**CNS Exp 2**

**Shashwat Tripathi  
D15A 64**

**Batch C**

**Aim:** Design and Implement a product cipher using Substitution ciphers and Transposition Cipher.

**Theory:**

A product cipher is a type of encryption scheme that combines multiple encryption techniques, such as substitution and transposition ciphers, in order to enhance security. By applying multiple layers of encryption, product ciphers aim to provide a higher level of complexity and make it more difficult for attackers to break the cipher.

Substitution Cipher:

The substitution cipher is a method where each letter of the plaintext is replaced with another letter or symbol according to a fixed rule or key. For example, a simple substitution cipher may replace each letter with the letter that appears three positions after it in the alphabet.

a. Substitution Encryption: Apply a substitution cipher to the plaintext to produce a modified plaintext (substituted characters).

b. Transposition Encryption: Rearrange the substituted characters from the previous step using a transposition cipher to create the ciphertext.

Transposition Cipher:

The transposition cipher is a method where the letters of the plaintext are rearranged or permuted based on a fixed rule or key. For example, a simple transposition cipher may rearrange the letters in the plaintext in a specific pattern, such as reading them in columns instead of rows.

a. Transposition Decryption: Reverse the transposition cipher to rearrange the ciphertext into the substituted characters.

b. Substitution Decryption: Apply the inverse of the substitution cipher to obtain the original plaintext.

**Code:**

import java.util.\*;

class ProductCipher {

public static void main(String args[]) {

System.out.println("Enter the input to be encrypted:");

String substitutionInput = new Scanner(System.in).nextLine();

System.out.println("Enter a number:");

int n = new Scanner(System.in).nextInt();

// Substitution encryption

StringBuffer substitutionOutput = new StringBuffer();

for (int i = 0; i < substitutionInput.length(); i++) {

char c = substitutionInput.charAt(i);

substitutionOutput.append((char) (c + 3));

}

System.out.println("\nSubstituted text:");

System.out.println(substitutionOutput);

// Transposition encryption

String transpositionInput = substitutionOutput.toString();

int modulus;

if ((modulus = transpositionInput.length() % n) != 0) {

modulus = n - modulus;

// ‘modulus’ is now the number of blanks/padding (X) to be appended

for (int a = modulus; a != 0; a--) {

transpositionInput += "/";

}

}

StringBuffer transpositionOutput = new StringBuffer();

System.out.println("\nTransposition Matrix:");

for (int i = 0; i < n; i++) {

for (int j = 0; j < transpositionInput.length() / n; j++) {

char c = transpositionInput.charAt(i + (j \* n));

System.out.print(c);

transpositionOutput.append(c);

}

System.out.println();

}

System.out.println("\nFinal encrypted text:");

System.out.println(transpositionOutput);// Transposition decryption

n = transpositionOutput.length() / n;

StringBuffer transpositionPlaintext = new StringBuffer();

for (int i = 0; i < n; i++) {

for (int j = 0; j < transpositionOutput.length() / n; j++) {

char c = transpositionOutput.charAt(i + (j \* n));

transpositionPlaintext.append(c);

