**NIS\_LAB\_2\_Assignment**

**CE\_055**

**Aim :-** Aim is to develop the code of Multiplicative inverse, Multiplicative

Cipher and to combine previous both the code implement Affine Cipher.

1. Implement the code to find multiplicative modular inverse.

Code :-

'''

Author : Dhruv B Kakadiya

'''

# Multiplicative Inverse code

# function to find multiplicative Inverse

def find\_mul\_inverse (a, n):

    t1, t2 = 0, 1

    while(a > 0):

        q = n // a

        r = n - (q \* a)

        n, a = a, r

        t = t1 - (q \* t2)

        t1, t2 = t2, t

    gcd, t = n, t1

    return gcd, t

if \_\_name\_\_ == "\_\_main\_\_":

    print("\nEnter the a and mod: \n")

    a, n = map(int, input().split())

    gcd, mul\_inverse = find\_mul\_inverse(a, n)

    if (gcd == 1):

        if (mul\_inverse < 0):

            print(f"Multiplicative inverse of {a} modulo {n} is => {mul\_inverse % n} and {mul\_inverse}\n")

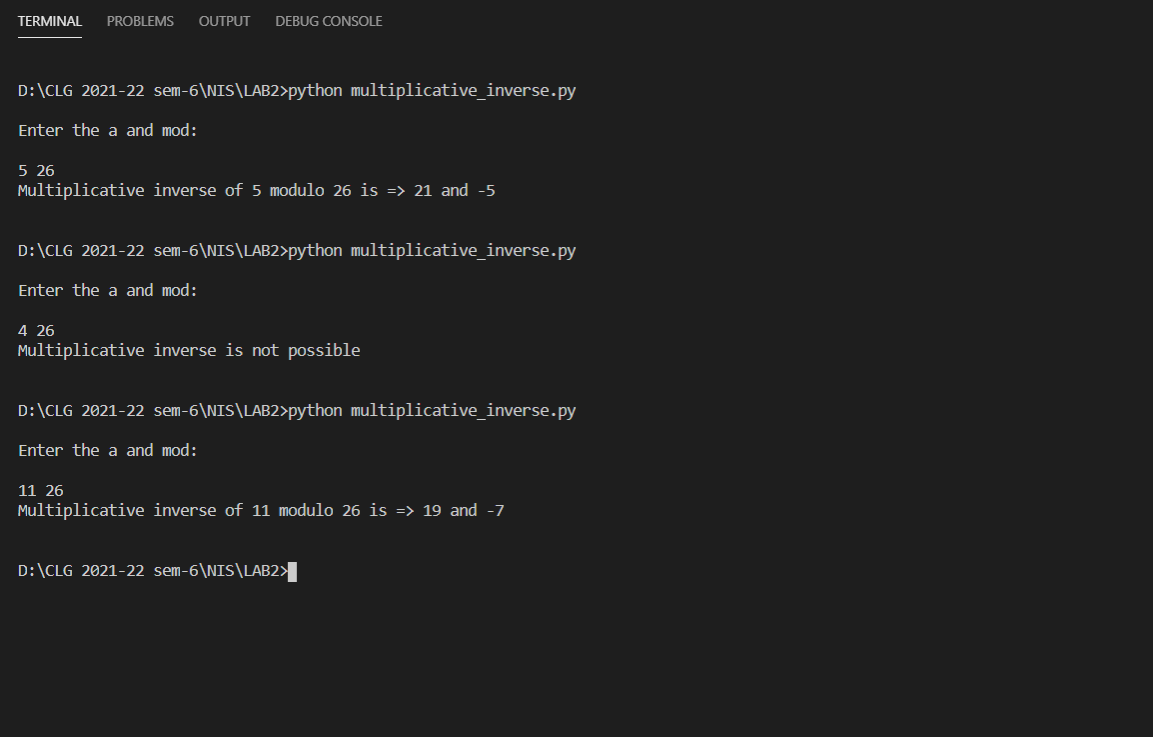
        else:

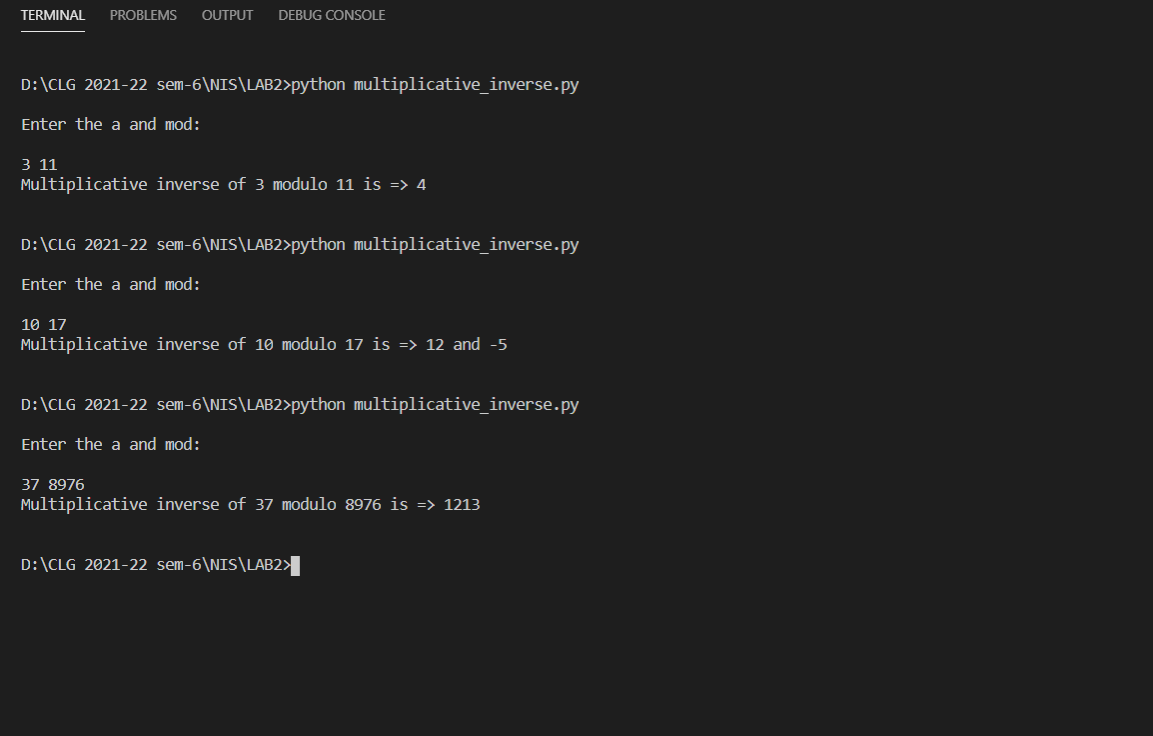
            print(f"Multiplicative inverse of {a} modulo {n} is => {mul\_inverse}\n")

    else:

        print(f"Multiplicative inverse is not possible\n")

Output :-





1. Implement the code for encryption and decryption of Multiplicative cipher.

Code :-

'''

Author : Dhruv B Kakadiya

'''

# Multiplicative Cipher

# static variable

mod = 26

from multiplicative\_inverse import find\_mul\_inverse

# function for encryption of plain\_text

def mul\_cipher\_encryption (plain\_text, key):

    encrypted\_text = ""

    for letter in plain\_text:

        if (letter.isupper()):

            encrypted\_text += chr(((ord(letter) - 65) \* key) % mod + 65)

        elif (letter.islower()):

            encrypted\_text += chr(((ord(letter) - 97) \* key) % mod + 97)

        else:

            encrypted\_text += letter

    return encrypted\_text

# function for decryption

def mul\_cipher\_decryption (encrypted\_text, mul\_inverse):

