Practical-4

Aim : Alice wants to send some confidential information to Bob over a secure network. Prepare a key matrix for the given key and apply encryption on the plain text (key is your surname & plain text is your name).

```
import string

def prepare_key(key):
    key = key.upper().replace('J', 'I')
    unique_chars = sorted(set(key), key=key.find)
    alphabet = list(string.ascii_uppercase.replace('J', ''))
    matrix = unique_chars + [letter for letter in alphabet if letter not in
unique_chars]
    playfair_matrix = [matrix[i:i+5] for i in range(0, 25, 5)]
    return playfair_matrix

def find_position(char, matrix):
    for i, row in enumerate(matrix):
        if char in row:
            return i, row.index(char)
    return -1, -1

def process_pair(char1, char2, matrix, operation):
    row1, col1 = find_position(char1, matrix)
    if row1 == -1:
        row1, col1 = find_position('X', matrix)
    row2, col2 = find_position(char2, matrix)
    if row2 == -1:
        row2, col2 = find_position('X', matrix)
    if row1 == row2:
        return matrix[row1][(col1 + operation) % 5] + matrix[row2][(col2 +
operation) % 5]
    elif col1 == col2:
        return matrix[(row1 + operation) % 5][col1] + matrix[(row2 +
operation) % 5][col2]
    else:
        return matrix[row1][col2] + matrix[row2][col1]

def encrypt(plaintext, matrix):
    ciphertext = ""
    plaintext = plaintext.upper().replace('J', 'I').replace(' ', '')
    pairs = [(plaintext[i], plaintext[i + 1] if i + 1 < len(plaintext) else
'X') for i in range(0, len(plaintext), 2)]

    for pair in pairs:
        ciphertext += process_pair(pair[0], pair[1], matrix, 1)
```

```
    return ciphertext, pairs

def decrypt(ciphertext, matrix):
    plaintext = ""
    pairs = [(ciphertext[i], ciphertext[i + 1]) for i in range(0,
len(ciphertext), 2)]

    for pair in pairs:
        plaintext += process_pair(pair[0], pair[1], matrix, -1)

    return plaintext, pairs

key_input = input("Enter the key: ")
plaintext_input = input("Enter the plaintext: ")

playfair_key_matrix = prepare_key(key_input)
encrypted_text_output, encryption_pairs_output = encrypt(plaintext_input,
playfair_key_matrix)

print("\nEncryption Output:")
print(f"Plaintext Pairs: {encryption_pairs_output}")
print(f"Encrypted Text: {encrypted_text_output}")
print("\nKey Matrix:")
for row in playfair_key_matrix:
    print(row)

encrypted_text_input = input("\nEnter the encrypted text for decryption: ")
decrypted_text_output, decryption_pairs_output = decrypt(encrypted_text_input,
playfair_key_matrix)
print("\nDecryption Output:")
print(f"Encrypted Text: {encrypted_text_input}")
print(f"Plaintext: {decrypted_text_output}")
print(f"Decryption Pairs: {decryption_pairs_output}")
```

Output: -

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS C:\Users\panjw> & C:/Users/panjw/AppData/Local/Programs/Python/Python311/p
Enter the key: PANJWANI
Enter the plaintext: ANIKET

Encryption Output:
Plaintext Pairs: [('A', 'N'), ('I', 'K'), ('E', 'T')]
Encrypted Text: NINLFS

Key Matrix:
['P', 'A', 'N', 'I', 'W']
['B', 'C', 'D', 'E', 'F']
['G', 'H', 'K', 'L', 'M']
['O', 'Q', 'R', 'S', 'T']
['U', 'V', 'X', 'Y', 'Z']

Enter the encrypted text for decryption: NINLFS

Decryption Output:
Encrypted Text: NINLFS
Plaintext: ANIKET
Decryption Pairs: [('N', 'I'), ('N', 'L'), ('F', 'S')]
PS C:\Users\panjw>
```