Name: Tushar Panchal

En.No: 21162101014

Sub: INS (INFORMATION SECURITY)

Branch: CBA

Batch:61

---------------------------PRACTICAL 03---------------------------

❖ Question :

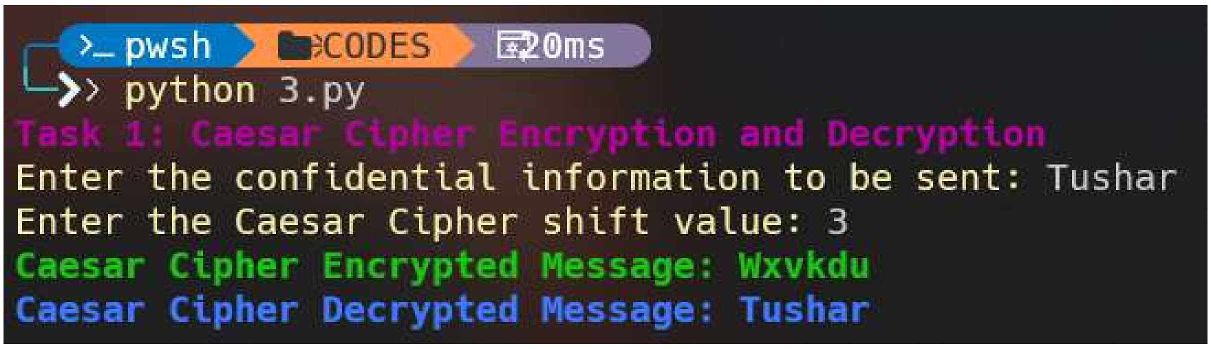
Alice wants to send some confidential information to Bob over a secure network, you have to create perform following task :
1)Provide Security using Caesar Cipher Algorithm
2)Find the all possible Cipher Text & Plaintext pairs
3)Provide Security Mono-alphabetic Cipher Algorithm

✓ Source Code :

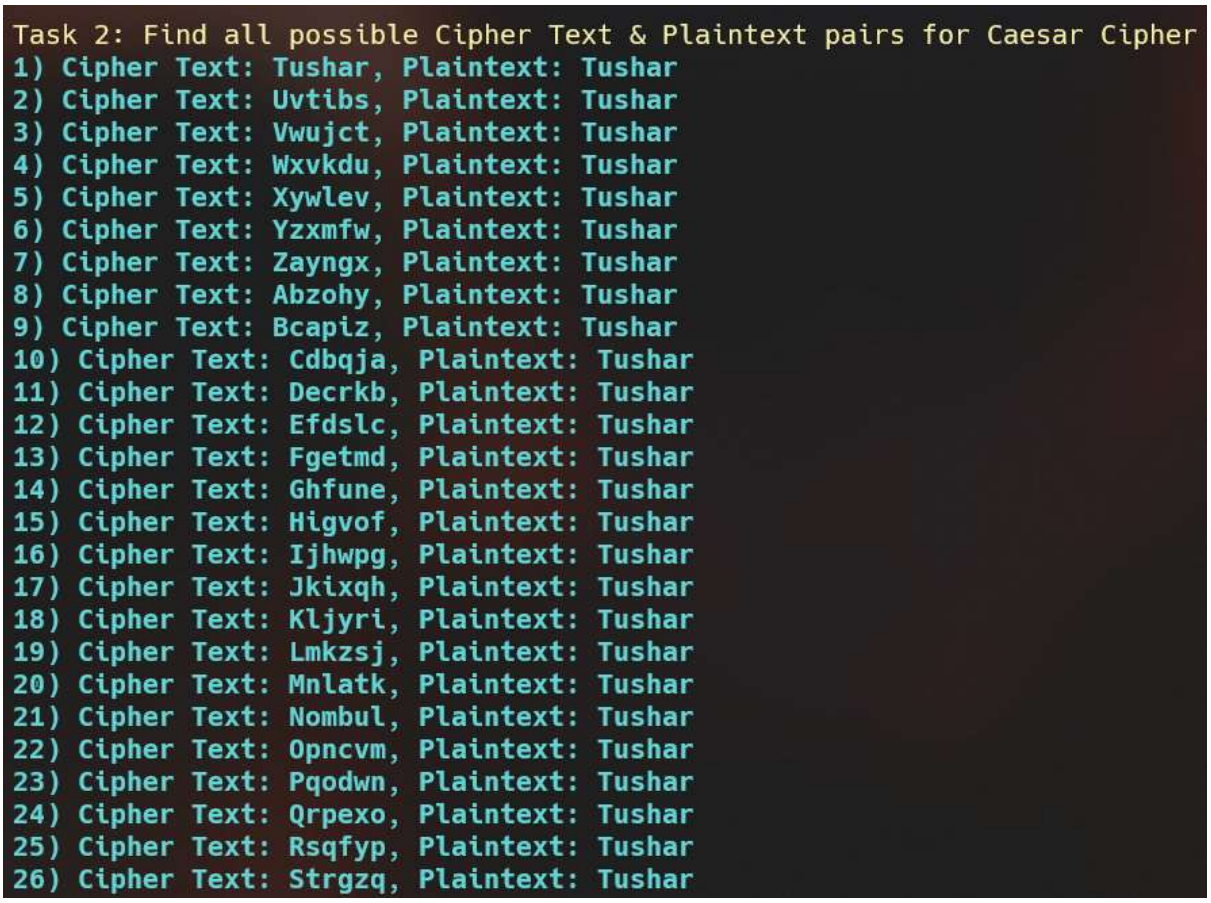
```python
def caesar_cipher_encrypt(plaintext, shift):
    ciphertext = ""
    for char in plaintext:
        if char.isalpha():
            if char.isupper():
                ciphertext += chr((ord(char) + shift - 65) % 26 + 65)
            else:
                ciphertext += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            ciphertext += char
    return ciphertext


def caesar_cipher_decrypt(ciphertext, shift):
    return caesar_cipher_encrypt(ciphertext, -shift)


def generate_caesar_cipher_pairs(plaintext):
    pairs = []
    for shift in range(26):
        ciphertext = caesar_cipher_encrypt(plaintext, shift)
        pairs.append((ciphertext, caesar_cipher_decrypt(ciphertext,
shift)))
    return pairs


def monoalphabetic_cipher_encrypt(plaintext, key):
    ciphertext = ""
    for char in plaintext:
        if char.isalpha():
            if char.isupper():
                ciphertext += key[ord(char) - 65].upper()
            else:
                ciphertext += key[ord(char) - 97]
        else:
            ciphertext += char
    return ciphertext


def monoalphabetic_cipher_decrypt(ciphertext, key):
    reversed_key = {char: orig_char for orig_char, char in zip(
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ", key.upper())}
    reversed_key.update({char.lower(): orig_char.lower() for orig_char,
char in zip(
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ", key.upper())})

    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
            plaintext += reversed_key.get(char, char)
        else:
            plaintext += char
    return plaintext


def validate_monoalphabetic_key(key):
    if len(set(key)) != 26 or not key.isalpha():
        raise ValueError(
            "Mono-alphabetic Cipher key must be a permutation of the
alphabet with 26 distinct characters.")

# Task 1: Caesar Cipher Encryption and Decryption
print("\033[95;1mTask 1: Caesar Cipher Encryption and Decryption\033[0m")
plaintext = input(
    "\033[93mEnter the confidential information to be sent: \033[0m")
shift = int(input("\033[93mEnter the Caesar Cipher shift value: \033[0m"))

caesar_ciphertext = caesar_cipher_encrypt(plaintext, shift)
```

```
print("\033[92;1mCaesar Cipher Encrypted Message: {}\033[0m".format(
    caesar_ciphertext))

caesar_decrypted_message = caesar_cipher_decrypt(caesar_ciphertext, shift)
print("\033[94;1mCaesar Cipher Decrypted Message: {}\033[0m\n".format(
    caesar_decrypted_message))

# Task 2: Find all possible Cipher Text & Plaintext pairs for Caesar
Cipher
print("\033[93mTask 2: Find all possible Cipher Text & Plaintext pairs for
Caesar Cipher\033[0m")
caesar_pairs = generate_caesar_cipher_pairs(plaintext)
for idx, pair in enumerate(caesar_pairs, 1):
    print("\033[96;1m{}) Cipher Text: {}, Plaintext: {}\033[0m".format(
        idx, pair[0], pair[1]))

# Task 3: Mono-alphabetic Cipher Encryption and Decryption
print("\033[95;1mTask 3: Mono-alphabetic Cipher Encryption and
Decryption\033[0m")
monoalphabetic_key = input(
    "\033[93mEnter the Mono-alphabetic Cipher key: \033[0m")

try:
    validate_monoalphabetic_key(monoalphabetic_key)
    monoalphabetic_ciphertext = monoalphabetic_cipher_encrypt(
        plaintext, monoalphabetic_key)
    print("\033[92;1mMono-alphabetic Cipher Encrypted Message:
{}\033[0m".format(monoalphabetic_ciphertext))

    monoalphabetic_decrypted_message = monoalphabetic_cipher_decrypt(
        monoalphabetic_ciphertext, monoalphabetic_key)
    print("\033[94;1mMono-alphabetic Cipher Decrypted Message:
{}\033[0m".format(
        monoalphabetic_decrypted_message))

except ValueError as e:
    print("\033[91;1mError: {}\033[0m".format(e))
```