}

}

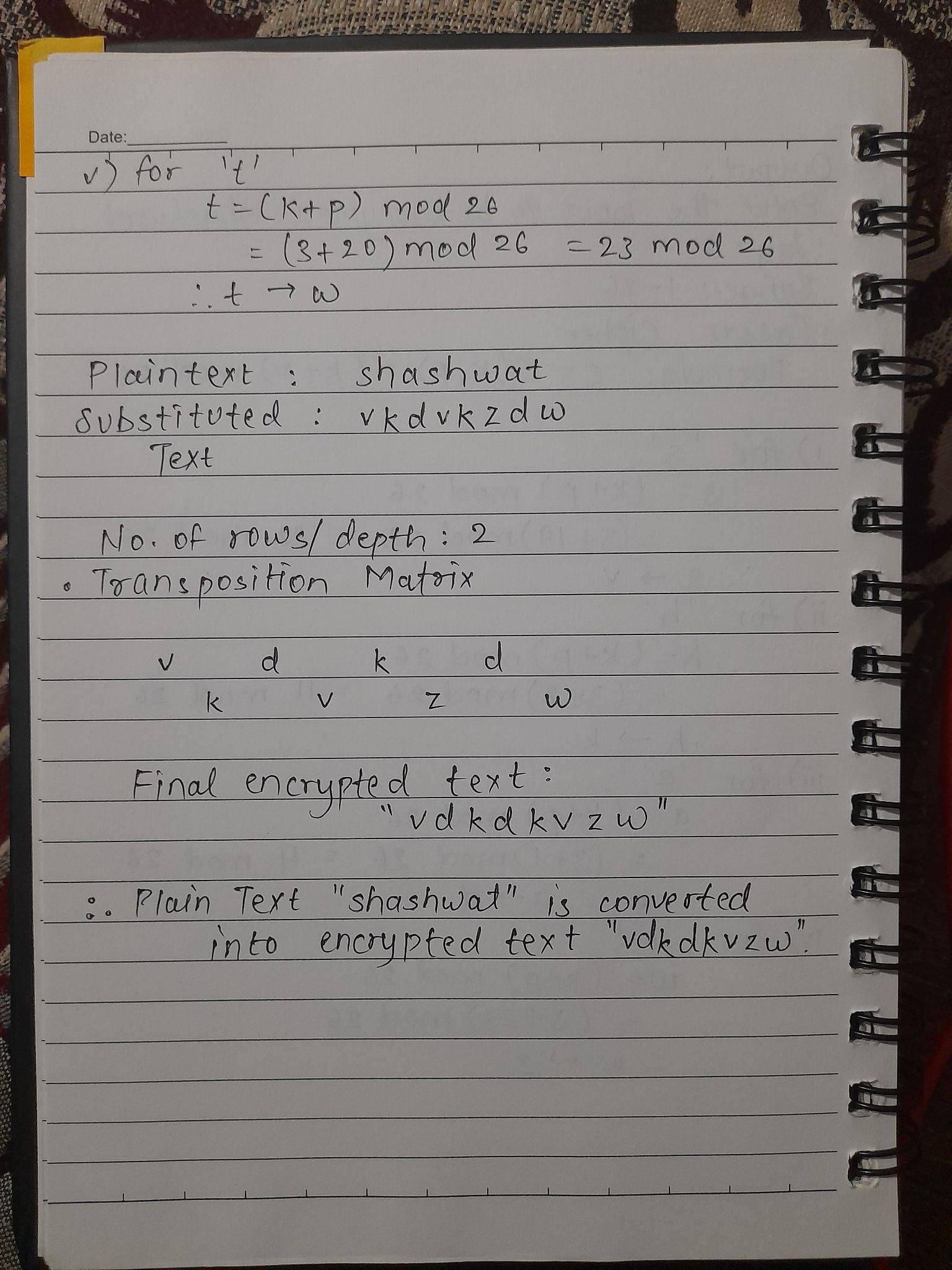
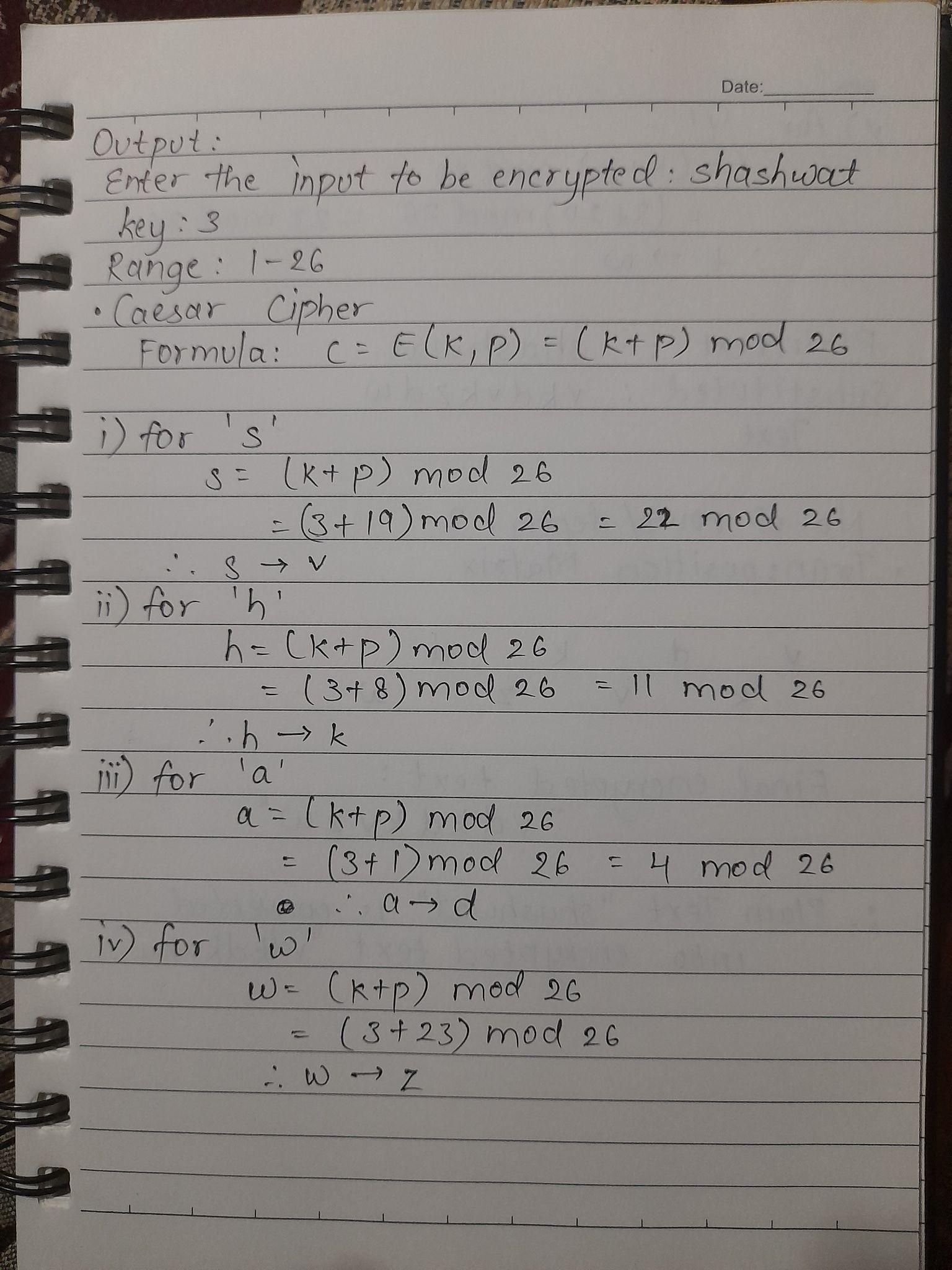
}

}

**Output:**

****

**Solving:**

****

**Conclusion:**

Thus, Design and Implement a product cipher using Substitution ciphers and Transposition Cipher.