Experiment – 1 Implementation of Product cipher using substitution and transposition cipher

Name:	**Shaikh Shadab**	**Rollno**	**:**	**17DCO74**
Class :	**TE.CO**	**Batch**	**:**	**B3**

#Source code Implementation in python

```
__author__ = 'Shadab Shaikh'
__title__ = 'Implementation of Product Cipher Using Mono-alphabetic Ceaser Cipher and
columnnar transposition cipher'
__date__ = '10-01-2019'
__version__ = '2.0'

print('Author          : ' + __author__)
print('Title           : ' + __title__)
print('Date            : ' + __date__)
print('Version         : ' + __version__)

plaintext=input("\n\nEnter the plain text\n")          #Accepting input in string format
plaintext=plaintext.replace(" ", "")                   #Removing all whitespaces
caeserkey=int(input("\nEnter the key\n"))              #Accepting input in numeric format
asci=[]                                                #Contains the ascii for each alphabet
caeser=[]                                              #Contains generated caeser cipher
char=""                                                #Used as a pointer for scanning alphabet
transpos=[]                                            #Contains generated transposition cipher
flist=[]                                               #Contains the final output
matrix2=[]                                  #Contains generated transposition cipher column wise
matrix3=[]                              #Contains generated transposition cipher sorted as per key

def matrixcolumn(i,j,trpklen):              #Function that traverse transpos matrix column wise
        matrix2.append(matrix[j][i])
        j=j+1
        if j<trpklen:
                matrixcolumn(i,j,trpklen)
        else:
                j=0
                i=i+1
                if i<trpklen:
                        matrixcolumn(i,j,trpklen)
        return matrix2

def printenct(matrix3,x,trpklen):  #Function that removes transpos key element from matrix
        matrix3[x].pop(0)
        x=x+1
        if x<trpklen:
```

```
        printenct(matrix3,x,trpklen)
    else:
        convstr(matrix3)                #Calling another function

def convstr(matrix3):                   #Function that flattens the list element
    for z in matrix3:
        if type(z) == list:
            convstr(z)
        else:
            flist.append(z)

for i in plaintext:
    asci.append(ord(i))                 #Getting ascii value for each character

for j in range(len(asci)):
    char=chr(asci[j])
    #converting ascii into character
    if char.isupper():                  #Checking for lower and upper case
        temp=(asci[j]+caeserkey-65)%26+65              #65 for upper case
    else:
        temp=(asci[j]+caeserkey-97)%26+97              #97 for lower case
    caeser.append(chr(temp))            #Generating caeser cipher for each alphabet and
                                        #storing in caeser mat


transposkey=input("\n\nEnter the key for transposition cipher\n")    #accepting transposition
key value in number format
transposkey=transposkey.replace(" ", "")                  #removing all whitespaces
trpklen=len(transposkey)                          #finding the length of transpos key

for k in transposkey:
    transpos.append(k)     #Inserting transposkey value into 0th index of transpos mat

for l in range(len(caeser)):
    transpos.append(caeser[l])          #Inserting/Appending generated caeser cipher
                                        #into transpos mat

#printing the transpos mat traversing column wise
matrix = [transpos[m:m+trpklen] for m in range(0,len(transpos),trpklen)]

for o in matrix:
    while (len(o)<trpklen):
        o.append('X')           #Appending character X if some indexes are left out

matrix2=matrixcolumn(0,0,trpklen)   #Calling matrixcolumn function
```

