Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

Aim : 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**

Code :

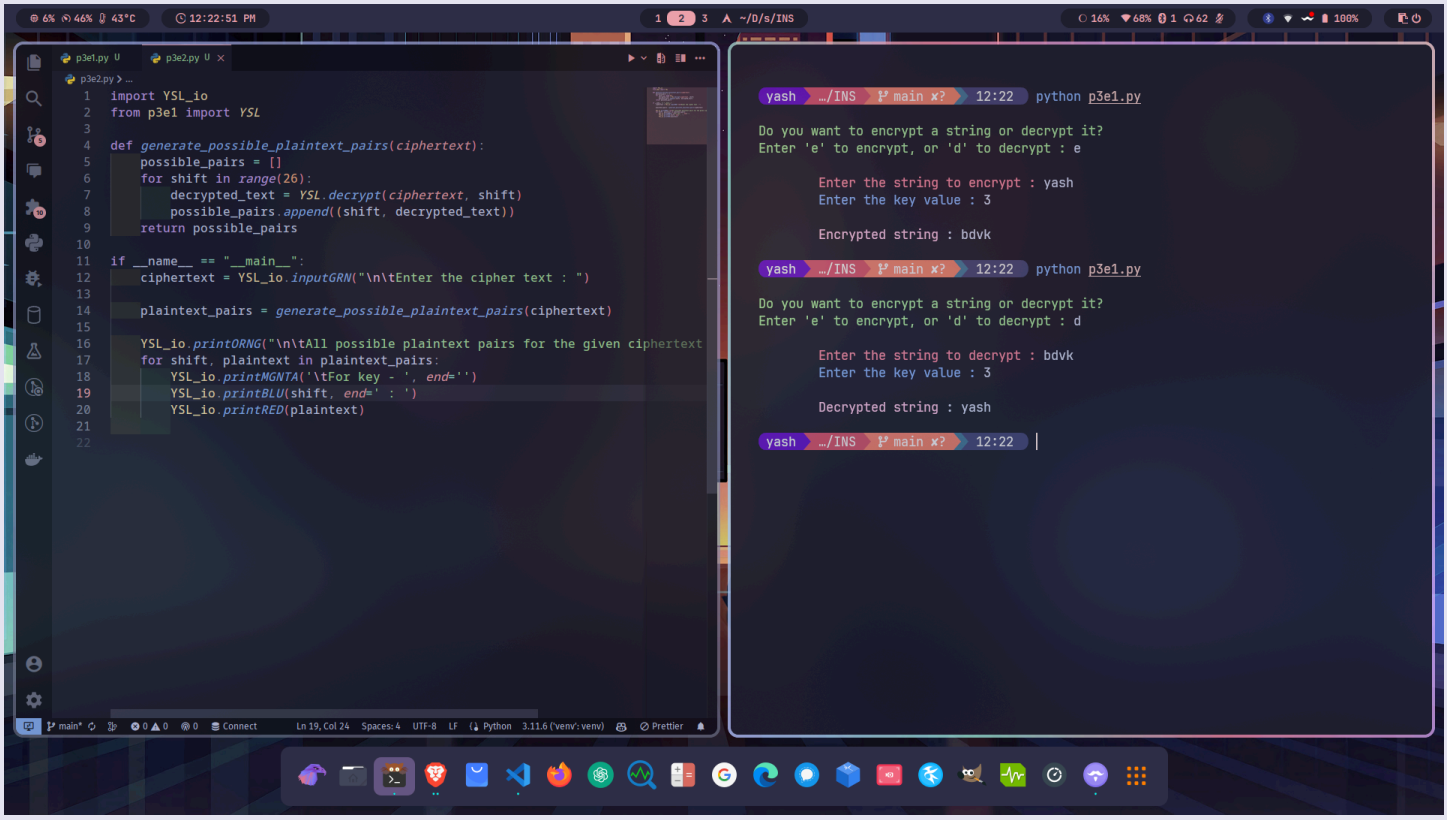
```
import YSL_io


class YSL:
    @staticmethod
    def encrypt(txt, shift: int):
        ncrptd = ""
        for char in txt:
            if char.isalpha():
                if char.isupper():
                    ncrptd += chr((ord(char) + shift - 65) % 26 + 65)
                else:
                    ncrptd += chr((ord(char) + shift - 97) % 26 + 97)
            else:
                ncrptd += char
        return ncrptd

    @staticmethod
    def decrypt(txt, shift):
        return YSL.encrypt(txt, -shift)


def perform_encryption():
    input_string = YSL_io.inputRED("\n\tEnter the string to encrypt : ")
    shift_value = int(YSL_io.inputBLU("\tEnter the key value : "))
    encrypted_string = YSL.encrypt(str(input_string), shift_value)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

