**Department of Computer Engineering**

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

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

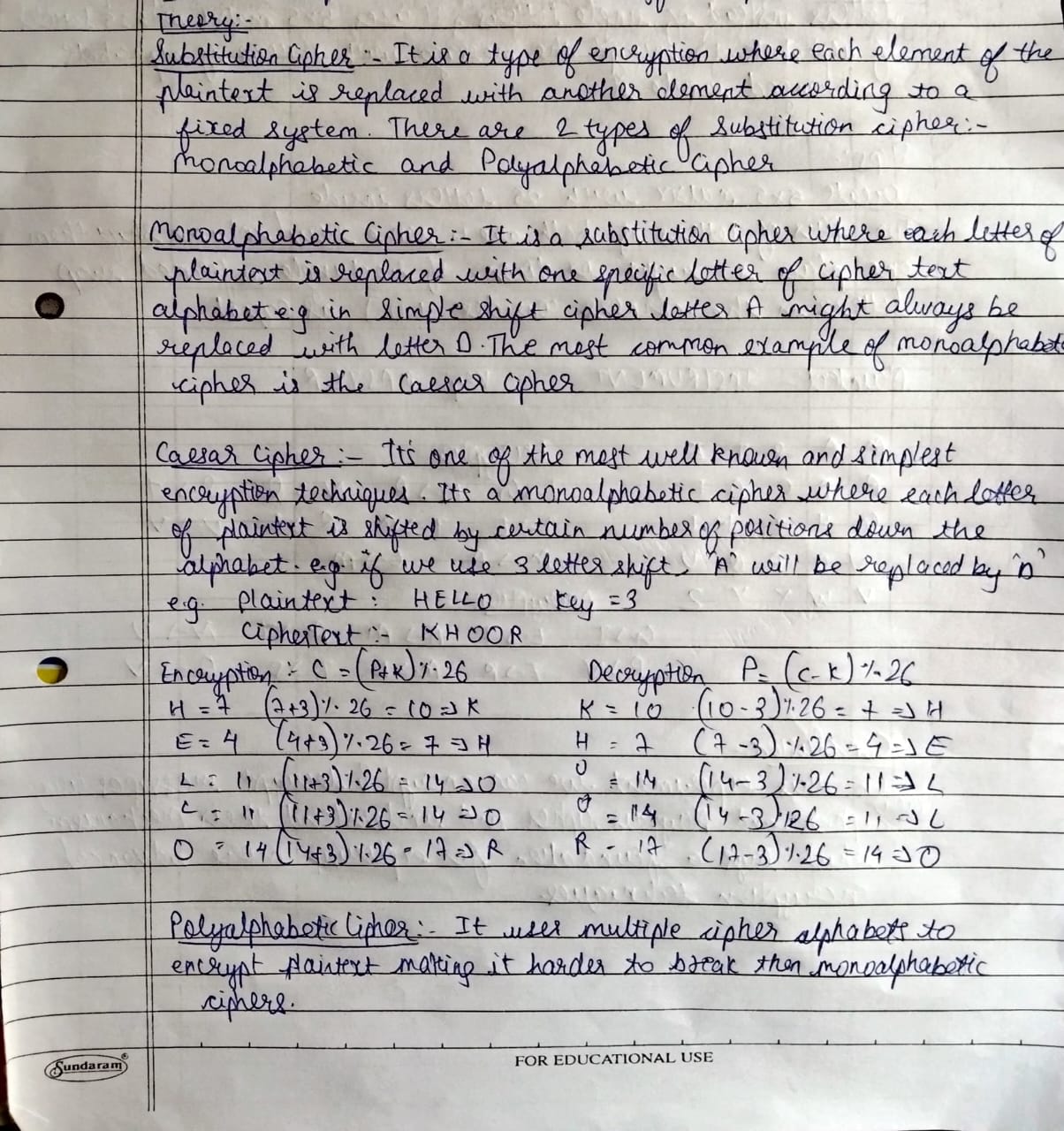
**Subject: Information Security**

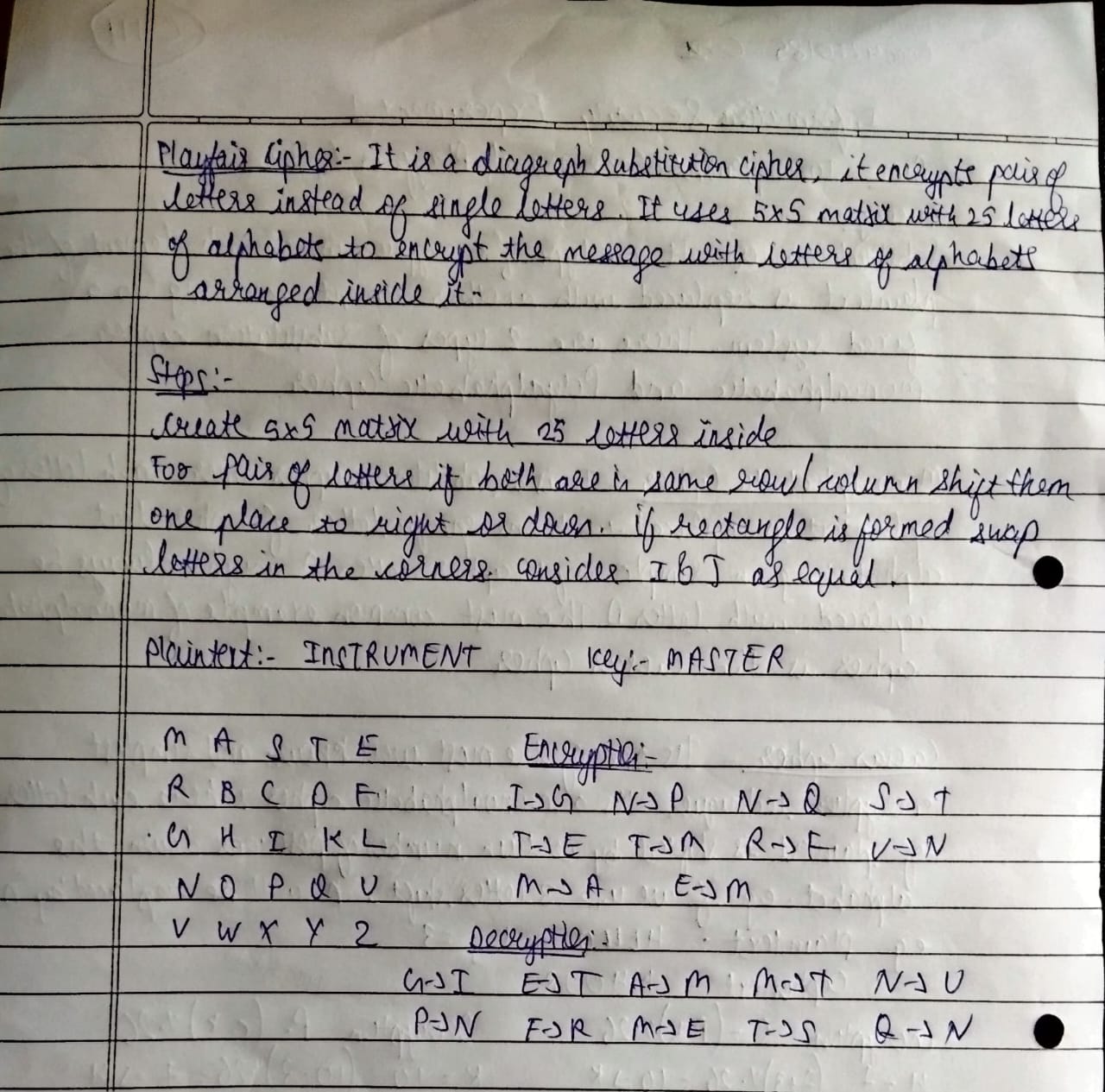
**DATE OF PERFORMANCE: 01/02/2025 DATE OF SUBMISION: 08/02/2025**

**EXPERIMENT NO: 1**

**AIM: To Implement Caesar and PlayFair Cipher**

**Theory:**

****

****

**SOURCE CODE WITH OUTPUTS:**

1. **Caesar Cipher**

**Code:-**

message = input('Enter your plain text:')

shift = 3

def caesar(message, shift):

alphabet = 'abcdefghijklmnopqrstuvwxyz'

encrypted\_text = ''

for char in message.lower():

if char == ' ':

encrypted\_text += 'p'

else:

index = alphabet.find(char)

new\_index = (index + shift) % len(alphabet)

encrypted\_text += alphabet[new\_index]

print('plain text:', message)

print('encrypted text:', encrypted\_text)

caesar(message, shift)

