Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

Name: Tushar Panchal

En.No: 21162101014

Sub: INS (INFORMATION SECURITY)

Branch: CBA

Batch:61

----------PRACTICAL 04----------

❖ Question :

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).

✓ Source Code :

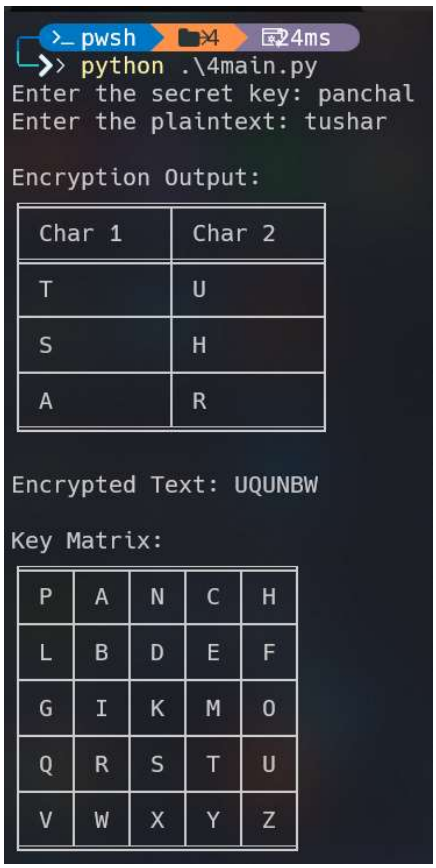
```
import string
from tabulate import tabulate

def generate_key_matrix(secret_key):
    secret_key = secret_key.upper().replace('J', 'I')
    unique_key_chars = sorted(set(secret_key), key=secret_key.find)
    alphabet = list(string.ascii_uppercase.replace('J', ''))

    # Fill the remaining matrix with unique letters from the alphabet
(excluding 'J')
    matrix = unique_key_chars + [char for char in alphabet if char not
in unique_key_chars]

    # Create a 5x5 matrix
    key_matrix = [matrix[i:i+5] for i in range(0, 25, 5)]
    return key_matrix

def find_char_position(char, key_matrix):
    for i, row in enumerate(key_matrix):
        if char in row:
```

```
            return i, row.index(char)
    return -1, -1

def encrypt(plain_text, key_matrix):
    cipher_text = ""
    plain_text = plain_text.upper().replace('J', 'I')
    char_pairs = []

    for i in range(0, len(plain_text), 2):
        char1 = plain_text[i]
        char2 = plain_text[i+1] if i+1 < len(plain_text) else 'X'
        char_pairs.append((char1, char2))

        row1, col1 = find_char_position(char1, key_matrix)
        if row1 == -1:
            row1, col1 = find_char_position('X', key_matrix)

        row2, col2 = find_char_position(char2, key_matrix)
        if row2 == -1:
            row2, col2 = find_char_position('X', key_matrix)

        if row1 == row2:
            cipher_text += key_matrix[row1][(col1 + 1) % 5] +
key_matrix[row2][(col2 + 1) % 5]
        elif col1 == col2:
            cipher_text += key_matrix[(row1 + 1) % 5][col1] +
key_matrix[(row2 + 1) % 5][col2]
        else:
            cipher_text += key_matrix[row1][col2] +
key_matrix[row2][col1]

    return cipher_text, char_pairs

def decrypt(cipher_text, key_matrix):
    plain_text = ""
    char_pairs = []

    for i in range(0, len(cipher_text), 2):
        char1 = cipher_text[i]
        char2 = cipher_text[i+1]
        char_pairs.append((char1, char2))

        row1, col1 = find_char_position(char1, key_matrix)
        row2, col2 = find_char_position(char2, key_matrix)

        if row1 == row2:
            plain_text += key_matrix[row1][(col1 - 1) % 5] +
key_matrix[row2][(col2 - 1) % 5]
```

```
        elif col1 == col2:
            plain_text += key_matrix[(row1 - 1) % 5][col1] +
key_matrix[(row2 - 1) % 5][col2]
        else:
            plain_text += key_matrix[row1][col2] +
key_matrix[row2][col1]

    # Remove trailing 'X' if it was a padding character
    if plain_text[-1] == 'X':
        plain_text = plain_text[:-1]

    return plain_text, char_pairs

def display_matrix(matrix):
    print(tabulate(matrix, tablefmt="fancy_grid"))

def display_pairs(pairs):
    print(tabulate(pairs, headers=["Char 1", "Char 2"],
tablefmt="fancy_grid"))

# Taking input from the user for encryption
secret_key = input("Enter the secret key: ")
plain_text = input("Enter the plaintext: ")

# Encrypting the plaintext
key_matrix = generate_key_matrix(secret_key)
cipher_text, char_pairs = encrypt(plain_text, key_matrix)

# Displaying encryption output
print("\nEncryption Output:")
display_pairs(char_pairs)
print(f"\nEncrypted Text: {cipher_text}")

# Displaying encryption output and key matrix
print("\nKey Matrix:")
display_matrix(key_matrix)

# Taking input from the user for decryption
cipher_text_input = input("\nEnter the encrypted text for decryption:
")

# Decrypting the encrypted text
decrypted_text, decryption_pairs = decrypt(cipher_text_input,
key_matrix)

# Displaying decryption output and key matrix
print("\nDecryption Output:")
print(f"Encrypted Text: {cipher_text_input}")
```

```
display_pairs(decryption_pairs)
print(f"\nPlaintext: {decrypted_text}")

# Check if the decrypted text matches the original plaintext
if decrypted_text.upper() == plain_text.upper():
    print("Decryption Successful!")
else:
    print("Decryption Failed!")

print("\nKey Matrix:")
display_matrix(key_matrix)
```

