B. Tech – CE | Semester: VI | Subject: NIS

Lab 2

1) Multiplicative inverse : This is used in a modular multiplicative inverse cryptography for encryption on the plain text with respect to modulo N, where N is an integer value.

→ If we find a modulo N inverse then we can get a 0 to N-1 value in the answer.but it is not possible all the time that we can get a modulo inverse of all the values for some value modulo inverse is not possible.

→ we find modulo inverse with the help of an extended euclidean algorithm From that algorithm we come to know that if GCD(N, A) = 1 then only Multiplicative inverse is possible. Where N is modular arithmetic and A is the value for which we find a Multiplicative inverse.

→ In this encryption technique we take one key if its multiplicative inverse possible then we do further encryption and if not then we give a exception and terminate a programme.if we get inverse then we encrypt it and for decryption we use a multiplicative inverse in receiver side and get back a plain text.

2) Affine cipher : In an additive cipher there is very little combination and anyone can do brute force (Cryptanalysis) and get our additive key in the additive cipher by just observing some few result.so this encryption method is not useful so we use affine cipher.
→ Affine cipher = Additive cipher + Multiplicative Cipher
→ Encryption = on this side the plain text is first multiplied by a key which has a possible multiplicative inverse on the decryption side and then the result will be added via an additive key.

Cipher Text = ((I * K) + B)

Where k is multiplicative key and B is additive key.

→ Decryption : this side first cipher text is subtracted from the additive key and multiplied by a multiplicative inverse and then will get our plain text back.

Plain text = ((I - B)* K^-1)

Where k is multiplicative key and B is additive key.

Task 1 : Implement a multiplicative inverse cipher and run a different test case on that algorithm.
Source Code :

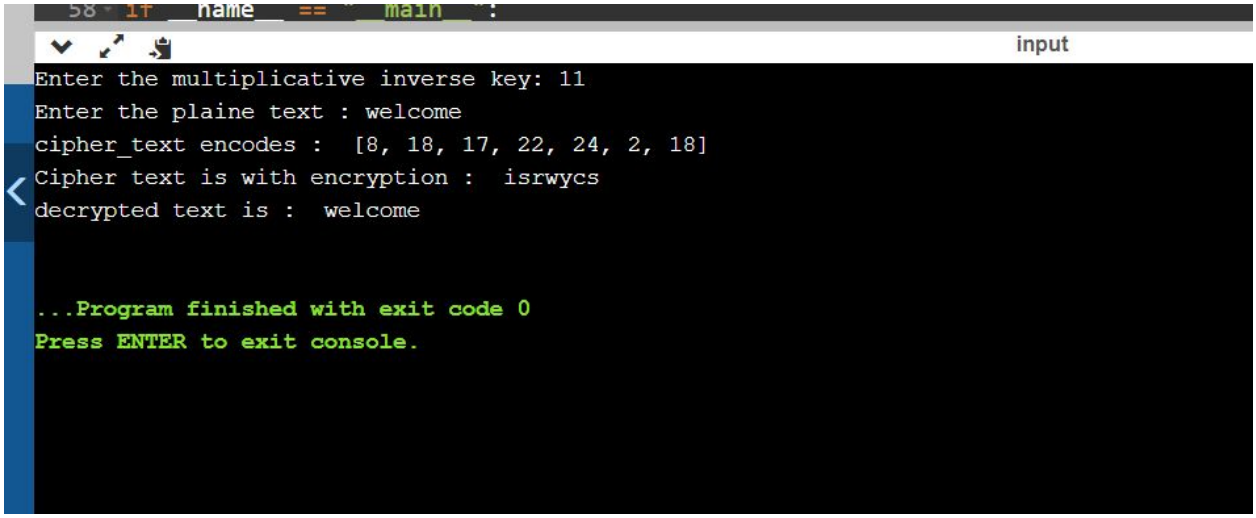
```python
#Multiplicative cipher for encryption and decryption.
from collections import defaultdict


d = defaultdict(lambda: "Not Present")
for i in range(27):
    d[chr(97+i)] = i

def DoMultiplictiveInverse(n , a):
    r1, r2, t1, t2= n, a, 0, 1
    while r2 > 0:
        q = r1//r2
        r = r1-(q*r2)
        r1 = r2
        r2 = r

        t = t1-(q*t2)
        t1 = t2
        t2 = t
    if r1 == 1:
        if t1 < 0:
            return t1 + n
        else:
            return t1

    else:
        return -1

def get_key(val):
    for key, value in d.items():
         if val == value:
             return key
    return "key doesn't exist"

def MultiplicativeEncryption(text , a):
    temp=list()
    for i in range(len(text)):
        temp.append((d[text[i]]*a)%26)
        # temp.append((*answer))
    print("cipher_text encodes : ", temp)
```

```
    cipher_text = ""
    for i in range(len(text)):
        cipher_text+=get_key(temp[i])
    print("Cipher text is with encryption : " ,cipher_text)
    return cipher_text

def MultiplicativeDecryption(cipher_text ,keyInverse):
    cipher_text = list(cipher_text)
    temp=list()
    for i in range(len(cipher_text)):
        temp.append((d[cipher_text[i]]*keyInverse)%26)
    decrypted_text=""
    for i in range(len(cipher_text)):
        decrypted_text+=get_key(temp[i])
    print("decrypted_text is : " , decrypted_text)


if __name__ == "__main__":
    n =  26
    a = int(input("Enter the multiplicative inverse key: "))

    answer =DoMultiplictiveInverse(n, a)
    if answer == -1:
        print("please select key which multiplicative inverse is possible
with respect to 26")
    else:
        text = input("Enter the plain text : ")
        text = list(text)
        cipher_text=MultiplicativeEncryption(text, a)

        MultiplicativeDecryption(cipher_text, answer)
```