```
    YSL_io.printMGNTA("\n\tEncrypted string", end=" : ")
    print(encrypted_string)


def perform_decryption():
    input_string = YSL_io.inputRED("\n\tEnter the string to decrypt : ")
    shift_value = int(YSL_io.inputBLU("\tEnter the key value : "))
    decrypted_string = YSL.decrypt(str(input_string), shift_value)
    YSL_io.printMGNTA("\n\tDecrypted string", end=" : ")
    print(decrypted_string)


if __name__ == "__main__":
    nxt = YSL_io.inputGRN(
        "\nDo you want to encrypt a string or decrypt it?\nEnter 'e' to
encrypt, or 'd' to decrypt : "
    )
    while nxt.lower() not in ["e", "d"]:
        nxt = YSL_io.inputGRN(
            "\nInvalid input!\nEnter 'e' to encrypt, or 'd' to decrypt : "
        )

    if nxt.lower() == "e":
        perform_encryption()

    if nxt.lower() == "d":
        perform_decryption()
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

Output :



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

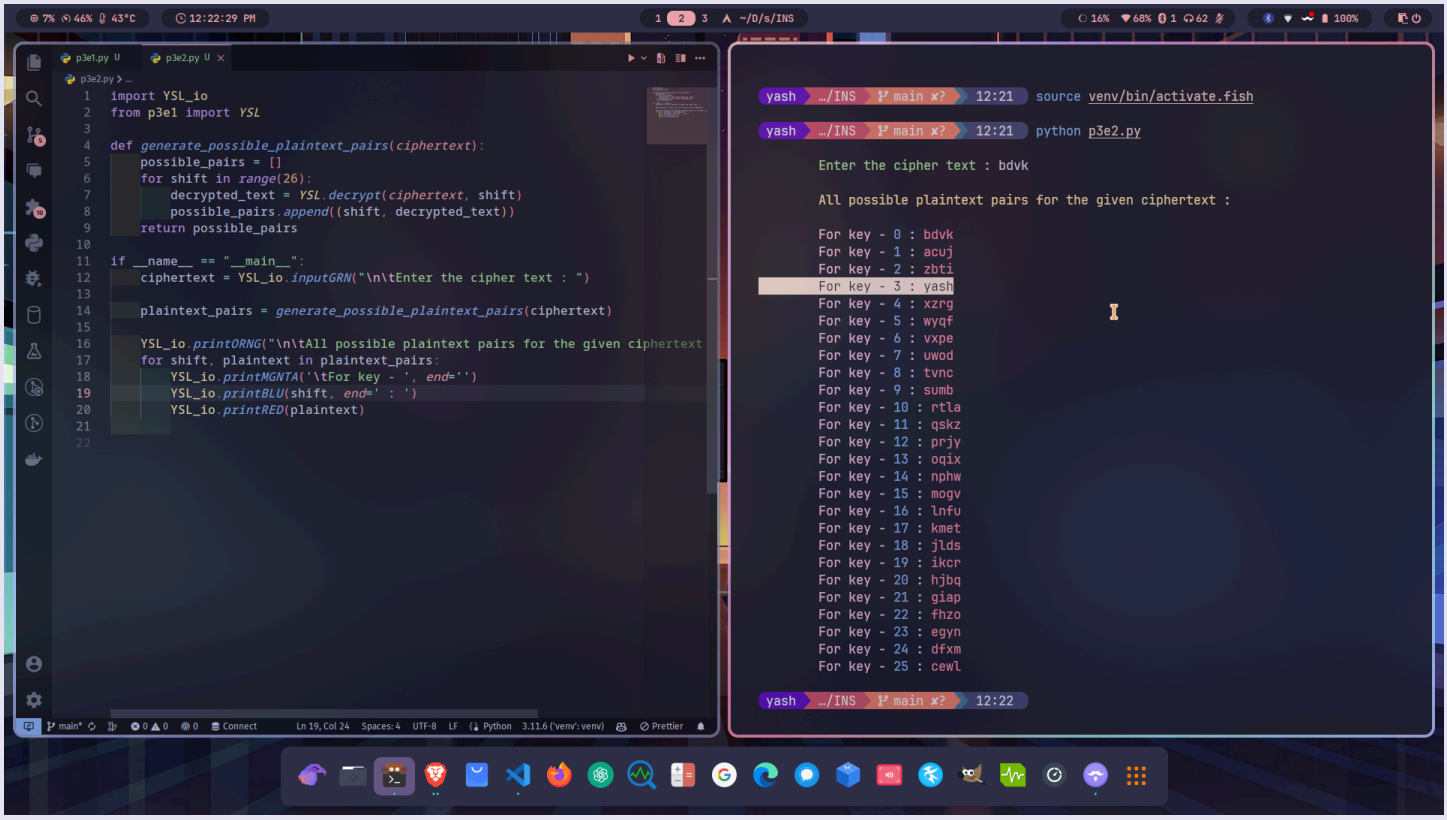
2) Find the all possible Cipher Text & Plaintext pairs

