Name: Abdurrahman Qureshi

Roll No: 242466

Practical No: 2

Date Of Performance: 14/07/2025

Aim: To encrypt and decrypt messages using different poly-alphabetic cipher techniques and frequency analysis

Playfair Cipher:

CODE:

```
alphabets: str = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
row_1: list = []
row_2: list = []
row_3: list = []
row_4: list = []
row_5: list = []
matrix = [row_1, row_2, row_3, row_4, row_5]
encrypted_pairs:list = []
decrypted_pairs:list = []


def playfair_matrix(key: str):
    for i in key:
        if len(row_1) != 5:
            if i == "I" or i == "J": row_1.append("I/J")
            else: row_1.append(i)

        elif len(row_2) != 5:
            if i == "I" or i == "J": row_2.append("I/J")
            else: row_2.append(i)
```

```
        elif len(row_3) != 5:
            if i == "I" or i == "J": row_3.append("I/J")
            else: row_3.append(i)

        elif len(row_4) != 5:
            if i == "I" or i == "J": row_4.append("I/J")
            else: row_4.append(i)

        elif len(row_5) != 5:
            if i == "I" or i == "J": row_5.append("I/J")
            else: row_5.append(i)

    for i in range(65, 91):
        char = chr(i)

        if char in key: continue
        elif char in row_1: continue
        elif char in row_2: continue
        elif char in row_3: continue
        elif char in row_4: continue
        elif char in row_5: continue

        if char == "I" or char == "J":
            if ("I" in key or "J" in key or "I/J" in key):continue
            elif ("I" in row_1 or "J" in row_1 or "I/J" in row_1):continue
            elif ("I" in row_2 or "J" in row_2 or "I/J" in row_2):continue
            elif ("I" in row_3 or "J" in row_3 or "I/J" in row_3):continue
            elif ("I" in row_4 or "J" in row_4 or "I/J" in row_4):continue
            elif ("I" in row_5 or "J" in row_5 or "I/J" in row_5):continue
            else:char = "I/J"

        if len(row_1) < 5: row_1.append(char)
        elif len(row_2) < 5: row_2.append(char)
        elif len(row_3) < 5: row_3.append(char)
        elif len(row_4) < 5: row_4.append(char)
        elif len(row_5) < 5: row_5.append(char)

    print()
    print("Playfair Matrix:")
    print()
```

```
    print(row_1)
    print(row_2)
    print(row_3)
    print(row_4)
    print(row_5)
    print()


def get_text_pairs(text_pairs: str, type_pairs: list):
    i = 0
    while(i < len(text_pairs)):
        prev_char = text_pairs[i - 1]
        current_char = text_pairs[i]
        next_char = text_pairs[i + 1]

        if current_char == "X" and i == len(text_pairs - 1):
            type_pairs.append(text_pairs[i] + "Z")
            break

        if i == len(text_pairs) - 1:
            type_pairs.append(text_pairs[i] + "X")
            break

        if current_char == next_char:
            type_pairs.append(text_pairs[i] + "X")
            i = i + 1
        elif current_char == prev_char:
            type_pairs.append(text_pairs[i] + text_pairs[i + 1])
            i = i + 2
        else:
            type_pairs.append(text_pairs[i] + text_pairs[i + 1])
            i = i + 2

    print("Text Pairs:")
    print(type_pairs, "\n")


def get_row_index(char: str):
    if char == "I" or char == "J":
        char = "I/J"
    if char in row_1:
```

```
            return {
                "index": row_1.index(char),
                "row_num": 1
            }
        if char in row_2:
            return {
                "index": row_2.index(char),
                "row_num": 2
            }
        if char in row_3:
            return {
                "index": row_3.index(char),
                "row_num": 3
            }
        if char in row_4:
            return {
                "index": row_4.index(char),
                "row_num": 4
            }
        if char in row_5:
            return {
                "index": row_5.index(char),
                "row_num": 5
            }


def playfair_cipher_encrypt(plain_text: str, key: str):
    if len(plain_text) == 0: return
    if len(key) == 0: return

    plain_text = plain_text.upper()
    plain_text = "".join(plain_text.split())

    key = key.upper()
    key = "".join(key.split())

    for i in plain_text:
        if i == " ":
            print(f"Frequency of ' ' in plain text is {(len(plain_text) /
plain_text.count(i))}%")
        else:
```

```
            print(f"Frequency of {i} in plain text is {(len(plain_text) /
plain_text.count(i))}%")

    playfair_matrix(key)

    get_text_pairs(plain_text, encrypted_pairs)

    encrypted_text = ""

    for i in encrypted_pairs:
        first_info = get_row_index(i[0])
        second_info = get_row_index(i[1])

        first_row, first_col = first_info["row_num"]-1, first_info["index"]
        second_row, second_col = second_info["row_num"]-1, second_info["index"]

        if first_row == second_row:
            encrypted_text += (matrix[first_row][(first_col + 1) % 5])
            encrypted_text += (matrix[second_row][(second_col + 1) % 5])
        elif first_col == second_col:
            encrypted_text += (matrix[(first_row + 1) % 5][first_col])
            encrypted_text += (matrix[(second_row + 1) % 5][second_col])
        else:
            encrypted_text += (matrix[first_row][second_col])
            encrypted_text += (matrix[second_row][first_col])

    return encrypted_text


def playfair_cipher_decrypt(cipher_text: str, key: str):
    if len(cipher_text) == 0: return
    if len(key) == 0: return

    cipher_text = cipher_text.upper()
    cipher_text = "".join(cipher_text.split())

    key = key.upper()
    key = "".join(key.split())

    get_text_pairs(cipher_text, decrypted_pairs)
```

```
    decrypted_text = ""

    for i in decrypted_pairs:
        first_info = get_row_index(i[0])
        second_info = get_row_index(i[1])

        first_row, first_col = first_info["row_num"]-1, first_info["index"]
        second_row, second_col = second_info["row_num"]-1, second_info["index"]

        if first_row == second_row:
            decrypted_text += (matrix[first_row][(first_col - 1) % 5])
            decrypted_text += (matrix[second_row][(second_col - 1) % 5])
        elif first_col == second_col:
            decrypted_text += (matrix[(first_row - 1) % 5][first_col])
            decrypted_text += (matrix[(second_row - 1) % 5][second_col])
        else:
            decrypted_text += (matrix[first_row][second_col])
            decrypted_text += (matrix[second_row][first_col])

    return decrypted_text


original_message = "The key is hidden under the door"
original_key = "domestic"

print("\nOriginal Message: ", original_message)

encrypted_message = playfair_cipher_encrypt(original_message, original_key)
print("Encrypted Message: ", encrypted_message)

print()
decrypted_message = playfair_cipher_decrypt(encrypted_message, original_key)
print("Decrypted Message: ", decrypted_message)
```

