Name: Moksha Dave
Roll no: 22BCP366
Group: G10

LAB Assignment 3

Theory:

The Rail Fence cipher is a transposition cipher that rearranges the plaintext into a zigzag pattern across multiple rows. The message is written diagonally down and up across the rows. After writing out the entire message, each row is read sequentially to produce the ciphertext. For example, with plaintext "HELLO" and 3 rows, the zigzag pattern would look like:

```
H . . . O
. E . L .
. . L . .
```

Reading row by row, the ciphertext is "HOELL". Decryption reverses the process by reconstructing the zigzag pattern and reading the message diagonally.

Code:

Rail Fence Cipher:

```
def rail_fence_encrypt(plain_text, num_rails):
    if num_rails == 1:
        return plain_text

    rails = ['' for _ in range(num_rails)]
    rail = 0
    direction = 1

    for char in plain_text:
        rails[rail] += char
        rail += direction
        if rail == 0 or rail == num_rails - 1:
            direction *= -1

    cipher_text = ''.join(rails)
    return cipher_text
```

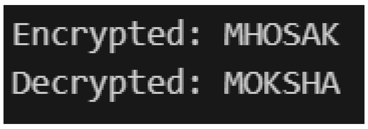
Name: Moksha Dave
Roll no: 22BCP366
Group: G10

```
def rail_fence_decrypt(cipher_text, num_rails):
    if num_rails == 1:
        return cipher_text

    rails = [['' for _ in range(len(cipher_text))] for _ in range(num_rails)]
    rail = 0
    direction = 1

    for i in range(len(cipher_text)):
        rails[rail][i] = '*'
        rail += direction
        if rail == 0 or rail == num_rails - 1:
            direction *= -1

    index = 0
    for r in range(num_rails):
        for c in range(len(cipher_text)):
            if rails[r][c] == '*':
                rails[r][c] = cipher_text[index]
                index += 1

    rail = 0
    direction = 1
    result = []
    for i in range(len(cipher_text)):
        result.append(rails[rail][i])
        rail += direction
        if rail == 0 or rail == num_rails - 1:
            direction *= -1

    return ''.join(result)
```

Name: Moksha Dave
Roll no: 22BCP366
Group: G10

```
plain_text = "MOKSHA"
num_rails = 3


encrypted_text = rail_fence_encrypt(plain_text, num_rails)
print("Encrypted:", encrypted_text)


decrypted_text = rail_fence_decrypt(encrypted_text, num_rails)
print("Decrypted:", decrypted_text)
```

Output:



```
Encrypted: MHOSAK
Decrypted: MOKSHA
```

Rail Fence with column shift (my modification):

Theory:

I combined the Rail Fence cipher with columnar transposition. Encryption involves writing plaintext in a zigzag pattern across multiple rails, then reading characters row by row to form an intermediate string. I then split this string into columns based on the length of a key and shuffled the columns according to the alphabetical order of the key to create the ciphertext. Decryption reverses this process by rearranging the columns back to their original order and reconstructing the zigzag pattern to retrieve the plaintext.

Code:

```
def encrypt(text, rails, key):
    rail_matrix = [['.' for _ in range(len(text))] for _ in range(rails)]
    row, direction = 0, 1


    for i in range(len(text)):
        rail_matrix[row][i] = text[i]
        if row == 0:
```

Name: Moksha Dave
Roll no: 22BCP366
Group: G10

```
                direction = 1
            elif row == rails - 1:
                direction = -1
            row += direction

    intermediate = ""
    for i in range(rails):
        for j in range(len(text)):
            if rail_matrix[i][j] != '.':
                intermediate += rail_matrix[i][j]

    key_pairs = sorted([(key[i], i) for i in range(len(key))])

    columns = ["" for _ in range(len(key))]
    for i in range(len(intermediate)):
        columns[i % len(key)] += intermediate[i]

    encrypted_text = ""
    for _, idx in key_pairs:
        encrypted_text += columns[idx]

    return encrypted_text

def decrypt(text, rails, key):
    key_pairs = sorted([(key[i], i) for i in range(len(key))])

    column_size = len(text) // len(key)
    extra_chars = len(text) % len(key)

    columns = ["" for _ in range(len(key))]
    index = 0
    for i in range(len(key)):
```

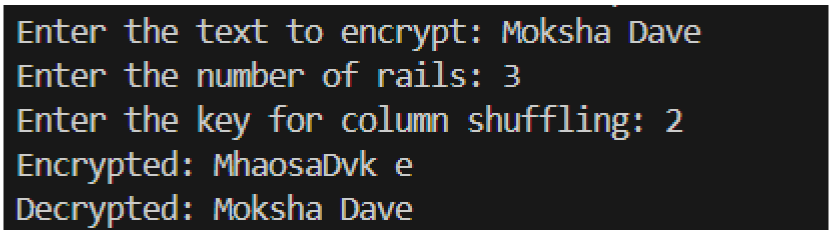
Name: Moksha Dave
Roll no: 22BCP366
Group: G10

```
        size = column_size + (1 if i < extra_chars else 0)
        columns[key_pairs[i][1]] = text[index:index + size]
        index += size

    intermediate = ""
    for i in range(len(text)):
        intermediate += columns[i % len(key)][i // len(key)]

    rail_matrix = [['.' for _ in range(len(intermediate))] for _ in range(rails)]
    row, direction = 0, 1

    for i in range(len(intermediate)):
        rail_matrix[row][i] = '*'
        if row == 0:
            direction = 1
        elif row == rails - 1:
            direction = -1
        row += direction

    index = 0
    for i in range(rails):
        for j in range(len(intermediate)):
            if rail_matrix[i][j] == '*':
                rail_matrix[i][j] = intermediate[index]
                index += 1

    decrypted_text = ""
    row, direction = 0, 1
    for i in range(len(intermediate)):
        decrypted_text += rail_matrix[row][i]
        if row == 0:
            direction = 1
```

Name: Moksha Dave
Roll no: 22BCP366
Group: G10

```
        elif row == rails - 1:
            direction = -1
        row += direction

    return decrypted_text


if __name__ == "__main__":
    text = input("Enter the text to encrypt: ")
    rails = int(input("Enter the number of rails: "))
    key = input("Enter the key for column shuffling: ")

    encrypted_text = encrypt(text, rails, key)
    print("Encrypted:", encrypted_text)

    decrypted_text = decrypt(encrypted_text, rails, key)
    print("Decrypted:", decrypted_text)
```

Output:



```
Enter the text to encrypt: Moksha Dave
Enter the number of rails: 3
Enter the key for column shuffling: 2
Encrypted: MhaosaDvk e
Decrypted: Moksha Dave
```