Code :

```
import YSL_io
from p3e1 import YSL

def generate_possible_plaintext_pairs(ciphertext):
   possible_pairs = []
   for shift in range(26):
       decrypted_text = YSL.decrypt(ciphertext, shift)
       possible_pairs.append((shift, decrypted_text))
   return possible_pairs

if __name__ == "__main__":
   ciphertext = YSL_io.inputGRN("\n\tEnter the cipher text : ")

   plaintext_pairs = generate_possible_plaintext_pairs(ciphertext)

   YSL_io.printORNG("\n\tAll possible plaintext pairs for the given
ciphertext : \n")
   for shift, plaintext in plaintext_pairs:
       YSL_io.printMGNTA('\tFor key - ', end='')
       YSL_io.printBLU(shift, end=' : ')
       YSL_io.printRED(plaintext)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

Output :



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

3) Provide Security Mono-alphabetic Cipher Algorithm

Code :

```
import YSL_io
import random
import os


def load_key(key_filename):
    if os.path.exists(key_filename):
        with open(key_filename, "r") as file:
            key = file.read().strip().split()
            YSL_io.printORNG("\n\tKey loaded from file:", end=" ")
            print(" ".join(key))
            return key
    return None


def generate_key(alphabet, key_filename):
    random.shuffle(alphabet)
    with open(key_filename, "w") as file:
        file.write(" ".join(alphabet))
    YSL_io.printORNG("\n\tKey generated and saved to file:", end=" ")
    print(" ".join(alphabet))
    return alphabet


def encrypt(plaintext, key):
    encrypted_text = ""
    for char in plaintext:
        if char.isalpha():
            index = ord(char.upper()) - ord("A")
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

```
            encrypted_char = key[index]
            if char.islower():
                encrypted_char = encrypted_char.lower()
            encrypted_text += encrypted_char
        else:
            encrypted_text += char
    return encrypted_text


def decrypt(ciphertext, key):
    decrypted_text = ""
    for char in ciphertext:
        if char.isalpha():
            index = key.index(char.upper())
            decrypted_char = chr(index + ord("A"))
            if char.islower():
                decrypted_char = decrypted_char.lower()
            decrypted_text += decrypted_char
        else:
            decrypted_text += char
    return decrypted_text


def save_ciphertext(ciphertext, ciphertext_filename):
    with open(ciphertext_filename, "w") as file:
        file.write(ciphertext)
    YSL_io.printRED("\n\tEncrypted text saved to file:", end=" ")
    print(ciphertext_filename)


def main():
    alphabet = list("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
    key_filename = "monoalphabetic_key.txt"
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

```
    ciphertext_filename = "encrypted_text.txt"

    operation = YSL_io.inputGRN(
        "\n\tDo you want to encrypt or decrypt? Enter 'e' or 'd' : "
    )

    if operation.lower() == "e":
        plaintext = YSL_io.inputRED("\n\tEnter the string to encrypt : ")

        generate_key_choice = YSL_io.inputBLU(
            "\n\tDo you want to generate a key? (y/n) : "
        )
        if generate_key_choice.lower() == "y":
            key = generate_key(alphabet.copy(), key_filename)
        else:
            key = load_key(key_filename)

        if key:
            encrypted_text = encrypt(plaintext, key)
            YSL_io.printMGNTA("\n\tEncrypted string:", end=" ")
            print(encrypted_text)

            save_ciphertext(encrypted_text, ciphertext_filename)

    elif operation.lower() == "d":
        ciphertext = YSL_io.inputRED("\n\tEnter the string to decrypt : ")
        key_choice = YSL_io.inputBLU("\n\tDo you want to use an existing
key? (y/n) : ")

        if key_choice.lower() == "y":
            key = load_key(key_filename)
        else:
            key = YSL_io.inputRED(
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

```
                "\n\tEnter the key (space-separated characters) : "
            ).split()

        if key:
            decrypted_text = decrypt(ciphertext, key)
            YSL_io.printMGNTA("\n\tDecrypted string:", end=" ")
            print(decrypted_text)

    else:
        YSL_io.printRED(
            "\n\tInvalid operation. Please enter 'e' for encryption or 'd'
for decryption."
        )


if __name__ == "__main__":
    main()
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 3

Output :