    decrypted\_text = ""

    for letter in encrypted\_text:

        if (letter.isupper()):

            decrypted\_text += chr(((ord(letter) - 65) \* mul\_inverse) % mod + 65)

        elif (letter.islower()):

            decrypted\_text += chr(((ord(letter) - 97) \* mul\_inverse) % mod + 97)

        else:

            decrypted\_text += letter

    return decrypted\_text

if \_\_name\_\_ == "\_\_main\_\_":

    plain\_text = input("\nEnter the plain Text : ")

    key = int(input("\nEnter the Key : "))

    gcd, mul\_inverse = find\_mul\_inverse(key, mod)

    if (gcd == 1):

        encrypted\_text = mul\_cipher\_encryption(plain\_text, key)

        print(f"\nThe encrypted text of '{plain\_text}' is => '{encrypted\_text}'")

        if (mul\_inverse < 0):

            mul\_inverse = (mul\_inverse) % mod

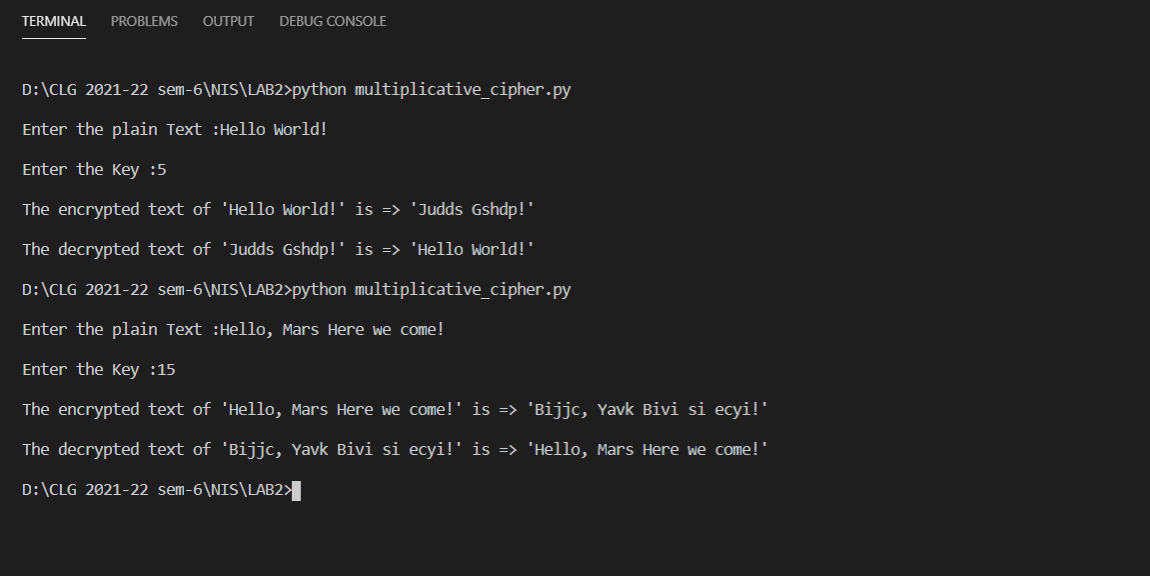
        decrypted\_text = mul\_cipher\_decryption(encrypted\_text, mul\_inverse)

        print(f"\nThe decrypted text of '{encrypted\_text}' is => '{decrypted\_text}'")

    else:

        print("\nNot a Valid key!")