OUTPUT:

```
Abdurrahman Qureshi@DESKTOP-H2RV5MQ MINGW64 /d/Degree/SEM 5/CNS/Experiments/EXP2 (master)
$ py playfair_cipher.py

Original Message:  The key is hidden under the door
Frequency of T in plain text is 13.0%
Frequency of H in plain text is 8.666666666666666%
Frequency of E in plain text is 5.2%
Frequency of K in plain text is 26.0%
Frequency of E in plain text is 5.2%
Frequency of Y in plain text is 26.0%
Frequency of I in plain text is 13.0%
Frequency of S in plain text is 26.0%
Frequency of H in plain text is 8.666666666666666%
Frequency of I in plain text is 13.0%
Frequency of D in plain text is 6.5%
Frequency of D in plain text is 6.5%
Frequency of E in plain text is 5.2%
Frequency of N in plain text is 13.0%
Frequency of U in plain text is 26.0%
Frequency of N in plain text is 13.0%
Frequency of D in plain text is 6.5%
Frequency of E in plain text is 5.2%
Frequency of R in plain text is 13.0%
Frequency of T in plain text is 13.0%
Frequency of H in plain text is 8.666666666666666%
Frequency of E in plain text is 5.2%
Frequency of D in plain text is 6.5%
Frequency of O in plain text is 13.0%
Frequency of O in plain text is 13.0%
Frequency of R in plain text is 13.0%
```

```
Playfair Matrix:

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

Text Pairs:
['TH', 'EK', 'EY', 'IS', 'HI', 'DX', 'DE', 'NU', 'ND', 'ER', 'TH', 'ED', 'OX', 'OR']

Encrypted Message:  CFARAEBOGCMVOSPNVTAYCFSOMWEP

Text Pairs:
['CF', 'AR', 'AE', 'BO', 'GC', 'MV', 'OS', 'PN', 'VT', 'AY', 'CF', 'SO', 'MW', 'EP']

Decrypted Message:  THEKEYI/JSHI/JDXDENUNDERTHEDOXOR
```