Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 2

Date of Performance : 25 February 2023 **Date of Submission:** 26 February 2023

SAP Id: 60004200132 **Name :** Ayush Jain

Div: B **Batch :** B3

Aim of Experiment

Design and Implement Playfair Cipher. Create two functions Encrypt() and Decrypt().

Theory / Algorithm / Conceptual Description:

The Playfair cipher works by first constructing a 5x5 matrix of letters (excluding any duplicates) based on a keyword provided by the user. The keyword is then used to populate the matrix, with any remaining letters filled in from the alphabet in order. For example, if the keyword is "PLAYFAIR", the matrix would be:

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

To encrypt a message, the plaintext is broken up into pairs of letters and then processed one pair at a time. If a pair of letters is the same, a filler letter (usually X) is inserted between them. Then, the position of each pair of letters in the matrix is determined, and a set of rules is applied to determine the corresponding ciphertext pair.

The rules for determining the ciphertext pair are as follows:

- If the two letters are in the same row, replace each letter with the letter to its right, wrapping around to the beginning of the row if necessary.
- If the two letters are in the same column, replace each letter with the letter below it, wrapping around to the top of the column if necessary.
- If the two letters are neither in the same row nor the same column, replace each letter with the letter in the same row but in the column of the other letter.

For example, if the plaintext pair is "HI", the ciphertext pair would be "NL" because "H" is in the third row and first column of the matrix, and "I" is in the fourth row and third column, so we replace "H" with the letter in the same row but in the column of "I" (which is "N"), and replace "I" with the letter in the same row but in the column of "H" (which is "L").

The Playfair cipher has some security weaknesses, such as the fact that the key space is relatively small compared to other modern ciphers, and that the use of a fixed 5x5 matrix makes it vulnerable to known plaintext attacks. However, it was considered a strong cipher at the time it was introduced, and it remains an interesting historical example of a poly-graphic substitution cipher.

Program

```
alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',
             'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def get_position(letter, matrix):
    for i in range(5):
        for j in range(5):
            if matrix[i][j] == letter:
                return i, j

def digraphs(plain_text):
    text = []
    i = 0
    j = 1
    n = len(plain_text)
    while i < n and j < n:
        if plain_text[i] != plain_text[j]:
            text.append([plain_text[i], plain_text[j]])
            i += 2
            j += 2
        else:
            text.append([plain_text[i], "x"])
            i += 1
            j += 1
    if i < n:
        text.append([plain_text[i], "x"])
    return text

def encrypt(matrix, text):
    encrypted_text = ""
```

```
    print(text)
    for i in range(len(text)):
        x1, y1 = get_position(text[i][0], matrix)
        x2, y2 = get_position(text[i][1], matrix)
        if x1 == x2:
            # same row
            encrypted_text += matrix[x1][(y1+1) % 5]
            encrypted_text += matrix[x2][(y2+1) % 5]
        elif y1 == y2:
            # same column
            encrypted_text += matrix[(x1+1) % 5][y1]
            encrypted_text += matrix[(x2+1) % 5][y2]
        else:
            # intersection
            encrypted_text += matrix[x1][y2]
            encrypted_text += matrix[x2][y1]
    return encrypted_text


def decrypt(matrix, text):
    encrypted_text = ""
    print(text)
    for i in range(len(text)):
        x1, y1 = get_position(text[i][0], matrix)
        x2, y2 = get_position(text[i][1], matrix)
        if x1 == x2:
            # same row
            encrypted_text += matrix[x1][(y1-1) % 5]
            encrypted_text += matrix[x2][(y2-1) % 5]
        elif y1 == y2:
            # same column
            encrypted_text += matrix[(x1-1) % 5][y1]
            encrypted_text += matrix[(x2-1) % 5][y2]
        else:
            # intersection
            encrypted_text += matrix[x1][y2]
            encrypted_text += matrix[x2][y1]
    return encrypted_text

def generate_matrix(key):
    temp_list = []
    for i in key:
        if i not in temp_list:
            temp_list.append(i)
    for i in alphabets:
        if i not in temp_list:
            temp_list.append(i)
```

```
    matrix = []
    for i in range(5):
        matrix.append(temp_list[:5])
        temp_list = temp_list[5:]
    return matrix

if __name__ == "__main__":
    key = input("Enter the key : ")
    plain_text = input("Enter plain text : ")
    matrix = generate_matrix(key)
    digraphs_text = digraphs(plain_text)
    cipher_text = encrypt(matrix, digraphs_text)
    c_digraphs_text = digraphs(cipher_text)
    print(decrypt(matrix, c_digraphs_text))
```

Output

```
Enter the key : hehe
Enter plain text : ayushhere
[['a', 'y'], ['u', 's'], ['h', 'x'], ['h', 'e'], ['r', 'e']]
[['b', 'x'], ['q', 't'], ['a', 'v'], ['e', 'a'], ['w', 'f']]
ayushxhere
```