**5.** **Write a Python 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.**

**CODE:**

**def egcd(a, b):**

**x,y, u,v = 0,1, 1,0**

**while a != 0:**

**q, r = b//a, b%a**

**m, n = x-u\*q, y-v\*q**

**b,a, x,y, u,v = a,r, u,v, m,n**

**gcd = b**

**return gcd, x, y**

**def modinv(a, m):**

**gcd, x, y = egcd(a, m)**

**if gcd != 1:**

**return None # modular inverse does not exist**

**else:**

**return x % m**

**def affine\_encrypt(text, key):**

**'''**

**C = (a\*P + b) % 26**

**'''**

**return ''.join([ chr((( key[0]\*(ord(t) - ord('A')) + key[1] ) % 26)**

**+ ord('A')) for t in text.upper().replace(' ', '') ])**

**def affine\_decrypt(cipher, key):**

**'''**

**P = (a^-1 \* (C - b)) % 26**

**'''**

**return ''.join([ chr((( modinv(key[0], 26)\*(ord(c) - ord('A') - key[1]))**

**% 26) + ord('A')) for c in cipher ])**

**def main():**

**text = input("enter the text=")**

**key = [17, 20]**

**affine\_encrypted\_text = affine\_encrypt(text, key)**

**print('Encrypted Text: {}'.format( affine\_encrypted\_text ))**

**print('Decrypted Text: {}'.format**

**( affine\_decrypt(affine\_encrypted\_text, key) ))**

**if \_\_name\_\_ == '\_\_main\_\_':**

**main()**

**OUTPUT:**

