**Department of Computer Engineering**

**Name of the Student: Parth Das Roll Number: C-111**

**SAP ID: 60004220185 Class: Comps Division: C-2 Batch: 2**

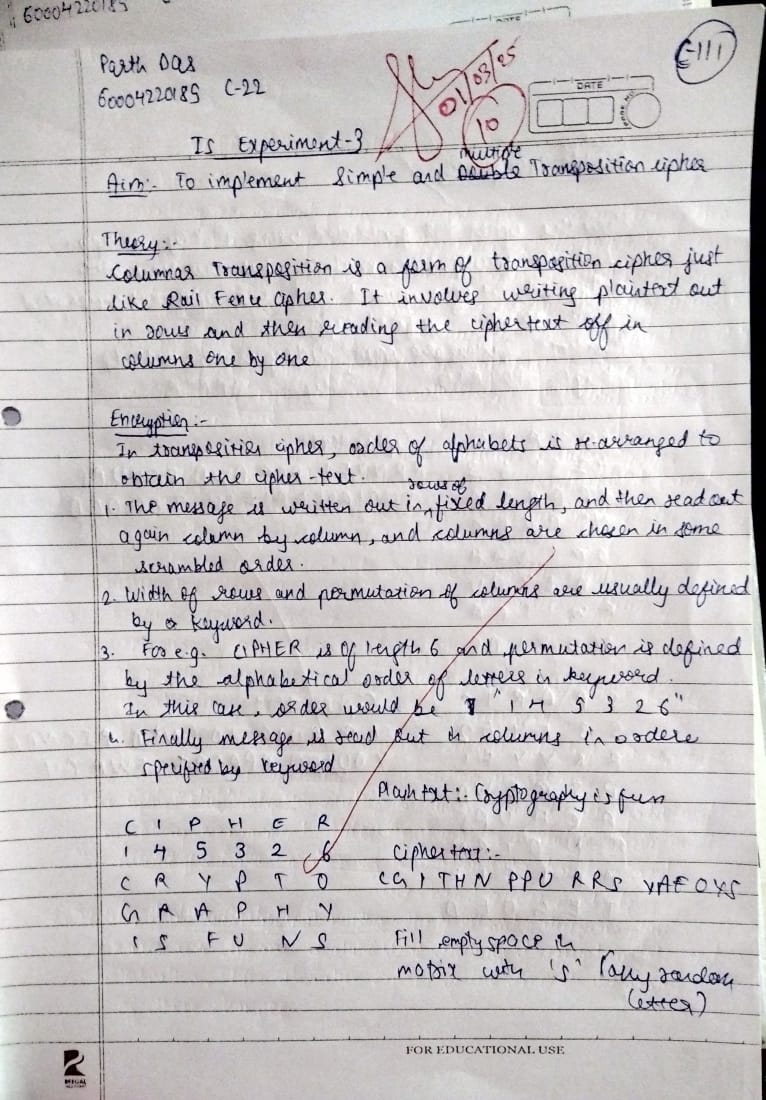
**Subject: Information Security**

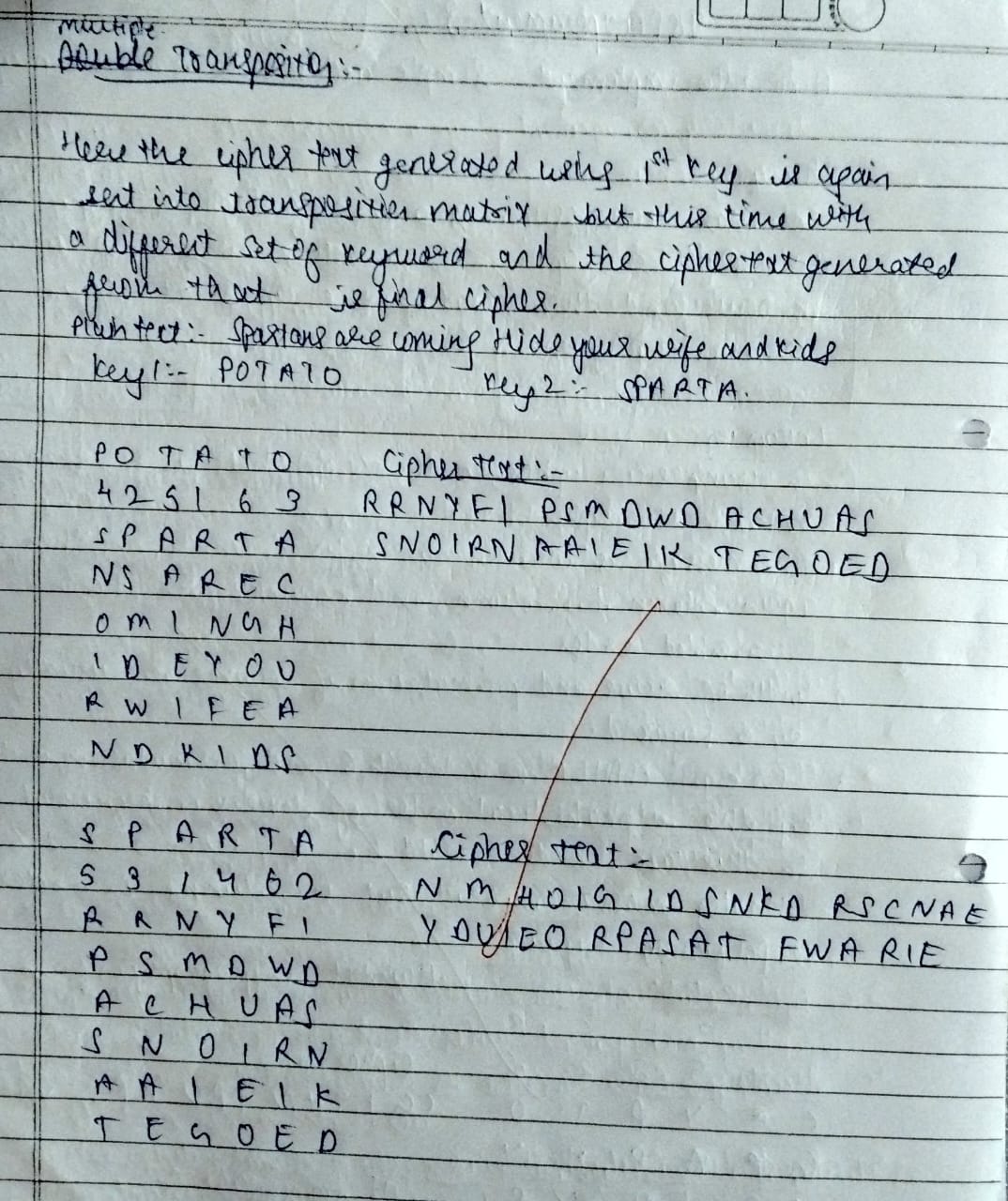
**DATE OF PERFORMANCE: 15/02/2025 DATE OF SUBMISION: 16/02/2025**