Output :-





1. Implement the code of affine Cipher, It is a combination of additive cipher and multiplicative cipher.

Code :-

'''

Author : Dhruv B Kakadiya

'''

# Multiplicative Cipher

from multiplicative\_inverse import find\_mul\_inverse

# static variable

mod = 26

# function for Encryption

def affine\_cipher\_encryption (plain\_text, key1, key2):

    encrypted\_text = ""

    for letter in plain\_text:

        if (letter.isupper()):

            encrypted\_text += chr(((ord(letter) - 65) \* key1 + key2) % mod + 65)

        elif (letter.islower()):

            encrypted\_text += chr(((ord(letter) - 97) \* key1 + key2) % mod + 97)

        else:

            encrypted\_text += letter

    return encrypted\_text

# function for Decryption

def affine\_cipher\_decryption (encrypted\_text, mul\_inverse, key2):

    decrypted\_text = ""

    for letter in encrypted\_text:

        if (letter.isupper()):

            decrypted\_text += chr(((ord(letter) - 65 - key2) \* mul\_inverse) % mod + 65)

        elif (letter.islower()):

            decrypted\_text += chr(((ord(letter) - 97 - key2) \* mul\_inverse) % mod + 97)

        else:

            decrypted\_text += letter

    return decrypted\_text

if \_\_name\_\_ == "\_\_main\_\_":

    plain\_text = input("\nEnter the plain Text : ")

    key1 = int(input("\nEnter the Key1 : "))

    key2 = int(input("\nEnter the Key2 : "))

    gcd, mul\_inverse = find\_mul\_inverse(key1, mod)

    if (gcd == 1):

        encrypted\_text = affine\_cipher\_encryption(plain\_text, key1, key2)

        print(f"\nThe encrypted text of '{plain\_text}' is => '{encrypted\_text}'")

        if (mul\_inverse < 0):

            mul\_inverse = (mul\_inverse) % mod

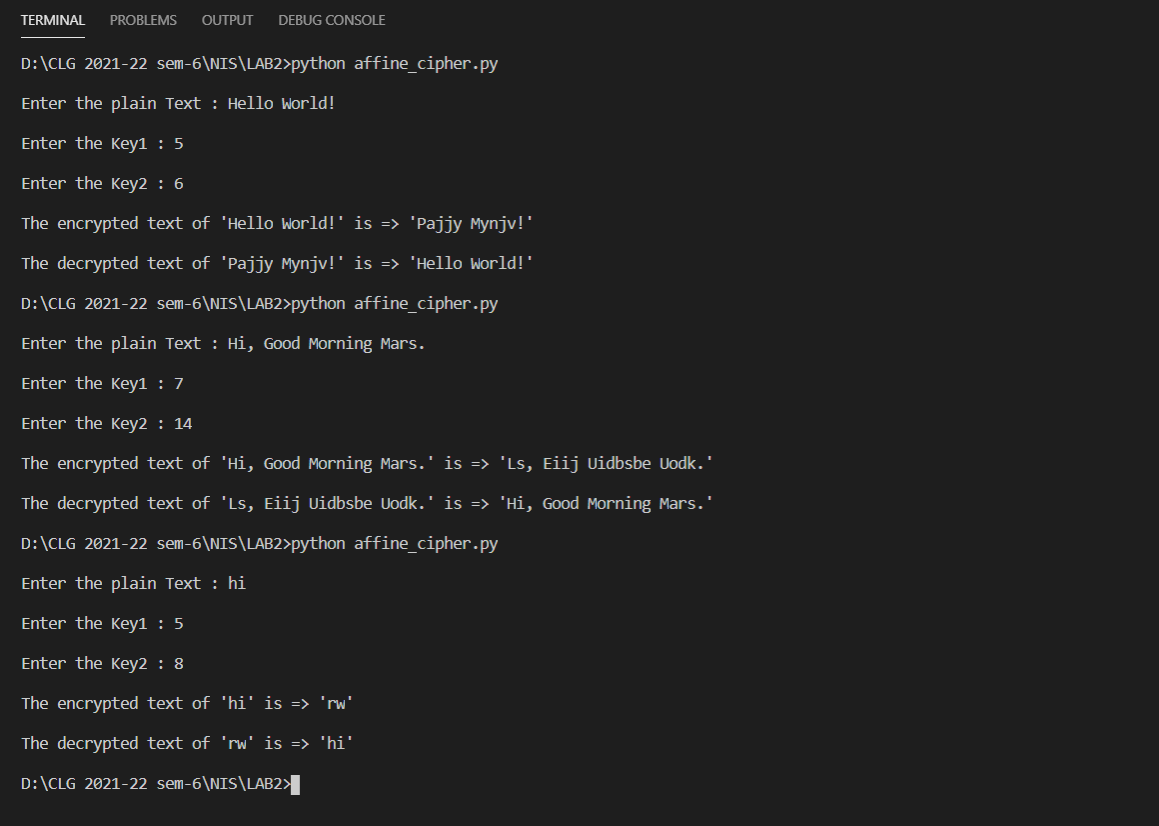
        decrypted\_text = affine\_cipher\_decryption(encrypted\_text, mul\_inverse, key2)