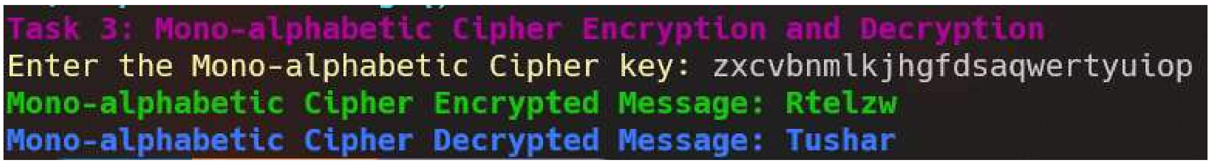
✓ **Output :**



```
>_ pwsh   CODES   20ms
>> python 3.py
Task 1: Caesar Cipher Encryption and Decryption
Enter the confidential information to be sent: Tushar
Enter the Caesar Cipher shift value: 3
Caesar Cipher Encrypted Message: Wxvkdu
Caesar Cipher Decrypted Message: Tushar
```



```
Task 2: Find all possible Cipher Text & Plaintext pairs for Caesar Cipher
1) Cipher Text: Tushar, Plaintext: Tushar
2) Cipher Text: Uvtibs, Plaintext: Tushar
3) Cipher Text: Vwujct, Plaintext: Tushar
4) Cipher Text: Wxvkdu, Plaintext: Tushar
5) Cipher Text: Xywlev, Plaintext: Tushar
6) Cipher Text: Yzxmfw, Plaintext: Tushar
7) Cipher Text: Zayngx, Plaintext: Tushar
8) Cipher Text: Abzohy, Plaintext: Tushar
9) Cipher Text: Bcapiz, Plaintext: Tushar
10) Cipher Text: Cdbqja, Plaintext: Tushar
11) Cipher Text: Decrkb, Plaintext: Tushar
12) Cipher Text: Efdslc, Plaintext: Tushar
13) Cipher Text: Fgetmd, Plaintext: Tushar
14) Cipher Text: Ghfune, Plaintext: Tushar
15) Cipher Text: Higvof, Plaintext: Tushar
16) Cipher Text: Ijhwpg, Plaintext: Tushar
17) Cipher Text: Jkixqh, Plaintext: Tushar
18) Cipher Text: Kljyri, Plaintext: Tushar
19) Cipher Text: Lmkzsj, Plaintext: Tushar
20) Cipher Text: Mnlatk, Plaintext: Tushar
21) Cipher Text: Nombul, Plaintext: Tushar
22) Cipher Text: Opncvm, Plaintext: Tushar
23) Cipher Text: Pqodwn, Plaintext: Tushar
24) Cipher Text: Qrpexo, Plaintext: Tushar
25) Cipher Text: Rsqfyp, Plaintext: Tushar
26) Cipher Text: Strgzq, Plaintext: Tushar
```



```
Task 3: Mono-alphabetic Cipher Encryption and Decryption
Enter the Mono-alphabetic Cipher key: zxcvbnmlkjhgfdsaqwertyuiop
Mono-alphabetic Cipher Encrypted Message: Rtelzw
Mono-alphabetic Cipher Decrypted Message: Tushar
```