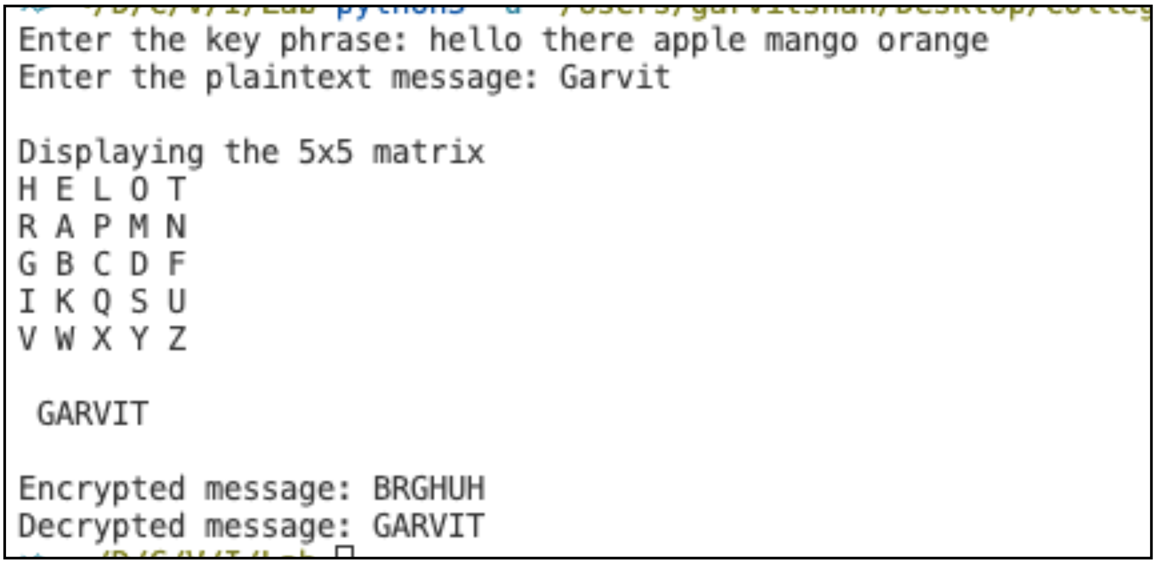
Lab Assignment 4

U21CS089 | Garvit Shah

1) Design and implement a program to perform encryption and decryption using the Playfair Cipher with both a 5x5 and a 6x6 matrix. Consider the following inputs for the program.
a) Input1 – key phrase, Generate key matrix
b) Input2 – plain text, arrange into valid digrams
c) Output – print Key matrix, print plain txt, and encrypted output
d) For decryption – Input2 is cipher txt, no need to rearrange, output
plain txt, remove padding

