**Transposition Ciphers**

**ROW TRANSPOSITION ENCRYPT**

**(WRITE PT HORIZONTALLY, READ CT VERTICALLY)**

**3 1 4 2**

**1 H E L L**

**2 O W O R**

**3 L D ! X**

**3142**

**EWDLRXHOLLO!**

**ROW TRANSPOSITION DECRYPT**

**(WRITE CT VERTICALLY, READ PT HORIZONTALLY)**

**3 1 4 2**

**1 H E L L**

**2 O W O R**

**3 L D ! X**

**3142**

**HELLOWORLD!X**

**COLUMN TRANSPOSITION ENCRYPT**

**(WRITE PT VERTICALLY, READ CT HORIZONTALLY)**

**1 2 3 4**

**3 H L O D**

**1 E O R !**

**2 L W L X**

**312**

**EOR!LWLXHLOD**

**COLUMN TRANSPOSITION DECRYPT**

**(WRITE CT HORIZONTALLY, READ PT VERTICALLY)**

**1 2 3 4**

**3 H L O D**

**1 E O R !**

**2 L W L X**

**312**

**HELLOWORLD!X**

**##############################################**

**DOUBLE:**

**ROW TRANSPOSITION ENCRYPT**

**(WRITE PT HORIZONTALLY, READ CT VERTICALLY)**

**3 1 4 2**

**1 H E L L**

**2 O W O R**

**3 L D ! X**

**3142**

**EWDLRXHOLLO!**

**COLUMN TRANSPOSITION ENCRYPT**

**(WRITE PT VERTICALLY, READ CT HORIZONTALLY)**

**1 2 3 4**

**3 E L H L**

**1 W R O O**

**2 D X L !**

**312**

**WROODXL!ELHL**

**COLUMN TRANSPOSITION DECRYPT**

**(WRITE CT HORIZONTALLY, READ PT VERTICALLY)**

**1 2 3 4**

**3 E L H L**

**1 W R O O**

**2 D X L !**

**312**

**EWDLRXHOLLO!**

**ROW TRANSPOSITION DECRYPT**

**(WRITE CT VERTICALLY, READ PT HORIZONTALLY)**

**3 1 4 2**

**1 H E L L**

**2 O W O R**

**3 L D ! X**

**3142**

**HELLOWORLD!X**

**import math**

**def encrypt\_row\_transposition(pt, key):**

**rows = int(math.ceil(len(pt)/len(key)))**

**cols = len(key)**

**matrix = [['' for \_ in range(cols)] for \_ in range(rows)]**

**index = 0**

**for r in range(rows):**

**for c in range(cols):**

**if index < len(pt):**

**matrix[r][c] = pt[index]**

**index += 1**

**else:**

**matrix[r][c] = 'X'**

**key\_order = sorted(range(len(key)), key=lambda k: key[k])**

**ct = ""**

**for col in key\_order:**

**for row in range(rows):**

**ct += matrix[row][col]**

**return ct**

**def decrypt\_row\_transposition(ct, key):**

**rows = int(math.ceil(len(ct)/len(key)))**

**cols = len(key)**

**matrix = [['' for \_ in range(cols)] for \_ in range(rows)]**

**index = 0**

**key\_order = sorted(range(len(key)), key=lambda k: key[k])**

**for col in key\_order:**

**for row in range(rows):**

**if index < len(ct):**

**matrix[row][col] = ct[index]**

**index += 1**

**pt = ""**

**for r in range(rows):**

**for c in range(cols):**

**pt += matrix[r][c]**

**return pt.rstrip('X')**

**def encrypt\_col\_transposition(pt, key):**

**rows = len(key)**

**cols = int(math.ceil(len(pt)/len(key)))**

**matrix = [['' for \_ in range(cols)] for \_ in range(rows)]**

**index = 0**

**for c in range(cols):**

**for r in range(rows):**

**if index < len(pt):**

**matrix[r][c] = pt[index]**

**index += 1**

**else:**

**matrix[r][c] = 'X'**

**key\_order = sorted(range(len(key)), key=lambda k: key[k])**

**ct = ""**

**for row in key\_order:**

**for col in range(cols):**

**ct += matrix[row][col]**

**return ct**

**def decrypt\_col\_transposition(ct, key):**

**rows = len(key)**

**cols = int(math.ceil(len(ct)/len(key)))**

**matrix = [['' for \_ in range(cols)] for \_ in range(rows)]**

**index = 0**

**key\_order = sorted(range(len(key)), key=lambda k: key[k])**

**for row in key\_order:**

**for col in range(cols):**

**if index < len(ct):**

**matrix[row][col] = ct[index]**

**index += 1**

**pt = ""**

**for c in range(cols):**

**for r in range(rows):**

**pt += matrix[r][c]**

**return pt.rstrip('X')**

**def encrypt\_double\_transposition(pt, row\_key, col\_key):**

**first\_stage = encrypt\_row\_transposition(pt, row\_key)**

**return encrypt\_col\_transposition(first\_stage, col\_key)**

**def decrypt\_double\_transposition(ct, row\_key, col\_key):**

**first\_stage = decrypt\_col\_transposition(ct, col\_key)**

**return decrypt\_row\_transposition(first\_stage, row\_key)**

**def main():**

**while True:**

**print("Menu:")**

**print("1. Encrypt Row Transposition Cipher")**

**print("2. Decrypt Row Transposition Cipher")**

**print("3. Encrypt Column Transposition Cipher")**

**print("4. Decrypt Column Transposition Cipher")**

**print("5. Encrypt Double Transposition Cipher (Row + Column)")**

**print("6. Decrypt Double Transposition Cipher (Column + Row)")**

**print("7. Exit")**

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

**if choice in [1, 3, 5]:**

**pt = input("Enter the plaintext: ").replace(" ", "")**

**if choice in [2, 4, 6]:**

**ct = input("Enter the ciphertext: ")**

**if choice == 1:**

**key = input("Enter the key: ")**

**print("Cipher Text: ", encrypt\_row\_transposition(pt, key))**

**elif choice == 2:**

**key = input("Enter the key: ")**

**print("Plain Text: ", decrypt\_row\_transposition(ct, key))**

**elif choice == 3:**

**key = input("Enter the key: ")**

**print("Cipher Text: ", encrypt\_col\_transposition(pt, key))**

**elif choice == 4:**

**key = input("Enter the key: ")**

**print("Plain Text: ", decrypt\_col\_transposition(ct, key))**

**elif choice == 5:**

**row\_key = input("Enter the row key: ")**

**col\_key = input("Enter the column key: ")**

**print("Cipher Text: ", encrypt\_double\_transposition(pt, row\_key, col\_key))**

**elif choice == 6:**

**row\_key = input("Enter the row key: ")**

**col\_key = input("Enter the column key: ")**

**print("Plain Text: ", decrypt\_double\_transposition(ct, row\_key, col\_key))**

**elif choice == 7:**

**print("Exiting the program!")**

**break**

**else:**

**print("Invalid Choice!")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**