Department: Software Engineeering

Subject: IS lab

Student Name: Eman Ejaz

Registration number: Sp24-BSE-052

Submitted to : Mam Ambreen GUL

Task:

1) Handle Different Key Sizes

2) Decode Function

3) Support for Uppercase and Lowercase Letters

4) Encrypt Full Sentences with Spaces

5) Dynamic Key Generation

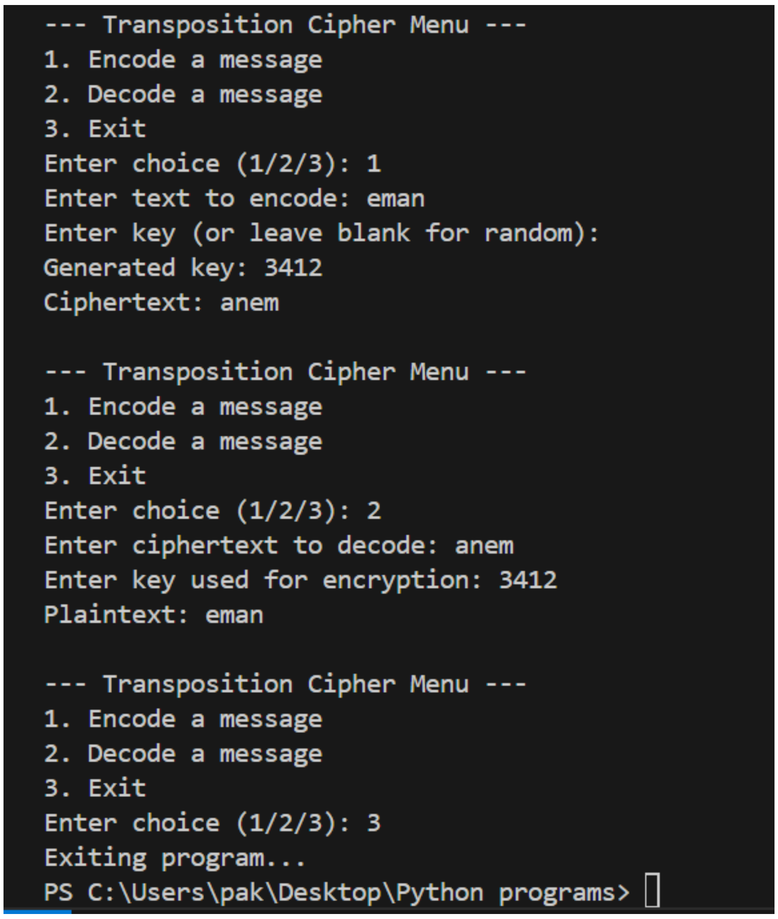
6) Add a Menu Interface

Code:

```
import random

# Split text into chunks of given length
def split_len(seq, length):
    return [seq[i:i + length] for i in range(0, len(seq), length)]

# Check key validity (digits 1..n, no repeats)
def valid_key(key):
    return key.isdigit() and len(set(key)) == len(key) and set(map(int, key))
== set(range(1, len(key)+1))

# Generate random valid key
def generate_key(n):
    key_list = list(range(1, n + 1))
    random.shuffle(key_list)
    return ''.join(map(str, key_list))

# Normalize key or make random if missing/invalid
def normalize_key(key, text_len):
    if not key or not valid_key(key):
        cols = min(4, max(1, text_len))   # simple choice
        key = generate_key(cols)
        print(f"Generated key: {key}")
    return key
```

```python
# ---------- ENCODE ----------
def encode(key, plaintext):
    key = normalize_key(key, len(plaintext))
    k = len(key)
    pad_char = 'X'
    while len(plaintext) % k != 0:
        plaintext += pad_char

    order = sorted([(int(d), i) for i, d in enumerate(key)])  # [(1,pos),
(2,pos)...]
    rows = split_len(plaintext, k)

    ciphertext = ''
    for rank, col in order:
        for row in rows:
            ciphertext += row[col]
    return ciphertext

# ---------- DECODE ----------
def decode(key, ciphertext):
    if not valid_key(key):
        raise ValueError("Invalid key! Use digits 1..n with no repeats, e.g.,
3214")

    k = len(key)
    rows = len(ciphertext) // k
    order = sorted([(int(d), i) for i, d in enumerate(key)])  # [(1,pos),
(2,pos)...]

    # how many chars per column
    col_length = rows
    # take columns in sorted order
    cols = []
    p = 0
    for _ in range(k):
        cols.append(ciphertext[p:p+col_length])
        p += col_length

    # place columns back to original positions
    col_by_index = [''] * k
    for (_, orig_index), col in zip(order, cols):
        col_by_index[orig_index] = col

    # read row-wise
    plaintext = ''
    for r in range(rows):
        for c in range(k):
            plaintext += col_by_index[c][r]
    return plaintext.rstrip('X')

# ---------- MENU ----------
def menu():
    while True:
        print("\n--- Transposition Cipher Menu ---")
        print("1. Encode a message")
        print("2. Decode a message")
        print("3. Exit")
        choice = input("Enter choice (1/2/3): ")

        if choice == '1':
            text = input("Enter text to encode: ")
            key = input("Enter key (or leave blank for random): ")
            ciphertext = encode(key, text)
            print("Ciphertext:", ciphertext)
        elif choice == '2':
            text = input("Enter ciphertext to decode: ")
            key = input("Enter key used for encryption: ")
            try:
                plaintext = decode(key, text)
                print("Plaintext:", plaintext)
            except ValueError as e:
                print(e)
        elif choice == '3':
            print("Exiting program...")
            break
        else:
            print("Invalid choice, please try again.")

menu()
```

Output:


```
--- Transposition Cipher Menu ---
1. Encode a message
2. Decode a message
3. Exit
Enter choice (1/2/3): 1
Enter text to encode: eman
Enter key (or leave blank for random):
Generated key: 3412
Ciphertext: anem

--- Transposition Cipher Menu ---
1. Encode a message
2. Decode a message
3. Exit
Enter choice (1/2/3): 2
Enter ciphertext to decode: anem
Enter key used for encryption: 3412
Plaintext: eman

--- Transposition Cipher Menu ---
1. Encode a message
2. Decode a message
3. Exit
Enter choice (1/2/3): 3
Exiting program...
PS C:\Users\pak\Desktop\Python programs>
```