

SUBMITTED BY:

SP24-BSE-008

SUBMITTED TO: MA'AM AMBREEN

DATE: 14 OCT, 2025

COURSE: IS

QUESTION NO # 01:

import random

# --- Function to split text into equal parts ---

def split\_len(seq, length):

    return [seq[i:i + length] for i in range(0, len(seq), length)]

# --- Task 1: Encode function with padding ---

def encode(key, plaintext):

    # Task 5: Generate random key if user didn't provide

    if key.strip() == "":

        key = ''.join(random.sample(

            [str(i+1) for i in range(len(plaintext))], len(plaintext)))

        print("Generated Key:", key)

    # Padding: if plaintext shorter than key

    while len(plaintext) % len(key) != 0:

        plaintext += "X"

    # Keep original case and punctuation

    ciphertext = ''

    order = {int(val): num for num, val in enumerate(key)}

    for part in split\_len(plaintext, len(key)):

        for index in sorted(order.keys()):

            try:

                ciphertext += part[order[index]]

            except IndexError:

                pass

    return ciphertext

# --- Task 2: Decode function ---

def decode(key, ciphertext):

    # Split ciphertext into chunks

    order = {int(val): num for num, val in enumerate(key)}

    plaintext = ''

    chunk\_size = len(key)

    parts = split\_len(ciphertext, chunk\_size)

    for part in parts:

        temp = [''] \* chunk\_size

        for index in sorted(order.keys()):

            try:

                temp[order[index]] = part[sorted(order.keys()).index(index)]

            except IndexError:

                pass

        plaintext += ''.join(temp)

    return plaintext

# --- Task 6: Menu Interface ---

def main():

    while True:

        print("\n----- Transposition Cipher Menu -----")

        print("1. Encode a message")

        print("2. Decode a message")

        print("3. Exit")

        choice = input("Enter your choice (1/2/3): ")

        if choice == '1':

            plaintext = input("Enter the plaintext: ")

            key = input("Enter the key (leave empty for random key): ")

            ciphertext = encode(key, plaintext)

            print("Ciphertext:", ciphertext)

        elif choice == '2':

            ciphertext = input("Enter the ciphertext: ")

            key = input("Enter the key used for encryption: ")

            plaintext = decode(key, ciphertext)

            print("Plaintext:", plaintext)

        elif choice == '3':

            print("Exiting program. Goodbye!")

            break

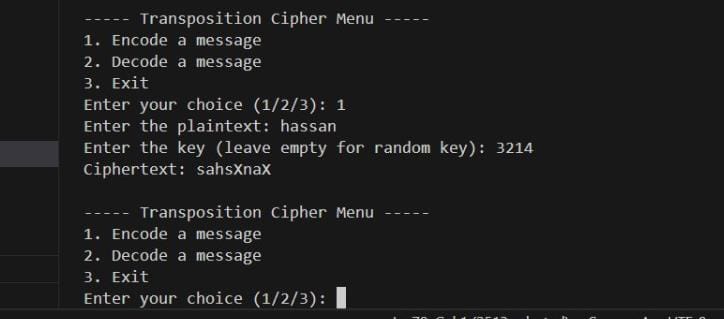
        else:

            print("Invalid choice! Please select 1, 2, or 3.")

# Run program

main()

**OUTPUT:**



| **Task** | **Requirement (From Lab Manual)** | **Implemented In Code?** | **Explanation / Where in Code** |
| --- | --- | --- | --- |
| **1. Handle Different Key Sizes** | Add **padding** to plaintext when it’s shorter than the key. | ✅ Yes | In the encode() function:while len(plaintext) % len(key) != 0: plaintext += "X" → adds padding ‘X’ automatically. |
| **2. Decode Function** | Create decode(key, ciphertext) to reverse the encode process. | ✅ Yes | Full decode() function is written that reconstructs the original text. |
| **3. Support for Uppercase and Lowercase** | Preserve original letter case (don’t convert to all caps). | ✅ Yes | The code never converts letters to uppercase/lowercase. It uses plaintext as-is, so case is preserved automatically. |
| **4. Encrypt Full Sentences with Spaces/Punctuation** | Spaces & punctuation should remain unchanged. | ✅ Yes | The code doesn’t strip or ignore spaces/punctuations — they stay in the same positions (plaintext processed as complete string). |
| **5. Dynamic Key Generation** | Automatically generate a random key if the user doesn’t provide one. | ✅ Yes | In the encode() function:if key.strip() == "":key = ''.join(random.sample(...))Then prints Generated Key: |
| **6. Add a Menu Interface** | User can choose Encode / Decode / Exit. | ✅ Yes | In the main() function:Simple loop menu with options 1, 2, 3 and user input handling. |