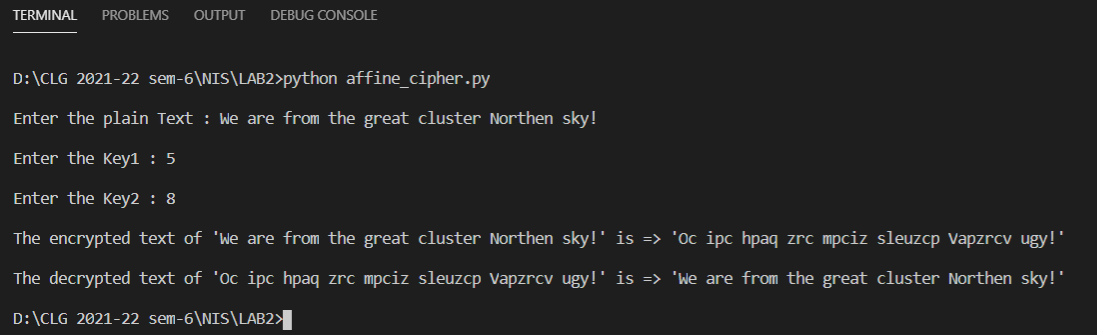
        print(f"\nThe decrypted text of '{encrypted\_text}' is => '{decrypted\_text}'")

    else:

        print("\nNot a Valid key1!")

Output :-





Description :-

1. Multiplicative Modular Inverse:-

* There are two integers ex. A and M if we want to find the modular multiplicative inverse of A with modulo M. assume the inverse is k.
* Formula is a \* k ~= 1 ( mod M ).
* The multiplicative inverse of A modulo M is exists if and only if a and m are co-prime i.e The gcd of A and M is 1.
* If the gcd is not equivalent to 1 then there is no inverse exists.

1. Multiplicative Cipher:-

* This cipher is not strong enough, because in this cipher the formula is

Encryption formula : - ( P \* key ) mod 26.

Decryption formula : - ( C \* key^-1 ) mod 26.

* In the range of 1 to 26 there are only 12 number possible as a key, that’s why it is easy for attacker to get into it and intercept between sender and receiver.
* Here key must have multiplicative inverse under the modulo.

1. Affine Cipher:-

* Affine cipher is type of monoalphabetic substitution cipher, also it is a combination of multiplicative cipher and additive cipher.
* Encryption :

It uses modular arithmetic to transform the integer that each plaintext letter corresponds to into another integer that correspond to a ciphertext letter.

The function for Encryption : -

( (P \* key1) + key2 ) mod 26

* In this formula key1 must be has multiplicative inverse under the mod. And key2 belongs to any 26 alphabet.
* There are 12 \* 26 combination possible for guess pair of key.
* Decryption :

In deciphering the ciphertext, we must perform the inverse functions on the ciphertext to retrieve the plaintext. The first step is to convert each of the ciphertext letters into their integer values.

The function for Decryption: -

( ( C – key2 ) \* key1 ^ -1 ) mod 26

* This is also not strong enough.