✓ **Output :**

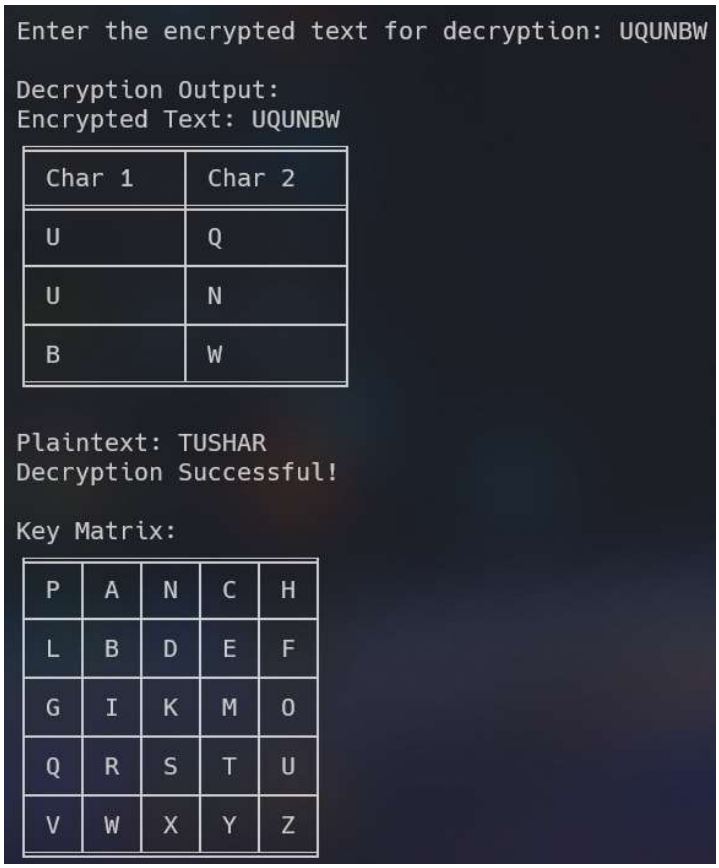


```
pwsh 4 4ms
> python .\4main.py
Enter the secret key: panchal
Enter the plaintext: tushar

Encryption Output:
```

Char 1	Char 2
T	U
S	H
A	R

```
Encrypted Text: UQUNBW

Key Matrix:
```

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



```
Enter the encrypted text for decryption: UQUNBW

Decryption Output:
Encrypted Text: UQUNBW
```

Char 1	Char 2
U	Q
U	N
B	W

```
Plaintext: TUSHAR
Decryption Successful!

Key Matrix:
```

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