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. a. Are there any limitations on the value of b? b. Determine which values of a are not allowed.

Program:

# Affine Caesar Cipher

def encrypt(text, a, b):

result = ""

for char in text.lower():

if char.isalpha():

p = ord(char) - ord('a')

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

result += chr(c + ord('a'))

else:

result += char

return result

def mod\_inverse(a, m):

for i in range(1, m):

if (a \* i) % m == 1:

return i

return None

def decrypt(cipher, a, b):

result = ""

a\_inv = mod\_inverse(a, 26)

if a\_inv is None:

return "Error: 'a' has no modular inverse. Invalid key!"

for char in cipher.lower():

if char.isalpha():

c = ord(char) - ord('a')

p = (a\_inv \* (c - b)) % 26

result += chr(p + ord('a'))

else:

result += char

return result

# Main

text = input("Enter plaintext: ")

a = int(input("Enter value of a: "))

b = int(input("Enter value of b: "))

if \_\_import\_\_('math').gcd(a, 26) != 1:

print("Invalid 'a'! gcd(a,26) must be 1. Not one-to-one.")

else:

encrypted = encrypt(text, a, b)

decrypted = decrypt(encrypted, a, b)

print("\nEncrypted text:", encrypted)

print("Decrypted text:", decrypted)

output:

