**Name :- Kaustubh Shrikant Kabra**

**CLass:- TE Computer**

**ERP :-38**

**Subject :-LP2(IS) (Transposition)**

**Code:-**

import math  
   
key = "HACK"  
   
# Encryption  
def encryptMessage(msg):  
 cipher = ""  
   
 # track key indices  
 k\_indx = 0  
   
 msg\_len = float(len(msg))  
 msg\_lst = list(msg)  
 key\_lst = sorted(list(key))  
   
 # calculate column of the matrix  
 col = len(key)  
   
 # calculate maximum row of the matrix  
 row = int(math.ceil(msg\_len / col))  
   
 # add the padding character '\_' in empty  
 # the empty cell of the matix   
 fill\_null = int((row \* col) - msg\_len)  
 msg\_lst.extend('\_' \* fill\_null)  
   
 # create Matrix and insert message and   
 # padding characters row-wise   
 matrix = [msg\_lst[i: i + col]   
 for i in range(0, len(msg\_lst), col)]  
   
 # read matrix column-wise using key  
 for \_ in range(col):  
 curr\_idx = key.index(key\_lst[k\_indx])  
 cipher += ''.join([row[curr\_idx]   
 for row in matrix])  
 k\_indx += 1  
   
 return cipher  
   
# Decryption  
def decryptMessage(cipher):  
 msg = ""  
   
 # track key indices  
 k\_indx = 0  
   
 # track msg indices  
 msg\_indx = 0  
 msg\_len = float(len(cipher))  
 msg\_lst = list(cipher)  
   
 # calculate column of the matrix  
 col = len(key)  
   
 # calculate maximum row of the matrix  
 row = int(math.ceil(msg\_len / col))  
   
 # convert key into list and sort   
 # alphabetically so we can access   
 # each character by its alphabetical position.  
 key\_lst = sorted(list(key))  
   
 # create an empty matrix to   
 # store deciphered message  
 dec\_cipher = []  
 for \_ in range(row):  
 dec\_cipher += [[None] \* col]  
   
 # Arrange the matrix column wise according   
 # to permutation order by adding into new matrix  
 for \_ in range(col):  
 curr\_idx = key.index(key\_lst[k\_indx])  
   
 for j in range(row):  
 dec\_cipher[j][curr\_idx] = msg\_lst[msg\_indx]  
 msg\_indx += 1  
 k\_indx += 1  
   
 # convert decrypted msg matrix into a string  
 try:  
 msg = ''.join(sum(dec\_cipher, []))  
 except TypeError:  
 raise TypeError("This program cannot",  
 "handle repeating words.")  
   
 null\_count = msg.count('\_')  
   
 if null\_count > 0:  
 return msg[: -null\_count]  
   
 return msg  
   
# Driver Code  
  
msg = (input("Enter Message: "))  
  
   
cipher = encryptMessage(msg)  
print("Encrypted Message: {}".  
 format(cipher))  
   
print("Decryped Message: {}".  
 format(decryptMessage(cipher)))

**Output:-**

Enter Message: Its KK29 aka Kaustubh

Encrypted Message: tKaKt\_s2kau\_IK sh 9aub\_

Decryped Message: Its KK29 aka Kaustubh

Process finished with exit code 0

