Shivam Pawar
2019230068
Batch C

# LABORATORY

CEL62: Cryptography and System Security
Winter 2021

| | |
|---|---|
| **Experiment 1:** | **Traditional Crypto Methods and Key Exchange** |

Note: Students are advised to read through this lab sheet before doing experiment. On-the-spot evaluation may be carried out during or at the end of the experiment. Your performance, teamwork/Personal effort, and learning attitude will count towards the marks.

Traditional Crypto Methods and Key exchange/PV

# Experiment 1: Traditional Crypto Methods and Key Exchange

## 1 OBJECTIVE

This experiment will be in two parts:

1) To implement Substitution, ROT 13, Transposition, Double Transposition, and Vernam Cipher in Scilab/C/Python/R. 2) Implement Diffie Hellman key exchange algorithm in Scilab/C/Python/R.

## 2. INTROUCTION TO CRYTO AND RELEVANT ALGORITHMS

<u>Cryptography:</u>
In cryptography, encryption is the process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is encrypted information (in cryptography, referred to as cipher text). In many contexts, the word encryption also implicitly refers to the reverse process, decryption (e.g. “software for encryption” can typically also perform decryption), to make the encrypted information readable again (i.e. to make it unencrypted). Encryption is used to protect data in transit, for example data being transferred via networks (e.g. the Internet, e-commerce), mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. There have been numerous reports of data in transit being intercepted in recent years/ Encrypting data in transit also helps to secure it as it is often difficult to physically secure all access to networks

<u>Substitution Technique:</u>
In cryptography, a substitution cipher is a method of encryption by which units of plaintext are replaced with ciphertext according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

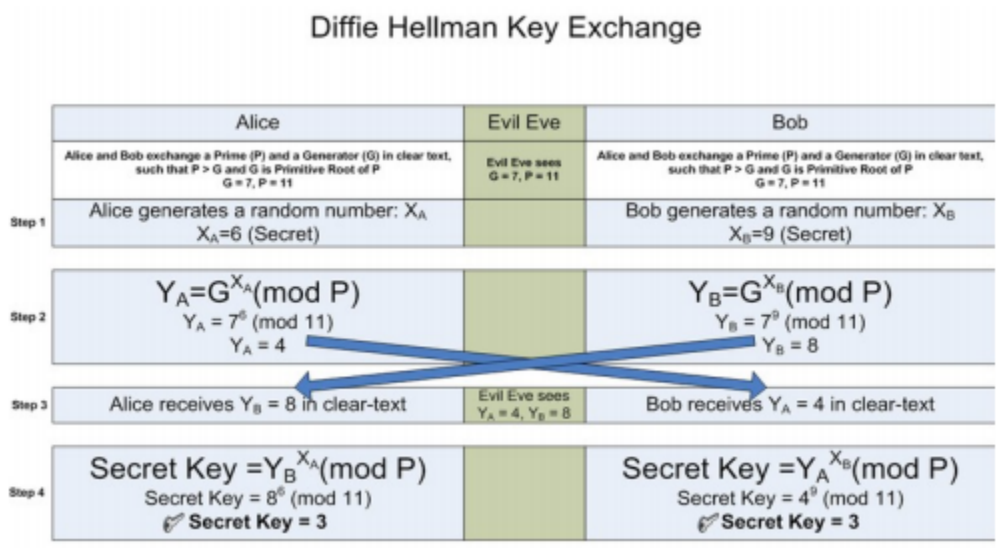
There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher; a cipher that operates on larger groups of letters is termed polygraphic. A monoalphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different times in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice-versa.

<u>Transposition Technique:</u>
In cryptography, a transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed. Mathematically a bijective function is used on the characters' positions to encrypt and an inverse function to decrypt.
In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the

width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".
In a regular columnar transposition cipher, any spare spaces are filled with nulls; in an irregular columnar transposition cipher, the spaces are left blank. Finally, the message is read off in columns, in the order specified by the keyword.
<u>Double Transposition</u>:
A single columnar transposition could be attacked by guessing possible column lengths, writing the message out in its columns (but in the wrong order, as the key is not yet known), and then looking for possible anagrams. Thus to make it stronger, a double transposition was often used. This is simply a columnar transposition applied twice. The same key can be used for both transpositions, or two different keys can be used.