Output → Case 1 : When Multiplicative inverse is not possible for the key.

```
Enter the multiplicative inverse key: 13
please select key which multiplicative inverse is possible with respect to 26


...Program finished with exit code 0
Press ENTER to exit console.
```

→ **Case 2 : When multiplicative inverse is possible.**



Task 2 : Implement a Affine cipher and run a different test case on that algorithm.

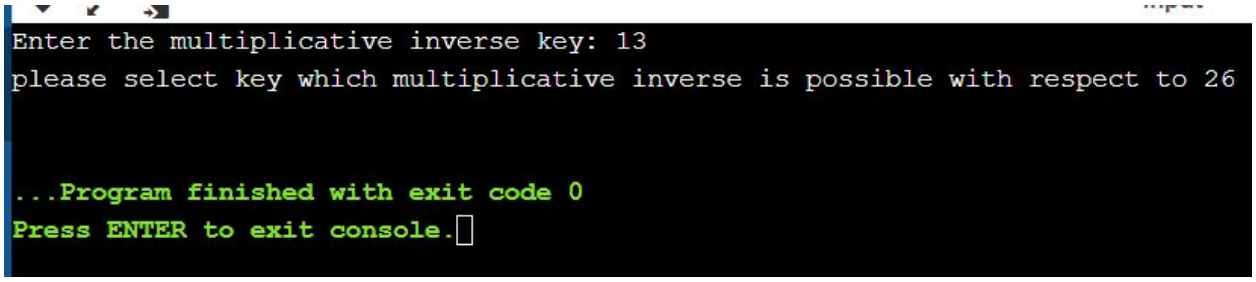
Source Code :

```
#affine cipher
from collections import defaultdict



d = defaultdict(lambda: "Not Present")
for i in range(27):
    d[chr(97+i)] = i

def DoMultiplictiveInverse(n , a):
    r1, r2, t1, t2= n, a, 0, 1
    while r2 > 0:
        q = r1//r2
        r = r1-(q*r2)
        r1 = r2
        r2 = r

        t = t1-(q*t2)
        t1 = t2
        t2 = t
    if r1 == 1:
        if t1 < 0:
            return t1 + n
        else:
```

```
            return t1

    else:
        return -1

def get_key(val):
    for key, value in d.items():
         if val == value:
             return key
    return "key doesn't exist"


def AffineEncryption(plain_text, mult_inverse, add_key):

    return ''.join([ chr((( mult_inverse*(ord(i) - ord('a')) + add_key ) %
26)
                  + ord('a')) for i in plain_text.lower().replace(' ',
'')])

def AffineDecryption(cipher_text, mult_inverse, add_key):
    return ''.join([ chr((( DoMultiplictiveInverse(26 ,
mult_inverse)*(ord(i) - ord('a') - add_key))
                    % 26) + ord('a')) for i in cipher_text ])


if __name__ == "__main__":
    n =  26
    mult_inverse = int(input("Enter the multiplicative inverse key: "))

    answer =DoMultiplictiveInverse(n, mult_inverse)
    if answer == -1:
        print("please select key which multiplicative inverse is possible
with respect to 26")
    else:
        add_key = int(input("Enter the addition key for affine cipher :
"))
        plain_text = input("Enter the plaine text : ")
        cipher_text = AffineEncryption(plain_text, mult_inverse, add_key)
        print("cipher text is-> ", cipher_text)
```

```
        decripted_text = AffineDecryption(cipher_text, mult_inverse,
add_key)
        print("decripted text is->", decripted_text)
```

Output → Case 1 : When Multiplicative inverse is not possible for the key.



→ Case 2 : When multiplicative inverse is possible.