PRACTICAL 7

AIM: Implement One time pad encryption-decryption.

Code:

```
def char_to_int(text):

    l1 = []
    l1.clear()
    for char in text:
        if char.isalpha():
            if char.isupper():
                l1.append(ord(char) - 65)
            else:
                l1.append(ord(char) - 97)
    return l1
def int_to_chat(number_list):
    l1 = []
    for integer in number_list:
        l1.append(chr(integer + 97))
    return l1


def otp_cipher_encoding(text,key):

    plain_text_int = char_to_int(text)
    key_text_int = char_to_int(key)
    l1 = []

    if(len(plain_text_int) == len(key_text_int)):
        for i in range(0,len(plain_text_int)):
```

```
        s1 = plain_text_int[i] + key_text_int[i]
        l1.append(s1)

    for i in range(len(l1)):
        if(l1[i] > 25):
            num = l1[i] - 26
            l1[i] = num
    encoing = "".join(int_to_chat(l1))
    return encoing

def otp_cipher_decoding(text,key):

    decoded_int = char_to_int(text)
    key_decoded_int = char_to_int(key)
    l2 = []
    if(len(decoded_int) == len(key_decoded_int)):
        for i in range(0,len(decoded_int)):
            s2 = decoded_int[i] - key_decoded_int[i]
            l2.append(s2)
    if(l2[i] < 0 ):
        l2[i] = l2[i] + 26

    decoing = "".join(int_to_chat(l2))
    return decoing


text = input("enter the plain text: ")

key_text = input("enter the key: ")
```

```
encoded_msg = otp_cipher_encoding(text,key_text)
decoded_msg = otp_cipher_decoding(encoded_msg , key_text)
print("encoded massage: ",encoded_msg)
print("decoded massage: ",decoded_msg)
```

output:

```
PS C:\work\7th sem> python -u "c:\work\7th sem\INS\otp.py"
enter the plain text: xyz
enter the key: abc
encoded massage:  xzb
decoded massage:  xyz
PS C:\work\7th sem>
```