5X5 Playfair

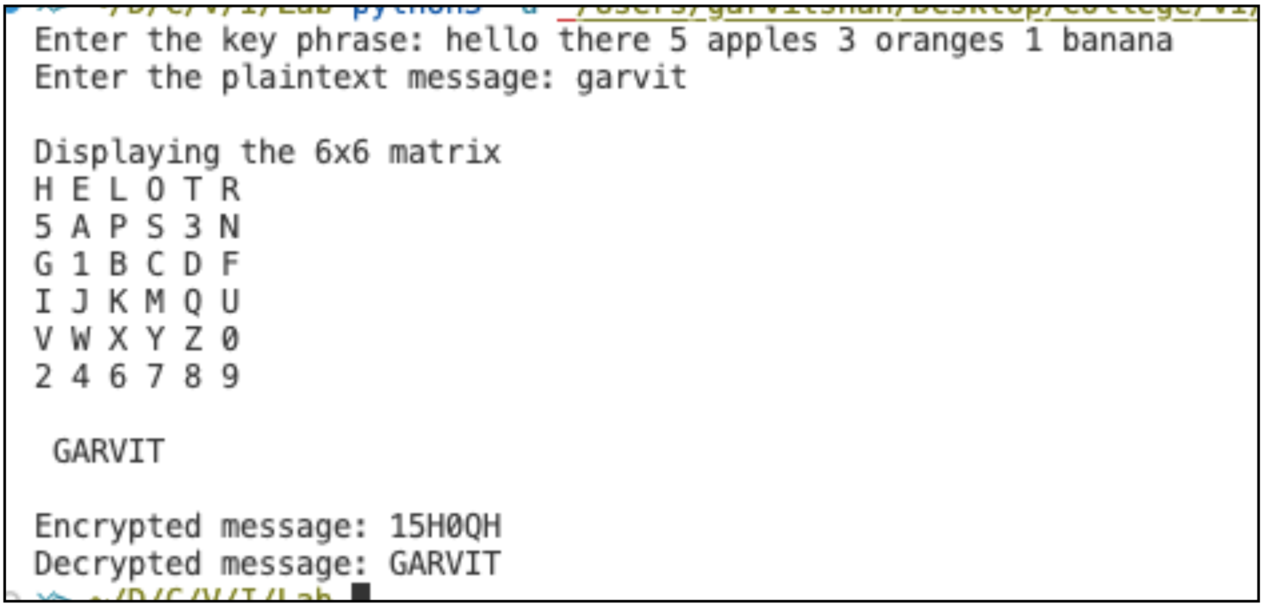
```
def generate_playfair_matrix(key):
    # Matrix Generation
    key = key.replace(" ", "").upper()
    alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
    matrix = []

    # Matrix filling
    for char in key:
        if char not in matrix and char in alphabet:
            matrix.append(char)
    for char in alphabet:
        if char not in matrix:
            matrix.append(char)

    # 1D to 2D
    playfair_matrix = [matrix[i:i+5] for i in range(0, 25, 5)]

    return playfair_matrix

def preprocess_input(text):
    # Preprocess the input text: convert to uppercase, remove
spaces, replace 'J' with 'I'
    text = text.upper().replace(" ", "")
    text = text.replace("J", "I")

    # Insert a dummy character 'X' between repeated characters and
at the end if necessary
    processed_text = ""
    i = 0
    while i < len(text):
        processed_text += text[i]
        if i < len(text) - 1 and text[i] == text[i + 1]:
            processed_text += 'X'
            # i += 1
        i += 1
    if len(processed_text) % 2 != 0:
        processed_text += 'X'
    return processed_text

def find_char_position(matrix, char):
    # Find the position of a character in the Playfair matrix
    for i, row in enumerate(matrix):
        if char in row:
            return (i, row.index(char))
    return None

def playfair_encrypt(plaintext, key):
    # Generate the Playfair matrix from the key
    playfair_matrix = generate_playfair_matrix(key)

    # Preprocess the plaintext
    plaintext = preprocess_input(plaintext)

    #Displaying Stuff
    print("\nDisplaying the 5x5 matrix")
    for l in playfair_matrix:
        for i in l:
            print(i, end =" ")
        print()

    print("\n", plaintext)

    # Encrypt the plaintext using the Playfair cipher
    ciphertext = ""
    for i in range(0, len(plaintext), 2):
        char1, char2 = plaintext[i], plaintext[i + 1]
        row1, col1 = find_char_position(playfair_matrix, char1)
        row2, col2 = find_char_position(playfair_matrix, char2)
        if row1 == row2:
            # Same row: shift to the right (circular)
            ciphertext += playfair_matrix[row1][(col1 + 1) % 5] +
playfair_matrix[row2][(col2 + 1) % 5]
        elif col1 == col2:
            # Same column: shift downwards (circular)
            ciphertext += playfair_matrix[(row1 + 1) % 5][col1] +
playfair_matrix[(row2 + 1) % 5][col2]
        else:
            # Different row and column: form rectangle and take
opposite corners
            ciphertext += playfair_matrix[row1][col2] +
playfair_matrix[row2][col1]
    return ciphertext

def playfair_decrypt(ciphertext, key):
    # Generate the Playfair matrix from the key
    playfair_matrix = generate_playfair_matrix(key)

    # Decrypt the ciphertext using the Playfair cipher
    plaintext = ""
    for i in range(0, len(ciphertext), 2):
        char1, char2 = ciphertext[i], ciphertext[i + 1]
        row1, col1 = find_char_position(playfair_matrix, char1)
        row2, col2 = find_char_position(playfair_matrix, char2)
        if row1 == row2:
            # Same row: shift to the left (circular)
            plaintext += playfair_matrix[row1][(col1 - 1) % 5] +
playfair_matrix[row2][(col2 - 1) % 5]
        elif col1 == col2:
            # Same column: shift upwards (circular)
            plaintext += playfair_matrix[(row1 - 1) % 5][col1] +
playfair_matrix[(row2 - 1) % 5][col2]
        else:
            # Form rectangle and take opposite corners
            plaintext += playfair_matrix[row1][col2] +
playfair_matrix[row2][col1]
    return ''.join(plaintext.split('X'))

def main():
    # Input key phrase and plain text message from the user
    key_phrase = input("Enter the key phrase: ")
    plaintext = input("Enter the plaintext message: ")

    # Encrypt the plaintext using the Playfair cipher
    ciphertext = playfair_encrypt(plaintext, key_phrase)
    print("\nEncrypted message:", ciphertext)

    # Decrypt the ciphertext using the Playfair cipher
    decrypted_text = playfair_decrypt(ciphertext, key_phrase)
    print("Decrypted message:", decrypted_text)

if __name__ == "__main__":
    main()
```



```
Enter the key phrase: hello there apple mango orange
Enter the plaintext message: Garvit

Displaying the 5x5 matrix
H E L O T
R A P M N
G B C D F
I K Q S U
V W X Y Z

 GARVIT

Encrypted message: BRGHUH
Decrypted message: GARVIT
```