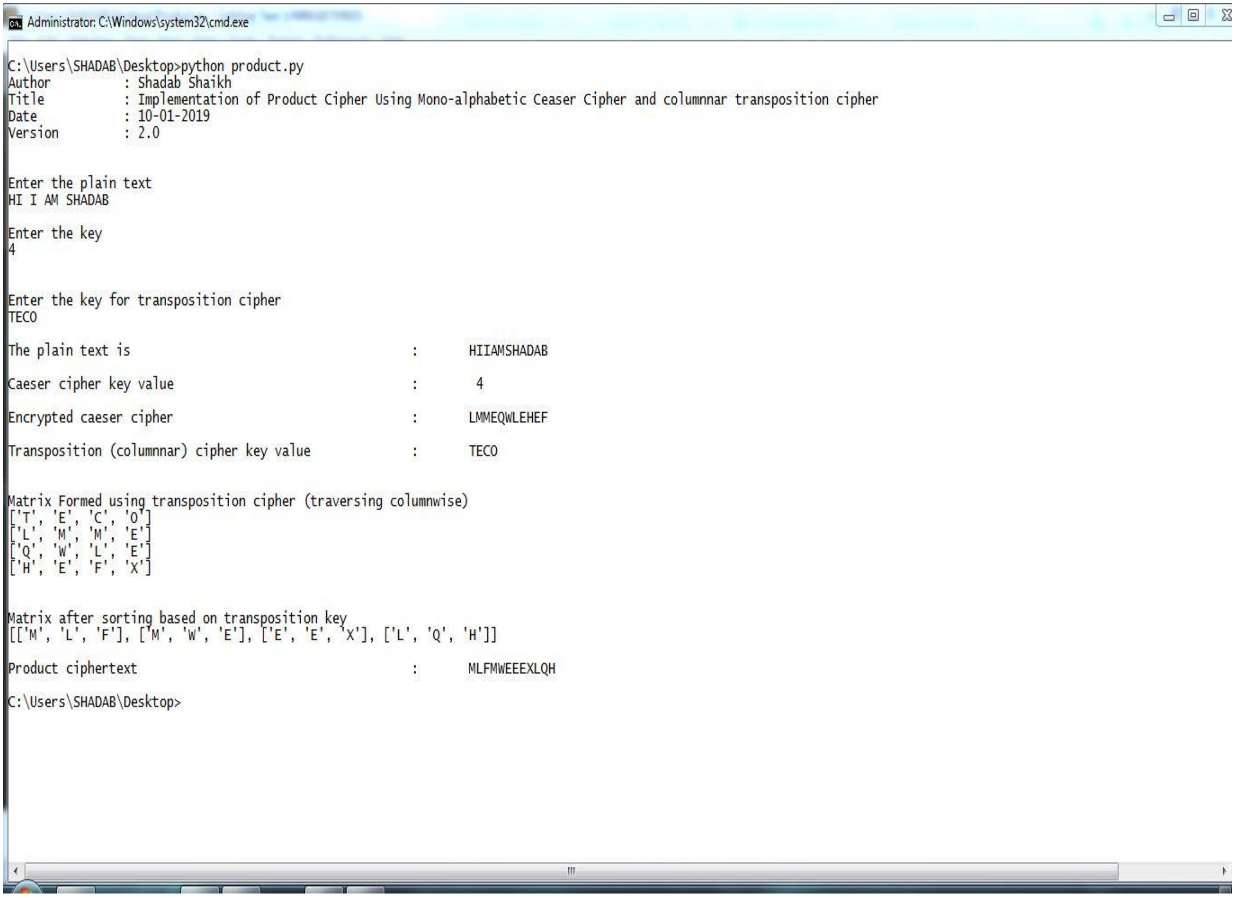
```
#Storing the traverse column mat into matrix3
matrix3 = [matrix2[m:m+trpklen] for m in range(0,len(matrix2),trpklen)]

matrix3.sort()                          #Sorting on the basis of transpos key

printenct(matrix3,0,trpklen)            #Calling printenct  function
caeserciphertext = ''.join(caeser)#Converting matrix list into string for displaying purpose
productciphertext = ''.join(flist)#Converting final output list into string for displaying purpose

#-------------------------------------Displaying Block-----------------------------------------------#
print('\nThe plain text is                                  :        '+plaintext)
print('\nCaeser cipher key value                                    :        ',caeserkey)
print('\nEncrypted caeser cipher                                    :        '+caeserciphertext)
print('\nTransposition (columnnar) cipher key value         :        '+transposkey)
print('\n\nMatrix Formed using transposition cipher (traversing columnwise)       ')
print(*matrix, sep='\n')
print('\n\nMatrix after sorting based on transposition key')
print(matrix3)
print('\nProduct ciphertext                                 :        '+productciphertext)
```

#Output



```
C:\Users\SHADAB\Desktop>python product.py
Author          : Shadab Shaikh
Title           : Implementation of Product Cipher Using Mono-alphabetic Ceaser Cipher and columnnar transposition cipher
Date            : 10-01-2019
Version         : 2.0


Enter the plain text
HI I AM SHADAB

Enter the key
4


Enter the key for transposition cipher
TECO

The plain text is                                  :       HIIAMSHADAB

Caeser cipher key value                            :        4

Encrypted caeser cipher                            :       LMMEQWLEHEF

Transposition (columnnar) cipher key value         :       TECO


Matrix Formed using transposition cipher (traversing columnwise)
['T', 'E', 'C', 'O']
['L', 'M', 'M', 'E']
['Q', 'W', 'L', 'E']
['H', 'E', 'F', 'X']


Matrix after sorting based on transposition key
[['M', 'L', 'F'], ['M', 'W', 'E'], ['E', 'E', 'X'], ['L', 'Q', 'H']]

Product ciphertext                                 :       MLFMWEEEXLQH

C:\Users\SHADAB\Desktop>
```