**Output:-**



1. **PlayFair Cipher**

**Code:-**

def create\_playfair\_matrix(key):

key = key.replace('j', 'i').lower()

matrix = []

seen = set()

for char in key:

if char not in seen and char.isalpha():

matrix.append(char)

seen.add(char)

alphabet = 'abcdefghiklmnopqrstuvwxyz' # j is merged with i

for char in alphabet:

if char not in seen:

matrix.append(char)

seen.add(char)

return [matrix[i:i + 5] for i in range(0, len(matrix), 5)]

def get\_position(matrix, char):

for i, row in enumerate(matrix):

for j, c in enumerate(row):

if c == char:

return i, j

# remove spaces, combine double letters

def preprocess\_message(message):

message = message.replace(' ', '').lower()

processed\_message = []

i = 0

while i < len(message):

if i + 1 < len(message) and message[i] == message[i + 1]:

processed\_message.append(message[i] + 'x')

i += 1

else:

processed\_message.append(message[i])

i += 1

# If the length of the message is odd, append an 'x' at the end

if len(processed\_message) % 2 != 0:

processed\_message.append('x')

return processed\_message

def encrypt(message, matrix):

message = preprocess\_message(message)

encrypted\_message = []

for i in range(0, len(message), 2):

row1, col1 = get\_position(matrix, message[i])

row2, col2 = get\_position(matrix, message[i + 1])

# If both letters are in the same row, take the letters to the immediate right

if row1 == row2:

encrypted\_message.append(matrix[row1][(col1 + 1) % 5])

encrypted\_message.append(matrix[row2][(col2 + 1) % 5])

# If both letters are in the same column, take the letters immediately below

elif col1 == col2:

encrypted\_message.append(matrix[(row1 + 1) % 5][col1])

encrypted\_message.append(matrix[(row2 + 1) % 5][col2])

# If neither of the above, form a rectangle and swap columns

else:

encrypted\_message.append(matrix[row1][col2])

encrypted\_message.append(matrix[row2][col1])

return ''.join(encrypted\_message)

def decrypt(message, matrix):

message = preprocess\_message(message)

decrypted\_message = []

for i in range(0, len(message), 2):

row1, col1 = get\_position(matrix, message[i])

row2, col2 = get\_position(matrix, message[i + 1])

# If both letters are in the same row, take the letters to the immediate left

if row1 == row2:

decrypted\_message.append(matrix[row1][(col1 - 1) % 5])

decrypted\_message.append(matrix[row2][(col2 - 1) % 5])

# If both letters are in the same column, take the letters immediately above

elif col1 == col2:

decrypted\_message.append(matrix[(row1 - 1) % 5][col1])

decrypted\_message.append(matrix[(row2 - 1) % 5][col2])

# If neither of the above, form a rectangle and swap columns

else:

decrypted\_message.append(matrix[row1][col2])

decrypted\_message.append(matrix[row2][col1])

return ''.join(decrypted\_message).replace('x', '')

# Main code to demonstrate encryption and decryption

def main():

key = input("Enter the key for Playfair cipher: ")

message = input("Enter the message to encrypt: ")

matrix = create\_playfair\_matrix(key)

print("Playfair Cipher Matrix:")

for row in matrix:

print(row)

encrypted\_message = encrypt(message, matrix)

print("Encrypted Message:", encrypted\_message)

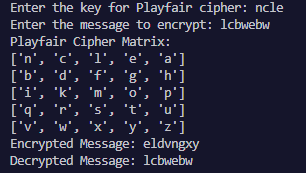
decrypted\_message = decrypt(encrypted\_message, matrix)

print("Decrypted Message:", decrypted\_message)

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

main()

**Output:-**



**CONCLUSION:**

In this Experiment, We learnt to encrypt and decrypt messages using Caesar Cipher and PlayFair Cipher and understand their mechanism which lays a foundation for exploring more advanced encryption techniques.Through this, we also analyzed their strengths and vulnerabilities, recognizing the limitations of classical encryption in modern security.