<u>Vernam cipher</u>:
In modern terminology, a Vernam cipher is a symmetrical stream cipher in which the plaintext is XORed with a random or pseudo random stream of data (the "keystream") of the same length to generate the ciphertext. If the keystream is truly random and used only once, this is effectively a one-time pad. Substituting pseudorandom data generated by a cryptographically secure pseudo-random number generator is a common and effective construction for a stream cipher.

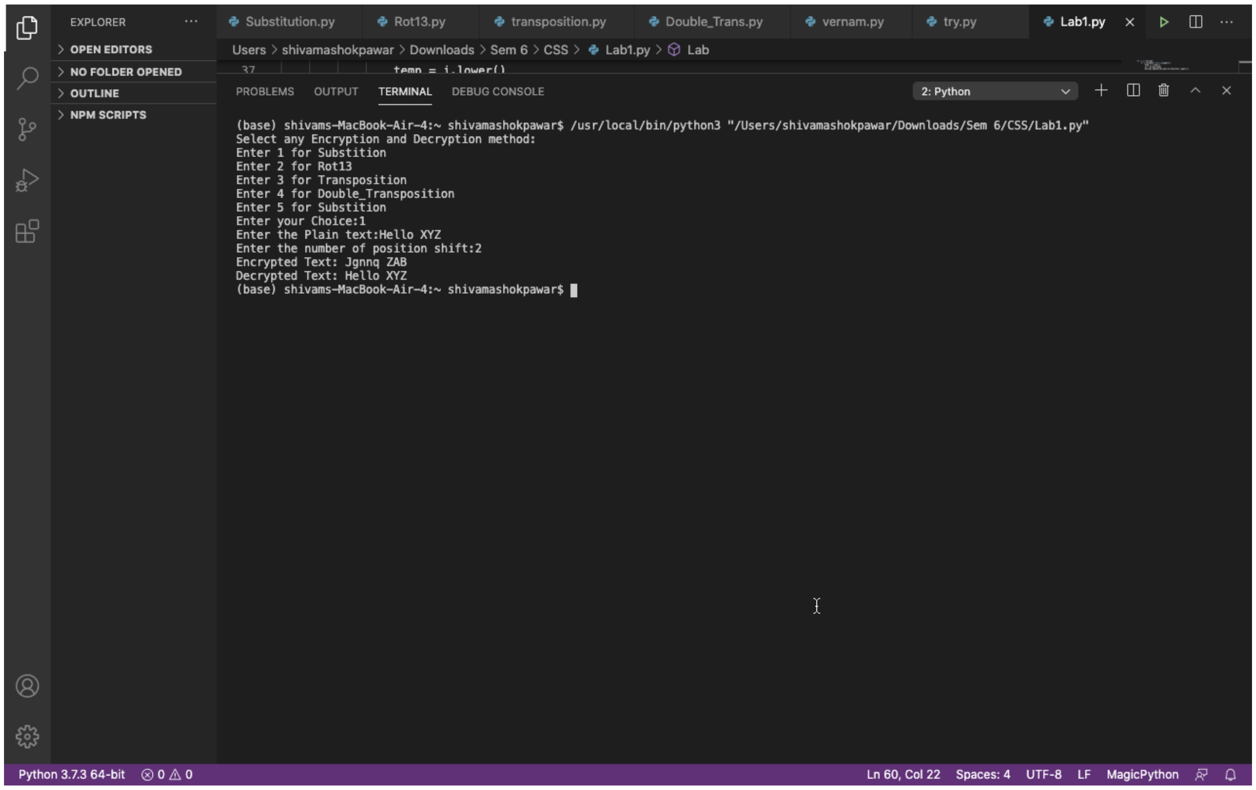
<u>Diffie –Hellman Key exchange algorithm:</u>
The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. Although Diffie–Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).


Diffie Hellman Key Exchange
Alice
Evil Eve
Bob
Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that P > G and G is Primitive Root of P
G = 7, P = 11
Evil Eve sees G = 7, P = 11
Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that P > G and G is Primitive Root of P
G = 7, P = 11
Step 1
Alice generates a random number: $X_A$
$X_A$=6 (Secret)
Bob generates a random number: $X_B$
$X_B$=9 (Secret)
Step 2
$Y_A=G^{X_A}$(mod P)
$Y_A = 7^6$ (mod 11)
$Y_A = 4$
$Y_B=G^{X_B}$(mod P)
$Y_B = 7^9$ (mod 11)
$Y_B = 8$
Step 3
Alice receives $Y_B = 8$ in clear-text
Evil Eve sees $Y_A = 4$, $Y_B = 8$
Bob receives $Y_A = 4$ in clear-text
Step 4
Secret Key =$Y_B^{X_A}$(mod P)
Secret Key = $8^6$ (mod 11)
Secret Key = 3
Secret Key =$Y_A^{X_B}$(mod P)
Secret Key = $4^9$ (mod 11)
Secret Key = 3

## 3 LAB TASKS

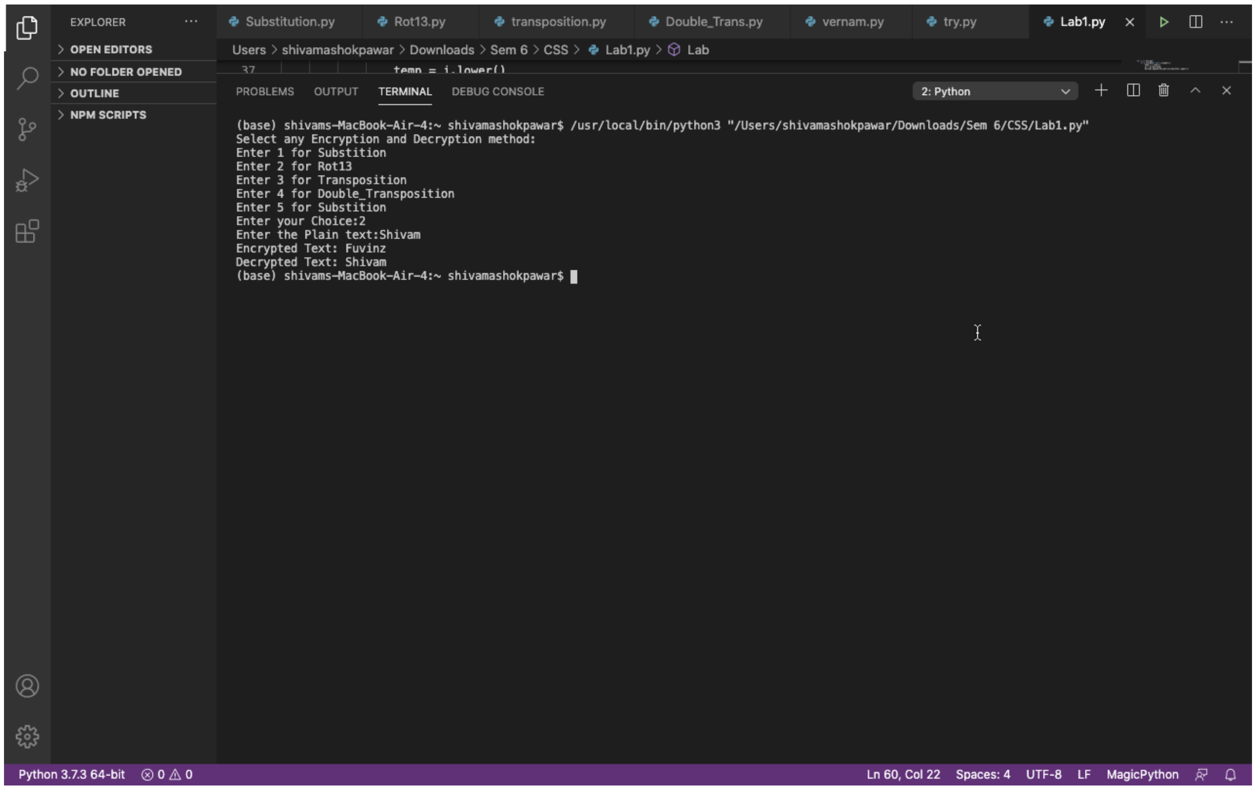
Write a single program which fits all algorithms. YOU should generate output in following manner:

1. Select the Cryptography Method Provide Choice 1…5 for subjected crypto methods
    a. Substitution
        i. Your choice
        ii. Enter Plain text to be encrypted
        iii. Enter the no. of Position shift
        iv. Encrypted Message
        v. Decrypted Message



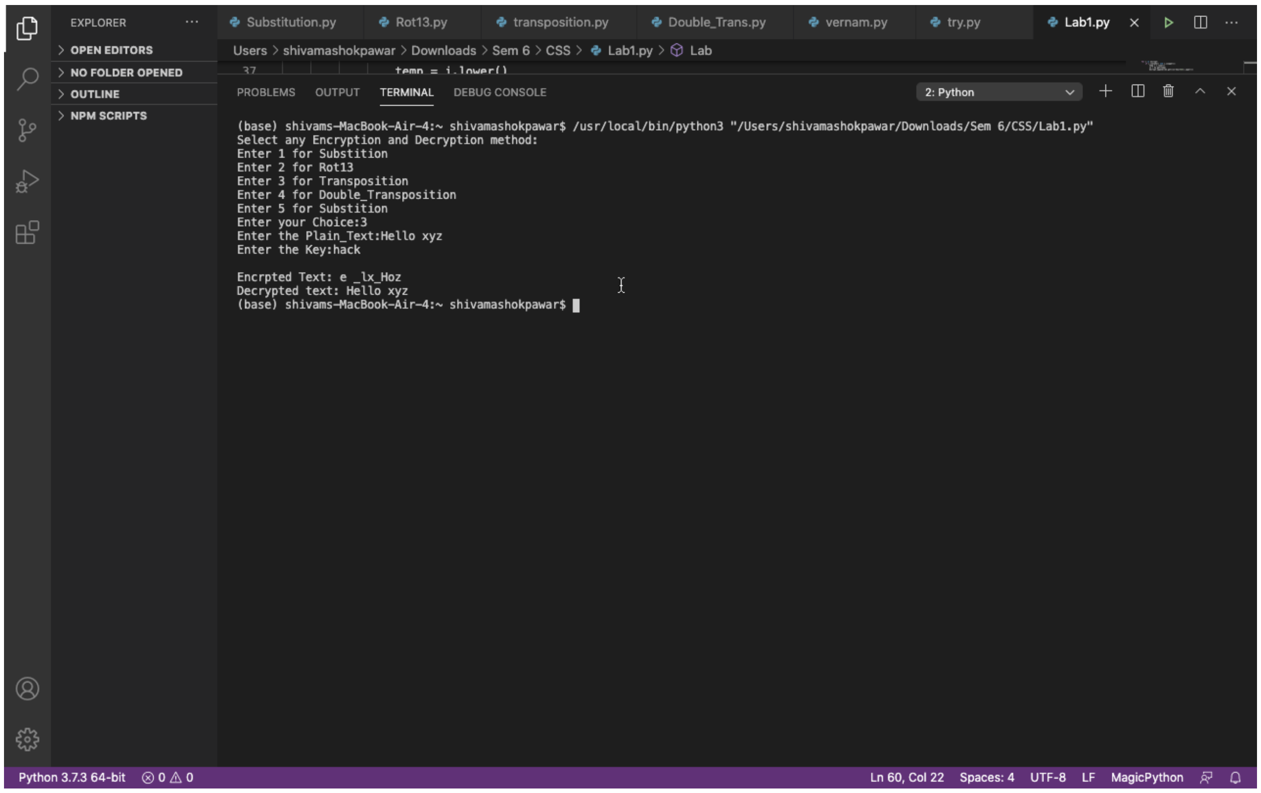
```
(base) shivams-MacBook-Air-4:~ shivamashokpawar$ /usr/local/bin/python3 "/Users/shivamashokpawar/Downloads/Sem 6/CSS/Lab1.py"
Select any Encryption and Decryption method:
Enter 1 for Substition
Enter 2 for Rot13
Enter 3 for Transposition
Enter 4 for Double_Transposition
Enter 5 for Substition
Enter your Choice:1
Enter the Plain text:Hello XYZ
Enter the number of position shift:2
Encrypted Text: Jgnnq ZAB
Decrypted Text: Hello XYZ
(base) shivams-MacBook-Air-4:~ shivamashokpawar$
```

b. ROT 13
    i. Your choice
    ii. Enter Plain text to be encrypted
    iii. Encrypted Message
    iv. Decrypted Message



