Name: Aarya Tiwari

Batch: B2

Roll number: 16010421119

Experiment : 2

**Column Transposition and Row Transposition**

Code:

def column\_transposition\_encrypt(plaintext, key): *# Format for key (2,1,3,4)*

    key\_order = [int(k) for k in key.split(",")]

    num\_rows = len(plaintext) // len(key\_order)

    if len(plaintext) % len(key\_order) != 0:

        num\_rows += 1

    grid = [[''] \* len(key\_order) for \_ in range(num\_rows)]

    idx = 0

    for row in range(num\_rows):

        for col in range(len(key\_order)):

            if idx < len(plaintext):

                grid[row][col] = plaintext[idx]

                idx += 1

    ciphertext = ""

    for col in key\_order:

        for row in range(num\_rows):

            ciphertext += grid[row][col - 1]

    return ciphertext

def column\_transposition\_decrypt(ciphertext, key): *# Format for key (2,1,3,4)*

    key\_order = [int(k) for k in key.split(",")]

    num\_rows = len(ciphertext) // len(key\_order)

    if len(ciphertext) % len(key\_order) != 0:

        num\_rows += 1

    inverse\_key\_order = [0] \* len(key\_order)

    for i, col in enumerate(key\_order):

        inverse\_key\_order[col - 1] = i + 1

*# Fill the grid with the ciphertext*

    grid = [[''] \* len(key\_order) for \_ in range(num\_rows)]

    idx = 0

    for col in inverse\_key\_order:

        for row in range(num\_rows):

            if idx < len(ciphertext):

                grid[row][col - 1] = ciphertext[idx]

                idx += 1

    plaintext = ""

    for row in range(num\_rows):

        for col in range(len(key\_order)):

            plaintext += grid[row][col]

    return plaintext

def row\_transposition\_encrypt(plaintext, key): *# Format for key (2134)*

    key\_order = [int(k) for k in key]

    num\_cols = len(key\_order)

    num\_rows = len(plaintext) // num\_cols

    if len(plaintext) % num\_cols != 0:

        num\_rows += 1

    grid = [[''] \* num\_cols for \_ in range(num\_rows)]

    idx = 0

    for row in range(num\_rows):

        for col in range(num\_cols):

            if idx < len(plaintext):

                grid[row][col] = plaintext[idx]

                idx += 1

    ciphertext = ""

    for row in key\_order:

        for col in range(num\_cols):

            ciphertext += grid[row - 1][col]

    return ciphertext

def row\_transposition\_decrypt(ciphertext, key): *# Format for key (2134)*

    num\_cols = len(key)

    num\_rows = len(ciphertext) // num\_cols

    if len(ciphertext) % num\_cols != 0:

        num\_rows += 1

    inverse\_key\_mapping = {int(k): i + 1 for i, k in enumerate(key)}

    grid = [[''] \* num\_cols for \_ in range(num\_rows)]

    idx = 0

    for row in inverse\_key\_mapping.values():

        for col in range(num\_cols):

            if idx < len(ciphertext):

                grid[row - 1][col] = ciphertext[idx]

                idx += 1

    plaintext = ""

    for col in range(num\_cols):

        for row in range(num\_rows):

            plaintext += grid[row][col]

    return plaintext

def double\_transposition\_encrypt(plaintext, key1, key2):

    row\_transposed = row\_transposition\_encrypt(plaintext, key1)

    ciphertext = column\_transposition\_encrypt(row\_transposed, key2)

    return ciphertext

def double\_transposition\_decrypt(ciphertext, key1, key2):

    row\_transposed = column\_transposition\_decrypt(ciphertext, key2)

    plaintext = row\_transposition\_decrypt(row\_transposed, key1)

    return plaintext

x = True

while(x):

    print("Welcome to Transposition Cipher Encryption and Decryption!!!")

    print("Enter 1 to perform Row Transposition Encryption")

    print("Enter 2 to perform Row Transposition Decryption")

    print("Enter 3 to perform Column Transposition Encryption")

    print("Enter 4 to perform Column Transposition Decryption")

    print("Enter 5 to perform Double Transposition Encryption")

    print("Enter 6 to perform Double Transposition Decryption")

    print("Enter 7 to exit")

    choice = int(input("Enter your choice: "))

    if(choice==1):

        plaintext = input("Enter the plaintext: ")

        key = input("Enter the key: ")

        print("Encrypted text: ", row\_transposition\_encrypt(plaintext, key))

    elif(choice==2):

        ciphertext = input("Enter the ciphertext: ")

        key = input("Enter the key: ")

        print("Decrypted text: ", row\_transposition\_decrypt(ciphertext, key))

    elif(choice==3):

        plaintext = input("Enter the plaintext: ")

        key = input("Enter the key: ")

        print("Encrypted text: ", column\_transposition\_encrypt(plaintext, key))

    elif(choice==4):

        ciphertext = input("Enter the ciphertext: ")

        key = input("Enter the key: ")

        print("Decrypted text: ", column\_transposition\_decrypt(ciphertext, key))

    elif(choice==5):

        plaintext = input("Enter the plaintext: ")

        key = input("Enter the key: ")

        print("Encrypted text: ", double\_transposition\_encrypt(plaintext, key))

    elif(choice==6):

        ciphertext = input("Enter the ciphertext: ")

        key = input("Enter the key: ")

        print("Decrypted text: ", double\_transposition\_decrypt(ciphertext, key))

    elif(choice==7):

        x = False

Output:

