NAME : N. karthik reddy

REG NO : 192111167

COURSE : CRYPTOGRAPHY & NETWORK SECURITY

DAY-1

**QUESTION 1 :**

**Write a High Level program for Caesar  
cipher involves replacing each letter of the alphabet with the letter standing  
k places further down the alphabet, for k in the range 1 through 25.**

PROGRAM:

n = "abcdefghijklmnopqrstuvwxyz"

li, mi = [], []

z, x = " "," "

key = int(input("Enter the key value:"))

s = str(input("Enter the plain text:"))

print("!encryption!")

for i in range(len(s)):

k = n.find(s[i])

k = k + key

m = k % 26

li.append(m)

for i in range(len(li)):

l = li[i]

z = z + n[l]

print(z)

print("!decryption!")

for i in range(len(z)):

o = n.find(z[i])

o = o - key

h = o % 26

mi.append(h)

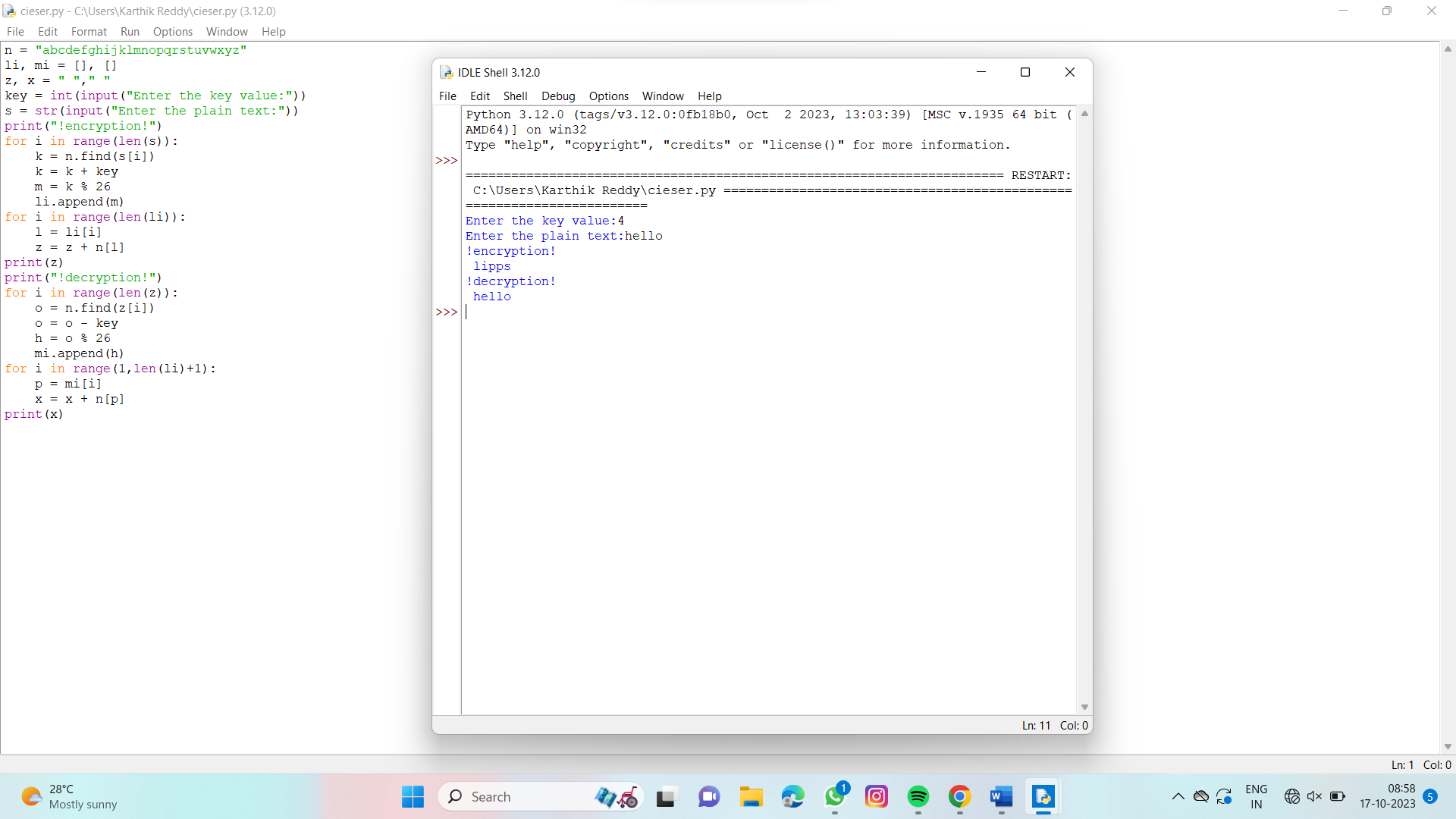
for i in range(1,len(li)+1):

p = mi[i]

x = x + n[p]

print(x)

OUTPUT :



QUESTION 2:

**Write a High level program for monoalphabetic substitution cipher maps a  
plaintext alphabet to a ciphertext alphabet, so that each letter of the  
plaintext alphabet maps to a single unique letter of the ciphertext alphabet.**

**PROGRAM :**

n = "abcdefghijklmnopqrstuvwxyz"

s = str(input("Enter the plain text:"))

p = "lzmynxowpvqurtsjaibhcgdfde"

li = []

a, b = " ", " "

for i in range(len(s)):

k = n.find(s[i])

li.append(k)

for j in range(len(s)):

a = a + p[li[j]]

for j in range(len(s)):

b = b + n[li[j]]

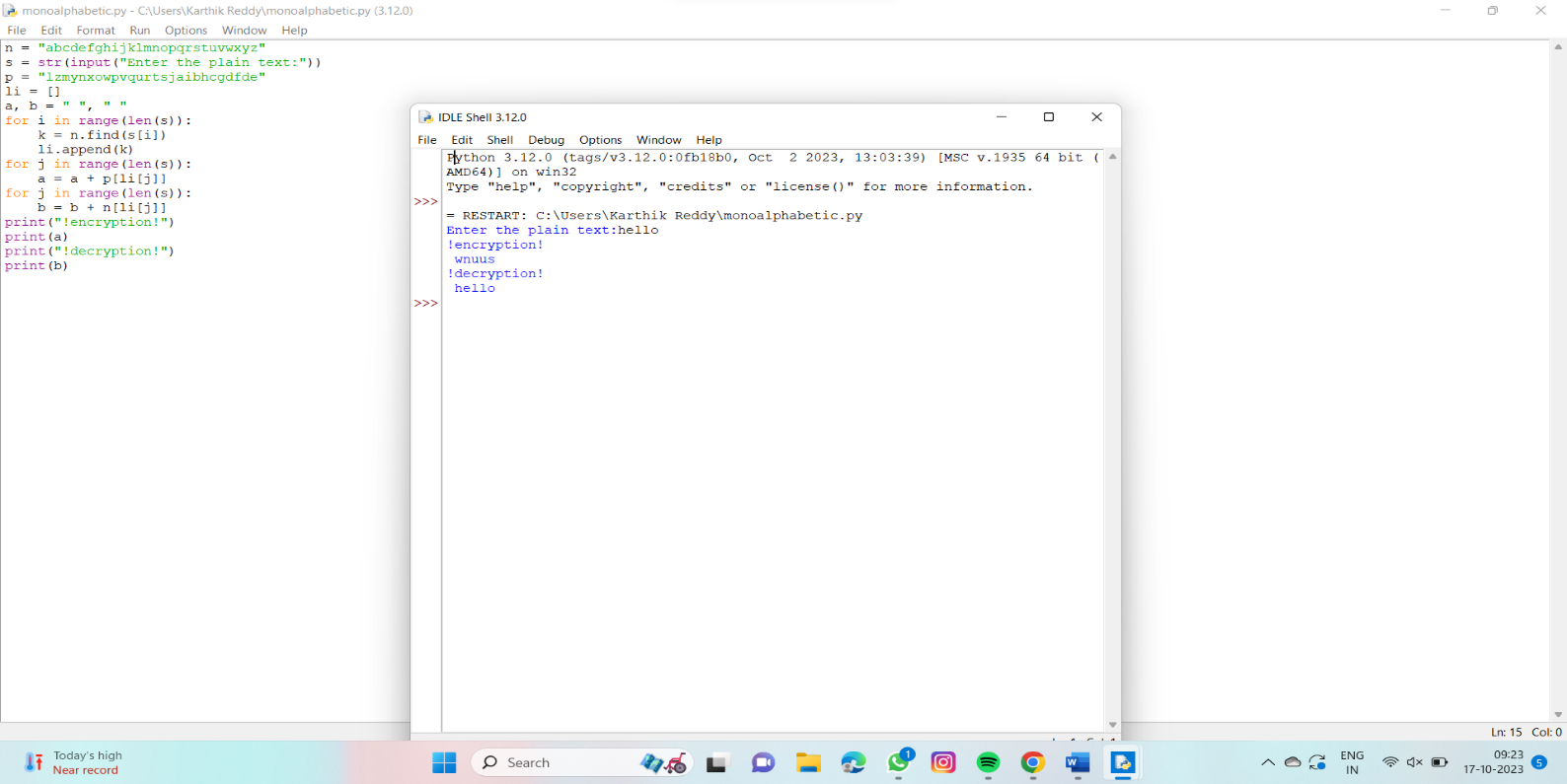
print("!encryption!")

print(a)

print("!decryption!")

print(b)

**OUTPUT :**

****

**QUESTION : 3**

Write a High level program for  
Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed  
using a keyword. Plaintext is encrypted two letters at a time using this  
matrix.

**PROGRAM :**

**"""from textwrap import wrap**

**n = "abcdefghijklmnopqrstuvwxyz"**

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

**key = input("Enter key content:")**

**b = []**

**for i in range(len(key)):**

**k = key[i]**

**b.append(k)**

**li = [\*set(b)]**

**for i in range(len(li)):**

**for j in range(len(li)):**

**if ord(li[i]) < ord(li[j]):**

**li[i], li[j] = li[j], li[i]**

**li1 = []**

**for i in range(len(n)):**

**m = n[i]**

**li1.append(m)**

**for i in range(len(li)):**

**l = n.find(li[i])**

**li1.remove(n[l])**

**for i in range(len(li1)):**

**key = key + li1[i]**

**print(key)**

**z = wrap(key,5)**

**print(z)**

**# alpha\_matrix = [list(key[i:i+5]) for i in range(0, 26, 5)]**

**alpha\_matrix = []**

**for i in range(0, 26, 5):**

**temp\_list = list(key[i: i+5])**

**alpha\_matrix.append(temp\_list)**

**print('\n')**

**for sub\_list in alpha\_matrix:**

**print(\*sub\_list)**

**alpha\_matrix[i][j]**

**alpha\_matrix[0][j]**

**alpha\_matrix[i][0]**

**"""**

**import string**

**def generate\_matrix(key):**

**"""**

**Generates a 5x5 matrix of letters using the given keyword.**

**"""**

**key = key.lower().replace("j", "i")**

**alphabet = string.ascii\_lowercase.replace("j", "")**

**key\_set = set(key)**

**matrix = []**

**for c in key:**

**if c not in matrix:**

**matrix.append(c)**

**for c in alphabet:**

**if c not in key\_set and c not in matrix:**

**matrix.append(c)**

**matrix = [matrix[i:i+5] for i in range(0, 25, 5)]**

**return matrix**

**def encrypt(plaintext, key):**

**"""**

**Encrypts the plaintext using the given key and returns the ciphertext.**

**"""**

**plaintext = plaintext.lower().replace("j", "i").replace(" ", "")**

**matrix = generate\_matrix(key)**

**ciphertext = ""**

**for i in range(0, len(plaintext), 2):**

**a = plaintext[i]**

**b = plaintext[i+1] if i+1 < len(plaintext) else "x"**

**if a == b:**

**b = "x"**

**a\_row, a\_col = get\_position(matrix, a)**

**b\_row, b\_col = get\_position(matrix, b)**

**if a\_row == b\_row:**

**a\_col = (a\_col + 1) % 5**

**b\_col = (b\_col + 1) % 5**

**elif a\_col == b\_col:**

**a\_row = (a\_row + 1) % 5**

**b\_row = (b\_row + 1) % 5**

**else:**

**a\_col, b\_col = b\_col, a\_col**

**ciphertext += matrix[a\_row][a\_col] + matrix[b\_row][b\_col]**

**return ciphertext**

**def decrypt(ciphertext, key):**

**"""**

**Decrypts the ciphertext using the given key and returns the plaintext.**

**"""**

**matrix = generate\_matrix(key)**

**plaintext = ""**

**for i in range(0, len(ciphertext), 2):**

**a = ciphertext[i]**

**b = ciphertext[i+1]**

**a\_row, a\_col = get\_position(matrix, a)**

**b\_row, b\_col = get\_position(matrix, b)**

**if a\_row == b\_row:**

**a\_col = (a\_col - 1) % 5**

**b\_col = (b\_col - 1) % 5**

**elif a\_col == b\_col:**

**a\_row = (a\_row - 1) % 5**

**b\_row = (b\_row - 1) % 5**

**else:**

**a\_col, b\_col = b\_col, a\_col**

**plaintext += matrix[a\_row][a\_col] + matrix[b\_row][b\_col]**

**plaintext = plaintext.replace("x", "")**

**return plaintext**

**def get\_position(matrix, letter):**

**"""**

**Returns the position of the given letter in the matrix.**

**"""**

**for i in range(5):**

**for j in range(5):**

**if matrix[i][j] == letter:**

**return i, j**

**return -1, -1**

**# Example usage**

**key = 'secretkey'**

**plaintext = 'hello world'**

**ciphertext = encrypt(plaintext, key)**

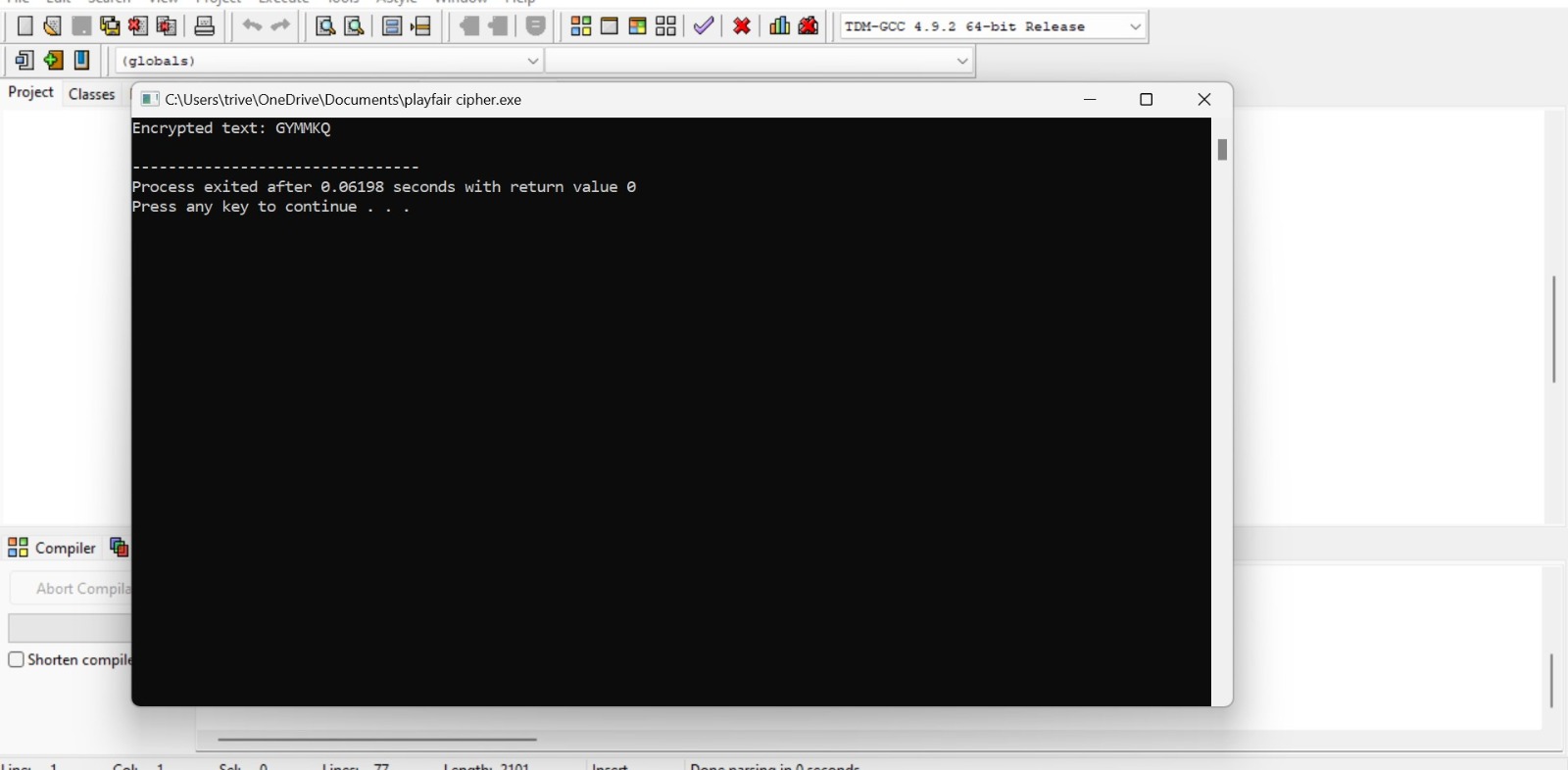
**print("!encryption!")**

**print(ciphertext)**

**print("!decryption!")**

**print(plaintext)**

**OUTPUT :**

****

**QUESTION 4 :**

**Write a High level program for  
polyalphabetic substitution cipher uses a separate   monoalphabetic substitution cipher for each  
successive letter of plaintext, depending on      
a key.**

**PROGRAM :**

key = str(input("Enter key content:"))

plain = str(input("Enter plain text:"))

y, z= " "," "

n = "abcdefghijklmnopqrstuvwxyz"

print("!encryption!")

li1, li2, li3= [], [], []

for i in range(len(key)):

m = n.find(key[i])

li1.append(m)

li4 = []

for j in range(len(plain)):

k = n.find(plain[j])

li4.append(k)

li2 += li1

for i in range(len(plain)):

li2.append(li2[i])

for j in range(len(plain)):

li3.append(li2[j])

#li4 contain plain text alphabet value

#li3 contain key content alphabet value

li5 = []

for i in range(len(plain)):

l = li4[i]+li3[i]

li5.append(l)

#li5 contains key+pt

li6 = []

for i in range(len(li5)):

o = li5[i]%26

li6.append(o)

y = y + n[o]

print(y)

li7 = []

print("!decryption!")

print(plain)

"""for i in range(len(plain)):

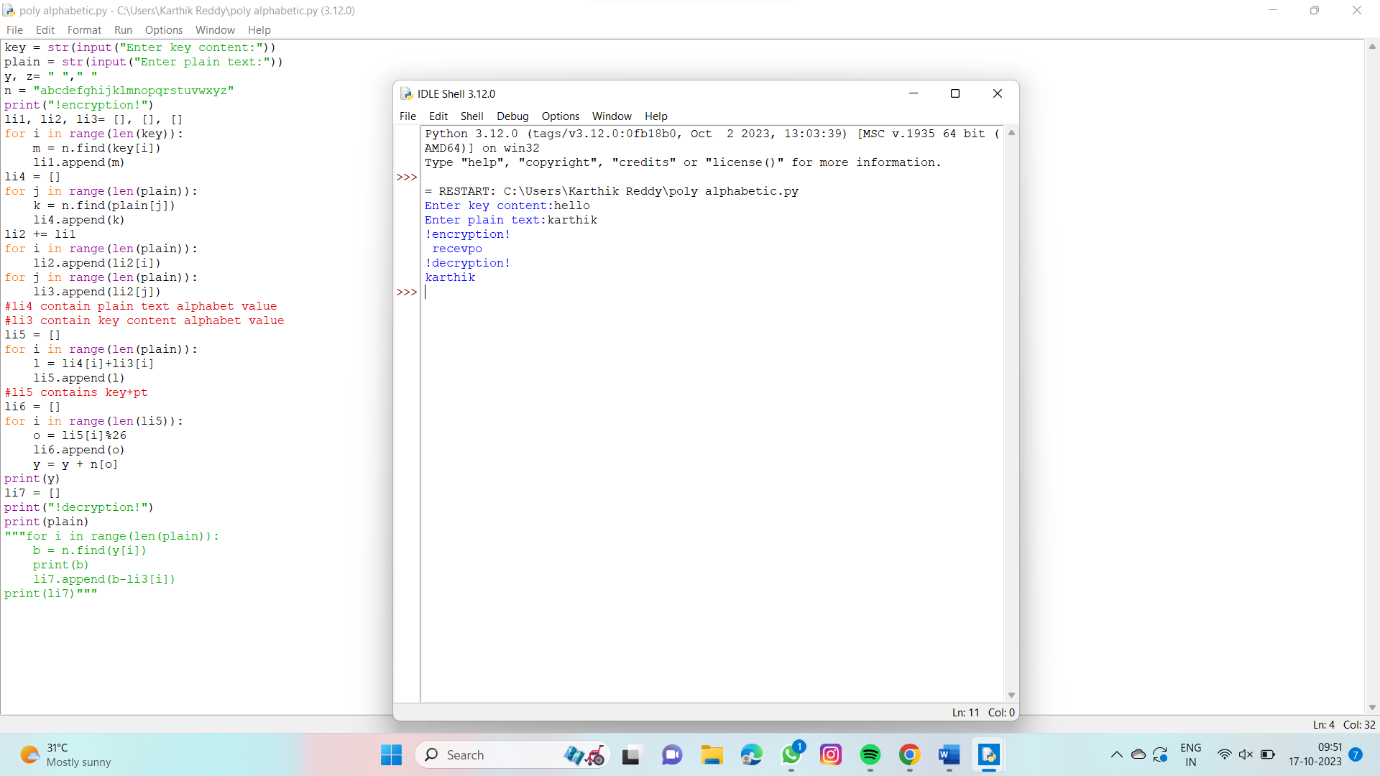
b = n.find(y[i])

print(b)

li7.append(b-li3[i])

print(li7)"""

**OUTPUT :**

****

**QUESTION 5 :**

**Write a C program for  
generalization of the Caesar cipher, known as the affine Caesar cipher, has the  
following form: For each plaintext letter p, substitute the ciphertext letter  
C: C = E([a, b], p) = (ap + b) mod 26 A basic requirement of any encryption algorithm  
is that it be one-to-one. That is, if p q,  
then E(k, p) E(k, q). Otherwise,  
decryption is impossible, because more than  
one plaintext character maps into the same ciphertext  
character. The affine Caesar cipher is  
not one-to-one for all values of a. For example, for a = 2 and b = 3, then E([a,  
b], 0) = E([a, b], 13) = 3.**

**PROGRAM :**

def gcd(a, b):

while b:

a, b = b, a % b

return a

def is\_valid(a, b):

return gcd(a, 26) == 1 and gcd(b, 26) == 1

def encrypt(a, b, plaintext):

ciphertext = ""

for p in plaintext:

if p.isalpha():

# convert plaintext letter to number (A=0, B=1, ..., Z=25)

p\_num = ord(p.upper()) - ord('A')

# apply encryption function

c\_num = (a \* p\_num + b) % 26

# convert ciphertext number back to letter

c = chr(c\_num + ord('A'))

else:

c = p

ciphertext += c

return ciphertext

# Example usage

a = 3

b = 5

plaintext = "karthikreddy"

if is\_valid(a, b):

ciphertext = encrypt(a, b, plaintext)

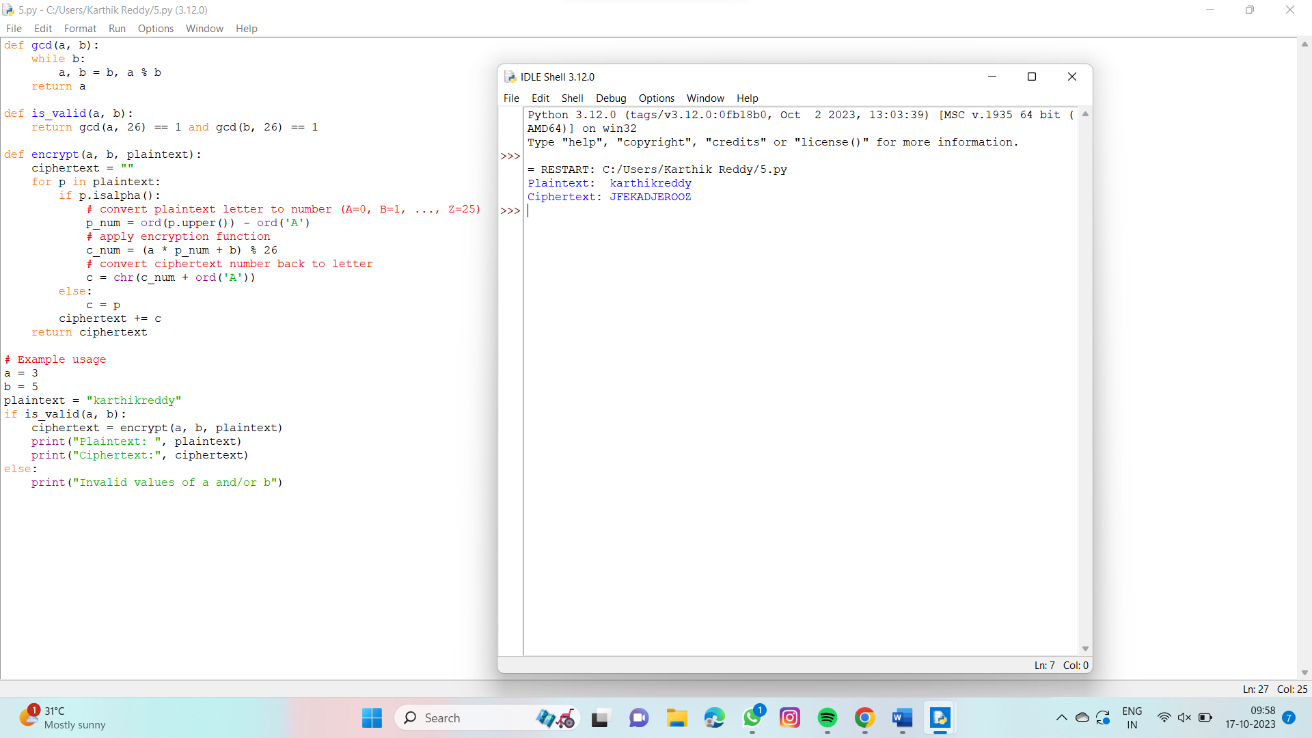
print("Plaintext: ", plaintext)

print("Ciphertext:", ciphertext)

else:

print("Invalid values of a and/or b")

**OUTPUT :**



QUESTION 6 :

**Write a C program for  
ciphertext has been generated with an affine cipher. The most frequent  
letter of the ciphertext is “B,” and the second most frequent letter of the  
ciphertext is “U.”Break this code.**

PROGRAM :