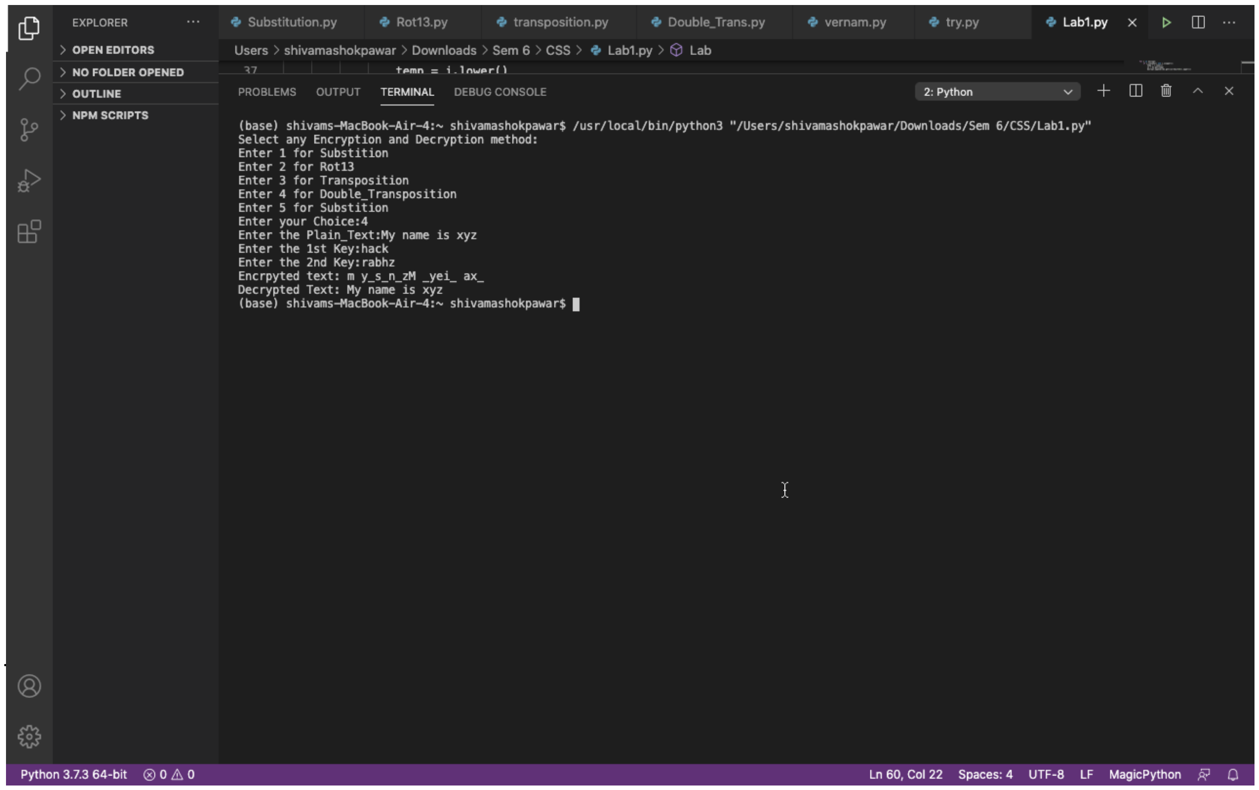
```
(base) shivams-MacBook-Air-4:~ shivamashokpawar$ /usr/local/bin/python3 "/Users/shivamashokpawar/Downloads/Sem 6/CSS/Lab1.py"
Select any Encryption and Decryption method:
Enter 1 for Substition
Enter 2 for Rot13
Enter 3 for Transposition
Enter 4 for Double_Transposition
Enter 5 for Substition
Enter your Choice:2
Enter the Plain text:Shivam
Encrypted Text: Fuvinz
Decrypted Text: Shivam
(base) shivams-MacBook-Air-4:~ shivamashokpawar$
```

c. Transpose
    i. Your choice
    ii. Enter Plain text to be encrypted
    iii. Encrypted Message
    iv. Decrypted Message



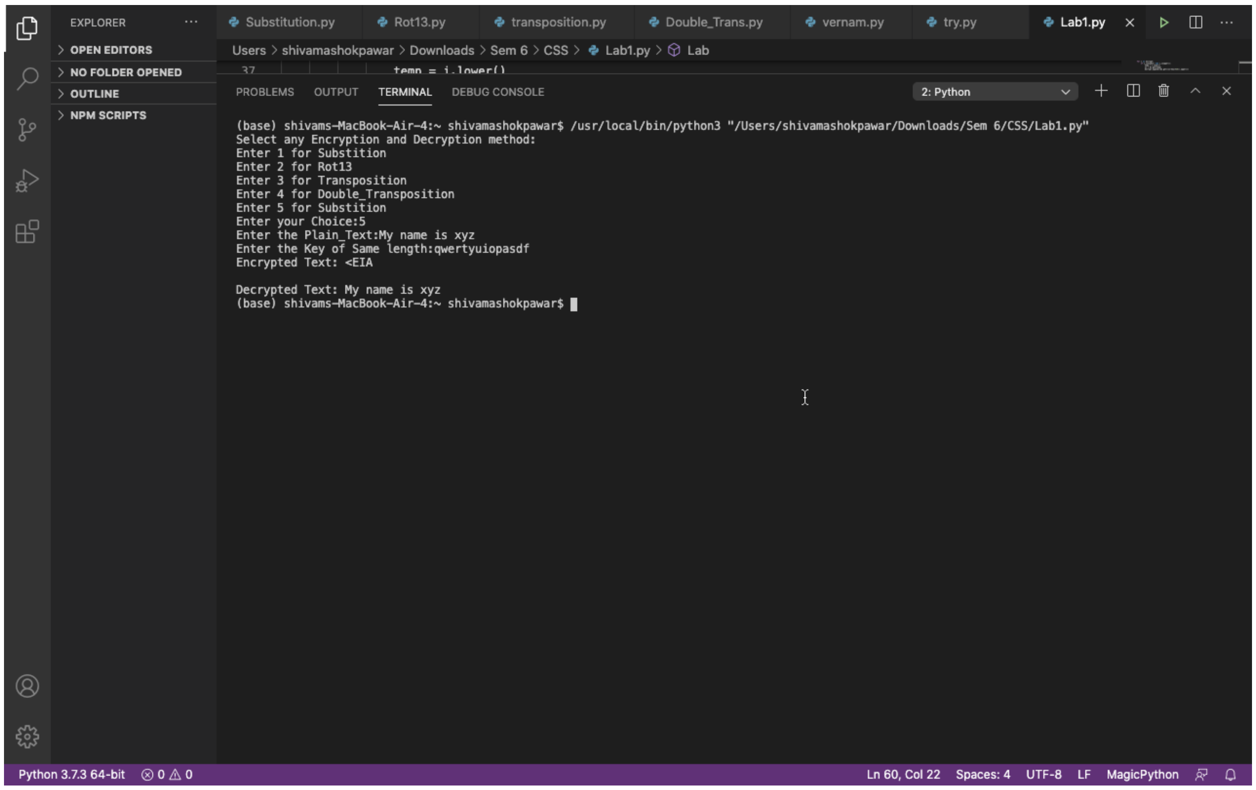
```
(base) shivams-MacBook-Air-4:~ shivamashokpawar$ /usr/local/bin/python3 "/Users/shivamashokpawar/Downloads/Sem 6/CSS/Lab1.py"
Select any Encryption and Decryption method:
Enter 1 for Substition
Enter 2 for Rot13
Enter 3 for Transposition
Enter 4 for Double_Transposition
Enter 5 for Substition
Enter your Choice:3
Enter the Plain_Text:Hello xyz
Enter the Key:hack

Encrpted Text: e _lx_Hoz
Decrypted text: Hello xyz
(base) shivams-MacBook-Air-4:~ shivamashokpawar$
```

d. Double Transposition
   i. Your choice
   ii. Enter Plain text to be encrypted
   iii. Encrypted Message
   iv. Decrypted Message



```
(base) shivams-MacBook-Air-4:~ shivamashokpawar$ /usr/local/bin/python3 "/Users/shivamashokpawar/Downloads/Sem 6/CSS/Lab1.py"
Select any Encryption and Decryption method:
Enter 1 for Substition
Enter 2 for Rot13
Enter 3 for Transposition
Enter 4 for Double_Transposition
Enter 5 for Substition
Enter your Choice:4
Enter the Plain_Text:My name is xyz
Enter the 1st Key:hack
Enter the 2nd Key:rabhz
Encrpyted text: m y_s_n_zM _yei_ ax_
Decrypted Text: My name is xyz
(base) shivams-MacBook-Air-4:~ shivamashokpawar$
```