6X6 Playfair

```
def generate_playfair_matrix(key):
    # Matrix Generation
    key = key.replace(" ", "").upper()
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
    matrix = []

    # Matrix filling
    for char in key:
        if char not in matrix and char in alphabet:
            matrix.append(char)
    for char in alphabet:
        if char not in matrix:
            matrix.append(char)

    # 1D to 2D
    playfair_matrix = [matrix[i:i+6] for i in range(0, 36, 6)]

    return playfair_matrix

def preprocess_input(text):
    # Preprocess the input text: convert to uppercase, remove
spaces, replace 'J' with 'I'
    text = text.upper().replace(" ", "")
    # text = text.replace("J", "I")

    # Insert a dummy character 'X' between repeated characters and
at the end if necessary
    processed_text = ""
    i = 0
    while i < len(text):
        processed_text += text[i]
        if i < len(text) - 1 and text[i] == text[i + 1]:
            processed_text += 'X'
            # i += 1
        i += 1
    if len(processed_text) % 2 != 0:
        processed_text += 'X'
    return processed_text

def find_char_position(matrix, char):
    # Find the position of a character in the Playfair matrix
    for i, row in enumerate(matrix):
        if char in row:
            return (i, row.index(char))
    return None

def playfair_encrypt(plaintext, key):
    # Generate the Playfair matrix from the key
    playfair_matrix = generate_playfair_matrix(key)
    # Preprocess the plaintext
    plaintext = preprocess_input(plaintext)

    #Displaying Stuff
    print("\nDisplaying the 6x6 matrix")
    for l in playfair_matrix:
        for i in l:
            print(i, end =" ")
        print()

    print("\n", plaintext)

    # Encrypt the plaintext using the Playfair cipher
    ciphertext = ""
    for i in range(0, len(plaintext), 2):
        char1, char2 = plaintext[i], plaintext[i + 1]
        row1, col1 = find_char_position(playfair_matrix, char1)
        row2, col2 = find_char_position(playfair_matrix, char2)
        if row1 == row2:
            # Same row: shift to the right (circular)
            ciphertext += playfair_matrix[row1][(col1 + 1) % 6] +
playfair_matrix[row2][(col2 + 1) % 6]
        elif col1 == col2:
            # Same column: shift downwards (circular)
            ciphertext += playfair_matrix[(row1 + 1) % 6][col1] +
playfair_matrix[(row2 + 1) % 6][col2]
        else:
            # Different row and column: form rectangle and take
opposite corners
            ciphertext += playfair_matrix[row1][col2] +
playfair_matrix[row2][col1]
    return ciphertext

def playfair_decrypt(ciphertext, key):
    # Generate the Playfair matrix from the key
    playfair_matrix = generate_playfair_matrix(key)

    # Decrypt the ciphertext using the Playfair cipher
    plaintext = ""
    for i in range(0, len(ciphertext), 2):
        char1, char2 = ciphertext[i], ciphertext[i + 1]
        row1, col1 = find_char_position(playfair_matrix, char1)
        row2, col2 = find_char_position(playfair_matrix, char2)
        if row1 == row2:
            # Same row: shift to the left (circular)
            plaintext += playfair_matrix[row1][(col1 - 1) % 6] +
playfair_matrix[row2][(col2 - 1) % 6]
        elif col1 == col2:
            # Same column: shift upwards (circular)
            plaintext += playfair_matrix[(row1 - 1) % 6][col1] +
playfair_matrix[(row2 - 1) % 6][col2]
        else:
            # Form rectangle and take opposite corners
            plaintext += playfair_matrix[row1][col2] +
playfair_matrix[row2][col1]
    return ''.join(plaintext.split('X'))

def main():
    # Input key phrase and plain text message from the user
    key_phrase = input("Enter the key phrase: ")
    plaintext = input("Enter the plaintext message: ")

    # Encrypt the plaintext using the Playfair cipher
    ciphertext = playfair_encrypt(plaintext, key_phrase)
    print("\nEncrypted message:", ciphertext)

    # Decrypt the ciphertext using the Playfair cipher
    decrypted_text = playfair_decrypt(ciphertext, key_phrase)
    print("Decrypted message:", decrypted_text)

if __name__ == "__main__":
    main()
```



```
 ~/D/C/V/I/Lab python3 -u /Users/garvitshah/Desktop/College/VI/
Enter the key phrase: hello there 5 apples 3 oranges 1 banana
Enter the plaintext message: garvit

Displaying the 6x6 matrix
H E L O T R
5 A P S 3 N
G 1 B C D F
I J K M Q U
V W X Y Z 0
2 4 6 7 8 9

 GARVIT

Encrypted message: 15H0QH
Decrypted message: GARVIT
 ~/D/C/V/I/Lab
```