#include <stdio.h>

#include <string.h>

void polyalphabeticCipher(char plaintext[], char key[]) {

int i, j;

int plaintextLength = strlen(plaintext);

int keyLength = strlen(key);

for (i = 0; i < plaintextLength; i++) {

char currentChar = plaintext[i];

char keyChar = key[i % keyLength];

char encryptedChar = 'A' + (currentChar - 'A' + keyChar - 'A') % 26;

printf("%c", encryptedChar);

}

printf("\n");

}

int main() {

char plaintext[100];

char key[100];

printf("Enter the plaintext: ");

scanf("%s", plaintext);

printf("Enter the key: ");

scanf("%s", key);

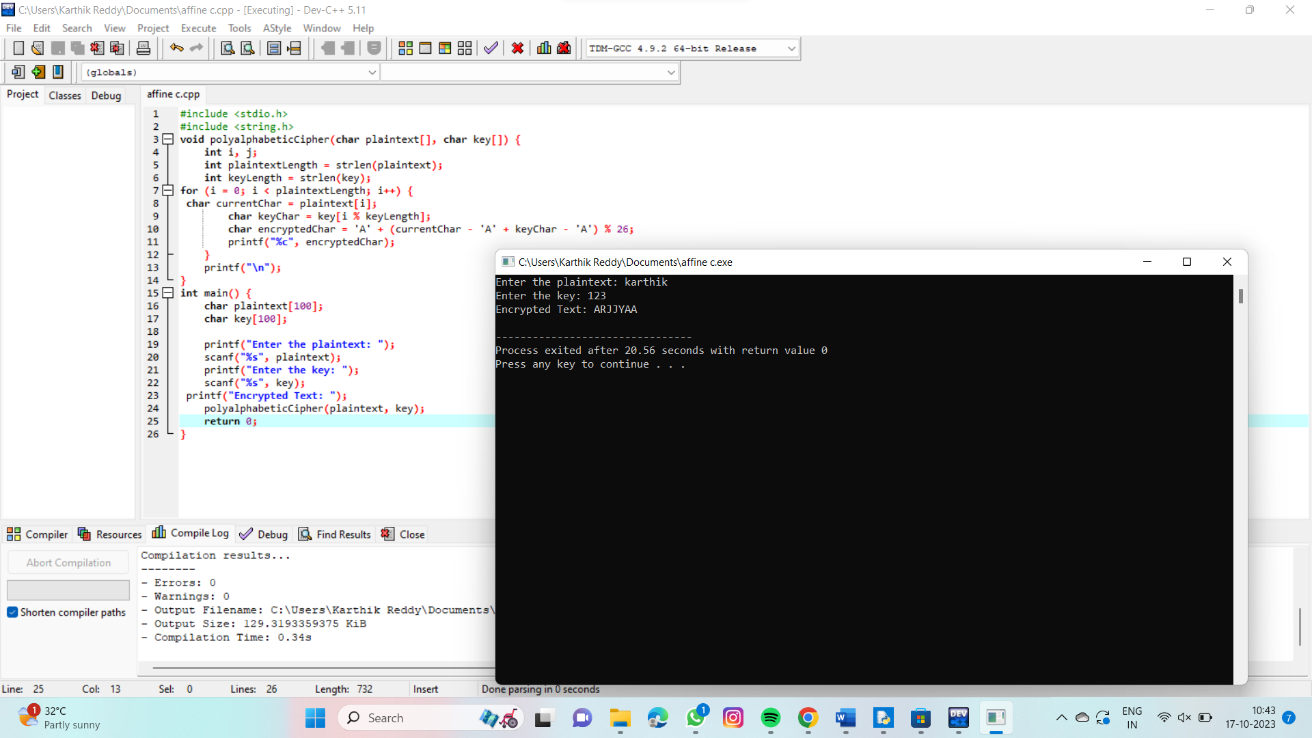
printf("Encrypted Text: ");

polyalphabeticCipher(plaintext, key);

return 0;

}

OUTPUT :

****