e. Vernam Cipher
    i. Your choice
    ii. Enter Plain text to be encrypted
    iii. Input Key
    iv. Encrypted Message
    v. Decrypted Message



```
(base) shivams-MacBook-Air-4:~ shivamashokpawar$ /usr/local/bin/python3 "/Users/shivamashokpawar/Downloads/Sem 6/CSS/Lab1.py"
Select any Encryption and Decryption method:
Enter 1 for Substition
Enter 2 for Rot13
Enter 3 for Transposition
Enter 4 for Double_Transposition
Enter 5 for Substition
Enter your Choice:5
Enter the Plain_Text:My name is xyz
Enter the Key of Same length:qwertyuiopasdf
Encrypted Text: <EIA

Decrypted Text: My name is xyz
(base) shivams-MacBook-Air-4:~ shivamashokpawar$
```

f. Diffie Hellman
    i. Enter the Prime Number g:
    ii. Enter second Prime Number n:
    iii. Enter the Secret x:
    iv. Enter the Secret y
    v. $K_1$:
    vi. $K_2$:

**Code:**

```
def Lab(i):
    if i == 1:
        import string
        letters = string.ascii_lowercase
        d1={}
```

```
d2={}
encrypt=[]
decrypt=[]

for i in range(26):
    d1[letters[(i)%26]] = i
for i in range(26):
    d2[i] = letters[(i)%26]


plain_text = input("Enter the Plain text:")
key = int(input("Enter the number of position shift:"))




for i in plain_text:
    if i.isalpha() and i.isupper():
        temp = i.lower()
        x1 = d1.get(temp)
        encrypt.append((d2.get((x1+key)%26)).upper())

    elif i.isalpha() and i.islower():
        x1 = d1.get(i)
        encrypt.append(d2.get((x1+key)%26))

    else:
        encrypt.append(i)

for i in encrypt:
    if i.isalpha() and i.isupper():
        temp = i.lower()
        x1 = d1.get(temp)
        decrypt.append((d2.get((x1-key)%26)).upper())

    elif i.isalpha() and i.islower():
        x1 = d1.get(i)
        decrypt.append(d2.get((x1-key)%26))
    else:
        decrypt.append(i)


print("Encrypted Text: ",end='')
for i in encrypt:
    print(i,end='')
```

```
        print()

        print("Decrypted Text: ",end='')
        for i in decrypt:
            print(i,end='')
        print()

    elif i == 2:

        import string
        letters = string.ascii_lowercase
        d1={}
        d2={}
        encryptRot=[]
        decryptRot=[]

        for i in range(26):
            d1[letters[(i)%26]] = i
        for i in range(26):
            d2[i] = letters[(i)%26]

        plain_text = input("Enter the Plain text:")

        for i in plain_text:
            if i.isalpha() and i.isupper():
                temp = i.lower()
                x1 = d1.get(temp)
                encryptRot.append((d2.get((x1+13)%26)).upper())

            elif i.isalpha() and i.islower():
                x1 = d1.get(i)
                encryptRot.append(d2.get((x1+13)%26))

            else:
                encryptRot.append(i)

        for i in encryptRot:
            if i.isalpha() and i.isupper():
                temp = i.lower()
                x1 = d1.get(temp)
                decryptRot.append((d2.get((x1-13)%26)).upper())

            elif i.isalpha() and i.islower():
                x1 = d1.get(i)
                decryptRot.append(d2.get((x1-13)%26))
            else:
                decryptRot.append(i)


        print("Encrypted Text: ",end='')
        for i in encryptRot:
            print(i,end='')
        print()

        print("Decrypted Text: ",end='')
        for i in decryptRot:
            print(i,end='')
        print()


    elif i == 3:
        import math
        import numpy as np
        encryptTrans=""
        decryptTrans1=""
        decryptTrans=[]

        plain_text= input("Enter the Plain_Text:")
        k = input("Enter the Key:")
        pt_len = len(plain_text)
        col= len(k)

        rows = int(math.ceil(pt_len/col))
        count = (rows*col) - pt_len


        #Encryption

        for i in range(count):
            plain_text=plain_text+"_"


        key_sorted = sorted(list(k))
        pt = list(plain_text)

        arr = np.array(pt)
        newarr =arr.reshape(rows,col)


        for i in range(col):
            position = k.index(key_sorted[i])
            temp = ''.join(map(str, newarr[:,position]))
            encryptTrans = encryptTrans + temp


        print()
        print("Encrpted Text:",encryptTrans[:pt_len])

        #Decryption

        enc_lst = list(encryptTrans)
        temp1 = []
        decp = []
        co=0
        co1=0
        for i in range(rows):
            temp1 = []
            for j in range(col):
                temp1.append(" ")
            decp.append(temp1)

        for i in range(col):
            position = k.index(key_sorted[co1])
            for j in range(rows):
                decp[j][position] = enc_lst[co]
