrix[p2][q1])

    print("\nCipher text: ", ''.join(cipher_matrix))

    return cipher_matrix

# Function to decrypt an encrypted text using Play Fair Cipher
def decrypt(cipher_matrix, keyword_matrix):
    i = 0
    new_cipher_matrix = []

    for x in range(int(len(cipher_matrix)/2)):
        new_cipher_matrix.append(cipher_matrix[i:i+2])
        i = i+2

    omessage=[]

    for e in new_cipher_matrix:
        p1,q1 = find_position(keyword_matrix,e[0])
        p2,q2 = find_position(keyword_matrix,e[1])
        if p1 == p2:
            omessage.append(keyword_matrix[p1][q1-1])
            omessage.append(keyword_matrix[p1][q2-1])
        elif q1 == q2:
            omessage.append(keyword_matrix[p1-1][q1])
            omessage.append(keyword_matrix[p2-1][q2])
        else:
            omessage.append(keyword_matrix[p1][q2])
            omessage.append(keyword_matrix[p2][q1])

    print("Original Message: ", ''.join(omessage))

plaintext = input("Enter the plaintext: ")
keyword = input("Enter the keyword: ")
plaintext = plaintext.upper()
keyword = keyword.upper()
```

```
keywordMatrix = getKeywordMatrix(plaintext, keyword)

# Function call for encryption
encryptedMessage = encrypt(plaintext, keywordMatrix)

# Function call for decryption
decrypt(encryptedMessage, keywordMatrix)
```

Output:

```
PS D:\III Year Engineering\CNS Lab Experiments> & "C:/Users/Ninad Rao/AppData/Local/Programs/Python/Python39/python.exe" "d:/III Year Engineering/CNS Lab Experiments/assignment3.py"
Enter the plaintext: My name is Ninad
Enter the keyword: leader
Plain Text: MYNAMEISNINAD

Digraphs:  [['M', 'Y'], ['N', 'A'], ['M', 'E'], ['I', 'S'], ['N', 'I'], ['N', 'A'], ['D', 'Z']]

Cipher text:  NXMDKAMPOKMDRY
Original Message:  MYNAMEISNINADZ
```

Conclusion: Thus we understood how to design and implement a Play Fair cipher.