**EXPERIMENT NO: 3**

**AIM:** Design and Implement Simple Columnar Transposition Cipher with Single Round & Multiple Round

**Theory:**

****

****

**SOURCE CODE WITH OUTPUTS:**

import math

MAX\_KEY\_LEN = 20

MAX\_TEXT\_LEN = 200

MAX\_ROUNDS = 10

def generate\_column\_order(key, len\_key):

sorted\_key = sorted(key)

col\_order = [-1] \* len\_key

index = 0

for i in range(len\_key):

for j in range(len\_key):

if key[j] == sorted\_key[i] and col\_order[j] == -1:

col\_order[j] = index

index += 1

break

return col\_order

def encrypt(key, plain\_text):

len\_key = len(key)

len\_text = len(plain\_text)

num\_rows = math.ceil(len\_text / len\_key)

matrix = [['X'] \* len\_key for \_ in range(num\_rows)]

col\_order = [-1] \* len\_key

index = 0

for i in range(num\_rows):

for j in range(len\_key):

if index < len\_text:

matrix[i][j] = plain\_text[index]

index += 1

col\_order = generate\_column\_order(key, len\_key)

cipher\_text = []

index = 0

for i in range(len\_key):

col = col\_order.index(i)

for r in range(num\_rows):

cipher\_text.append(matrix[r][col])

return ''.join(cipher\_text)

def decrypt(key, cipher\_text):

len\_key = len(key)

len\_cipher = len(cipher\_text)

num\_rows = math.ceil(len\_cipher / len\_key)

matrix = [['X'] \* len\_key for \_ in range(num\_rows)]

col\_order = generate\_column\_order(key, len\_key)

index = 0

for i in range(len\_key):

col = col\_order.index(i)

for r in range(num\_rows):

matrix[r][col] = cipher\_text[index]

index += 1

decrypted\_text = []

for i in range(num\_rows):

for j in range(len\_key):

if matrix[i][j] != 'X':

decrypted\_text.append(matrix[i][j])

return ''.join(decrypted\_text)

def main():

num\_rounds = int(input("Enter number of encryption rounds: "))

keys = []

for i in range(num\_rounds):

key = input(f"Enter keyword {i + 1} (Unique letters only): ")

keys.append(key.strip())

plain\_text = input("Enter plain text (Letters only): ").strip()

cipher\_text = plain\_text

print("\nEncryption Process")

for i in range(num\_rounds):

cipher\_text = encrypt(keys[i], cipher\_text)

print(f"Round {i + 1} Cipher Text using Key {i + 1}: {cipher\_text}")

decrypted\_text = cipher\_text

print("\nDecryption Process")

for i in range(num\_rounds - 1, -1, -1):

decrypted\_text = decrypt(keys[i], decrypted\_text)

print(f"Round {num\_rounds - i} Decrypted Text using Key {i + 1}: {decrypted\_text}")

print("\nFinal Results")

print(f"Original Plain Text: {plain\_text}")

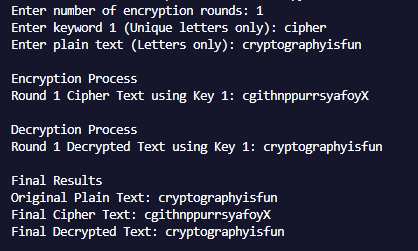
print(f"Final Cipher Text: {cipher\_text}")

print(f"Final Decrypted Text: {decrypted\_text}")

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

main()

**Output:-**



**CONCLUSION:**

The single-round columnar transposition cipher provides a simple yet effective method of encryption by rearranging plaintext columns based on a predefined key. While it offers basic security, it is vulnerable to frequency analysis and anagramming techniques.

In contrast, the multiple-round columnar transposition cipher enhances security by applying transposition iteratively, making pattern detection and cryptanalysis significantly harder. The increased complexity strengthens resistance against brute-force attacks but also demands careful key management to ensure decryption accuracy.