                co += 1
            co1 += 1

        for i in range(rows):
            temp = ''.join(map(str, newarr[i,:]))
            decryptTrans1 = decryptTrans1 + temp

        c = decryptTrans1.count("_")
        print("Decrypted text:",decryptTrans1[:-c])


    elif i == 4:
        import math
        import numpy as np

        def Encryption(text,key):
            encryptTrans=""
            pt_len = len(text)
            col= len(key)
            rows = int(math.ceil(pt_len/col))
            count = (rows*col) - pt_len

            for i in range(count):
                text=text+"_"

            key_sorted = sorted(list(key))
            pt = list(text)
            arr = np.array(pt)
            newarr =arr.reshape(rows,col)


            for i in range(col):
                position = key.index(key_sorted[i])
                temp = ''.join(map(str, newarr[:,position]))
                encryptTrans = encryptTrans + temp


            return(encryptTrans,newarr)

        def Decryption(text,key,dec_arr):
            decryptTrans1=""
            pt_len = len(text)
            col= len(key)
            rows = int(math.ceil(pt_len/col))
            count = (rows*col) - pt_len
            key_sorted = sorted(list(key))

            count = (rows*col) - pt_len

            for i in range(count):
                text=text+" "

            enc_lst = list(text)
            temp1 = []
            decp = []
            co=0
            co1=0
            for i in range(rows):
                temp1 = []
                for j in range(col):
                    temp1.append(" ")
                decp.append(temp1)

            for i in range(col):
                position = key.index(key_sorted[co1])
```

```
            for j in range(rows):
                decp[j][position] = enc_lst[co]
                co += 1
            co1 += 1

        for i in range(rows):
            temp = ''.join(map(str, dec_arr[i,:]))
            decryptTrans1 = decryptTrans1 + temp

        c = decryptTrans1.count("_")
        return(decryptTrans1[:-c])

    plain_text= input("Enter the Plain_Text:")
    k1 = input("Enter the 1st Key:")
    k2 = input("Enter the 2nd Key:")

    encrypted_plainText,enc_arr = Encryption(plain_text,k1)
    double_encryption,doub_arr = Encryption(encrypted_plainText,k2)


    print("Encrpyted text:",double_encryption)


    f_d = Decryption(double_encryption,k2,doub_arr)
    s_d = Decryption(f_d,k1,enc_arr)


    print("Decrypted Text:",s_d)


  elif i == 5:
    plain_text= input("Enter the Plain_Text:")
    key_vernam = input("Enter the Key of Same length:")

    encryption_vernam=""


    for i in range(len(plain_text)):
        encryption_vernam = encryption_vernam + chr(ord(plain_text[i]) ^ ord(key_vernam[i]))

    print("Encrypted Text:", encryption_vernam)

    decryption_vernam=""
    for i in range(len(encryption_vernam)):
                decryption_vernam = decryption_vernam + chr(ord(encryption_vernam[i]) ^
ord(key_vernam[i]))
        print("Decrypted Text:",decryption_vernam)



if __name__ == "__main__":
    print("Select any Encryption and Decryption method:")
    print("Enter 1 for Substition")
    print("Enter 2 for Rot13")
    print("Enter 3 for Transposition")
    print("Enter 4 for Double_Transposition")
    print("Enter 5 for Substition")
    x=int(input("Enter your Choice:"))
    Lab(x)
```

**Observation:**
1.In Substitution method we replace every character with the character that would be obtained after adding the key value to it.

2.In Rot-13 we replace the character with next 13th character to get the encrypted text. If the value is greater then 26 then we use mod to get the proper alphabet.

3. In Transposition we do not change any character but we only change the position of that character with the help of the key by taking characters column wise and decrypt it with properly arranging the columns with help of key.

4. In Double Transposition we perform columnar transposition twice. We can either take same key or different key.

5.In Vernam Cipher the key need to be of same length as of text. We need to use same key for encryption and decryption. We Xor the plain text with key and we get the encrypted text. Similarly to decrypt the text we xor it with the same key.

## 4 SUBMISSION

You need to submit a detailed lab report to describe what you have done and what you have observed as per the suggested output format for all method ; you also need to provide explanation to the observations that are interesting or surprising. In your report, you need to answer all the questions as per the suggested formant listed above.