EXPERIMENT 3

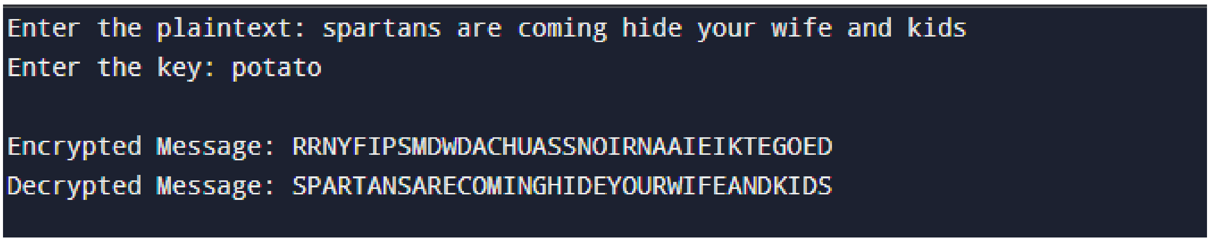
Single Columnar Cipher:

Code:

```
import math


def encrypt(message, key):
    # Remove spaces and convert message to uppercase
    message = message.replace(" ", "").upper()

    # Determine number of columns and rows
    cols = len(key)
    rows = math.ceil(len(message) / cols)

    # Add padding if necessary
    message += 'X' * (rows * cols - len(message))

    # Create a matrix and populate it row-wise
    matrix = [list(message[i * cols: (i + 1) * cols]) for i in range(rows)]

    # Get the order of columns based on the sorted key
    sorted_indices = sorted(range(len(key)), key=lambda k: key[k])

    # Read the matrix column-wise according to sorted key order
    ciphertext = ".join(".join(matrix[row][col] for row in range(rows)) for col in
sorted_indices)

    return ciphertext


def decrypt(ciphertext, key):
    # Determine number of columns and rows
    cols = len(key)
    rows = len(ciphertext) // cols

    # Get the order of columns based on the sorted key
    sorted_indices = sorted(range(len(key)), key=lambda k: key[k])

    # Create an empty matrix
    matrix = [["] * cols for _ in range(rows)]

    # Fill the matrix column-wise using the sorted column order
    idx = 0
    for col in sorted_indices:
        for row in range(rows):
            matrix[row][col] = ciphertext[idx]
            idx += 1

    # Read the matrix row-wise to reconstruct the plaintext
    plaintext = ".join(".join(row) for row in matrix).rstrip('X')  # Remove padding

    return plaintext

# Take user input
message = input("Enter the plaintext: ")
key = input("Enter the key: ")

# Encrypt the message
cipher = encrypt(message, key)
print("\nEncrypted Message:", cipher)

# Decrypt the message
decrypted = decrypt(cipher, key)
```

print("Decrypted Message:", decrypted)

Output:



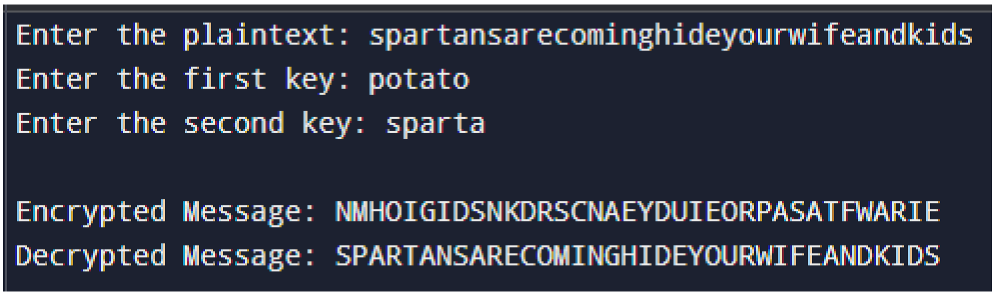
```
Enter the plaintext: spartans are coming hide your wife and kids
Enter the key: potato

Encrypted Message: RRNYFIPSMDWDACHUASSNOIRNAAIEIKTEGOED
Decrypted Message: SPARTANSARECOMINGHIDEYOURWIFEANDKIDS
```

Double Transposition:

Code:

```
import math


def columnar_encrypt(message, key):
    """Encrypts a message using a single columnar transposition cipher."""
    message = message.replace(" ", "").upper()  # Remove spaces & convert to uppercase
    cols = len(key)
    rows = math.ceil(len(message) / cols)

    # Add padding if necessary
    message += 'X' * (rows * cols - len(message))

    # Create a matrix row-wise
    matrix = [list(message[i * cols:(i + 1) * cols]) for i in range(rows)]

    # Get column order based on sorted key
    sorted_indices = sorted(range(len(key)), key=lambda k: key[k])

    # Read the matrix column-wise based on key order
    ciphertext = ''.join(''.join(matrix[row][col] for row in range(rows)) for col in
sorted_indices)
    return ciphertext


def columnar_decrypt(ciphertext, key):
    """Decrypts a message using a single columnar transposition cipher."""
    cols = len(key)
    rows = len(ciphertext) // cols

    # Get column order based on sorted key
    sorted_indices = sorted(range(len(key)), key=lambda k: key[k])

    # Create an empty matrix
    matrix = [["] * cols for _ in range(rows)]

    # Fill the matrix column-wise using sorted key order
    idx = 0
    for col in sorted_indices:
        for row in range(rows):
            matrix[row][col] = ciphertext[idx]
            idx += 1

    # Read the matrix row-wise to reconstruct the plaintext
    plaintext = ".join(".join(row) for row in matrix).rstrip('X')  # Remove padding
    return plaintext

def double_transposition_encrypt(message, key1, key2):
    """Encrypts a message using double columnar transposition."""
    first_pass = columnar_encrypt(message, key1)
    second_pass = columnar_encrypt(first_pass, key2)
    return second_pass

def double_transposition_decrypt(ciphertext, key1, key2):
    """Decrypts a message using double columnar transposition."""
    first_pass = columnar_decrypt(ciphertext, key2)
    second_pass = columnar_decrypt(first_pass, key1)
    return second_pass

# Take user input
message = input("Enter the plaintext: ")
key1 = input("Enter the first key: ")
key2 = input("Enter the second key: ")

# Encrypt the message
ciphertext = double_transposition_encrypt(message, key1, key2)
print("\nEncrypted Message:", ciphertext)

# Decrypt the message
decrypted_text = double_transposition_decrypt(ciphertext, key1, key2)
print("Decrypted Message:", decrypted_text)
```

Output:



```
Enter the plaintext: spartansarecominghideyourwifeandkids
Enter the first key: potato
Enter the second key: sparta

Encrypted Message: NMHOIGIDSNKDRSCNAEYDUIEORPASATFWARIE
Decrypted Message: SPARTANSARECOMINGHIDEYOURWIFEANDKIDS
```