JAGANNATH UNIVERSITY(JnU)

Assignment on

Cryptography And Information Security Lab

Submitted To:

Name: Dr. Mohammed Nasir Uddin

Designation: Professor

Department: CSE

Submitted By:

Name: Md. Aminul Islam Rafi

Student ID: B-180305022

Department: CSE

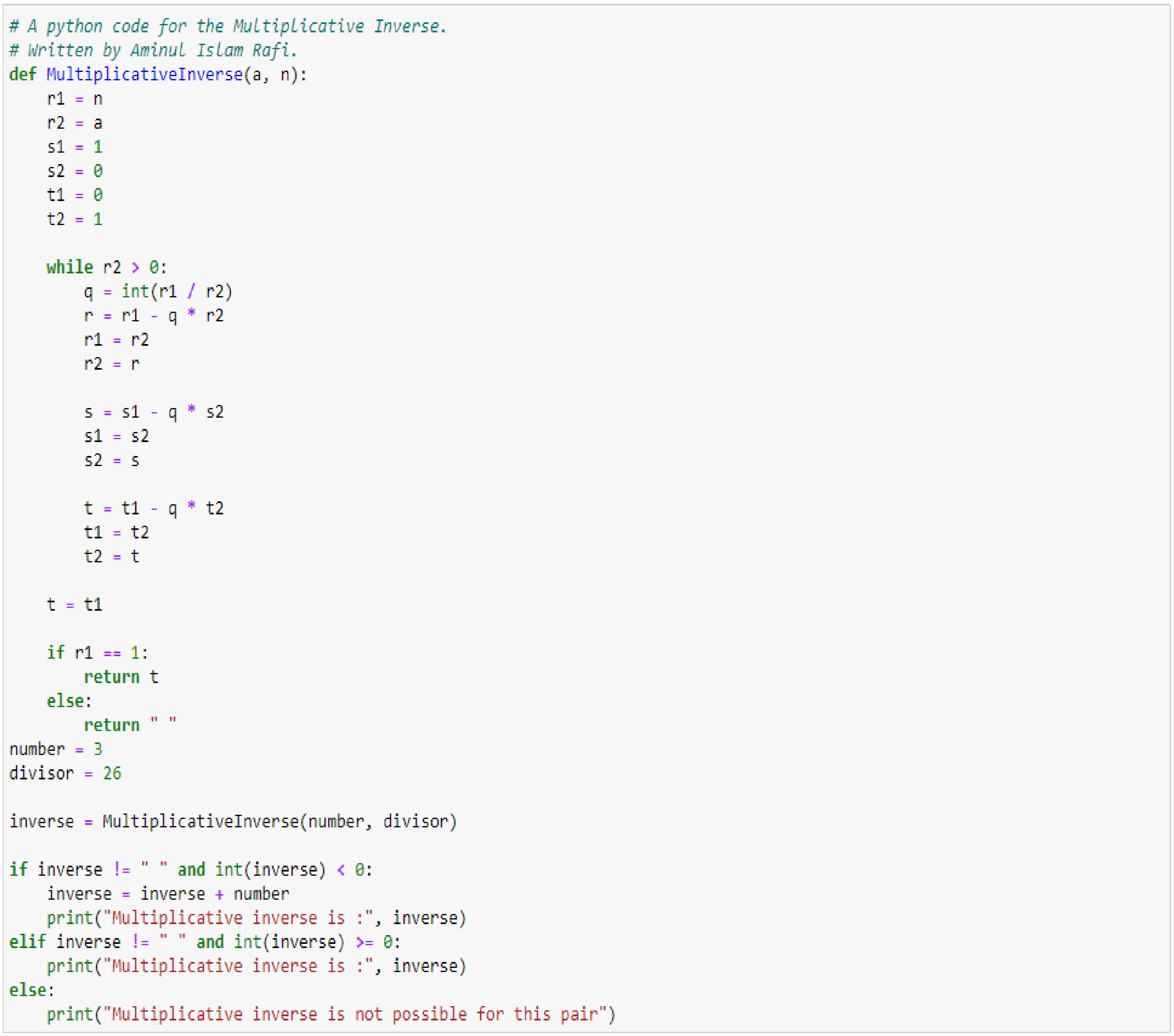
Date of Submission: 20-June-2023

Traditional Cipher

Monoalphabetic Cipher

1.Write a code that find the multiplicative inverse of a given number under a module ’n’?

Code



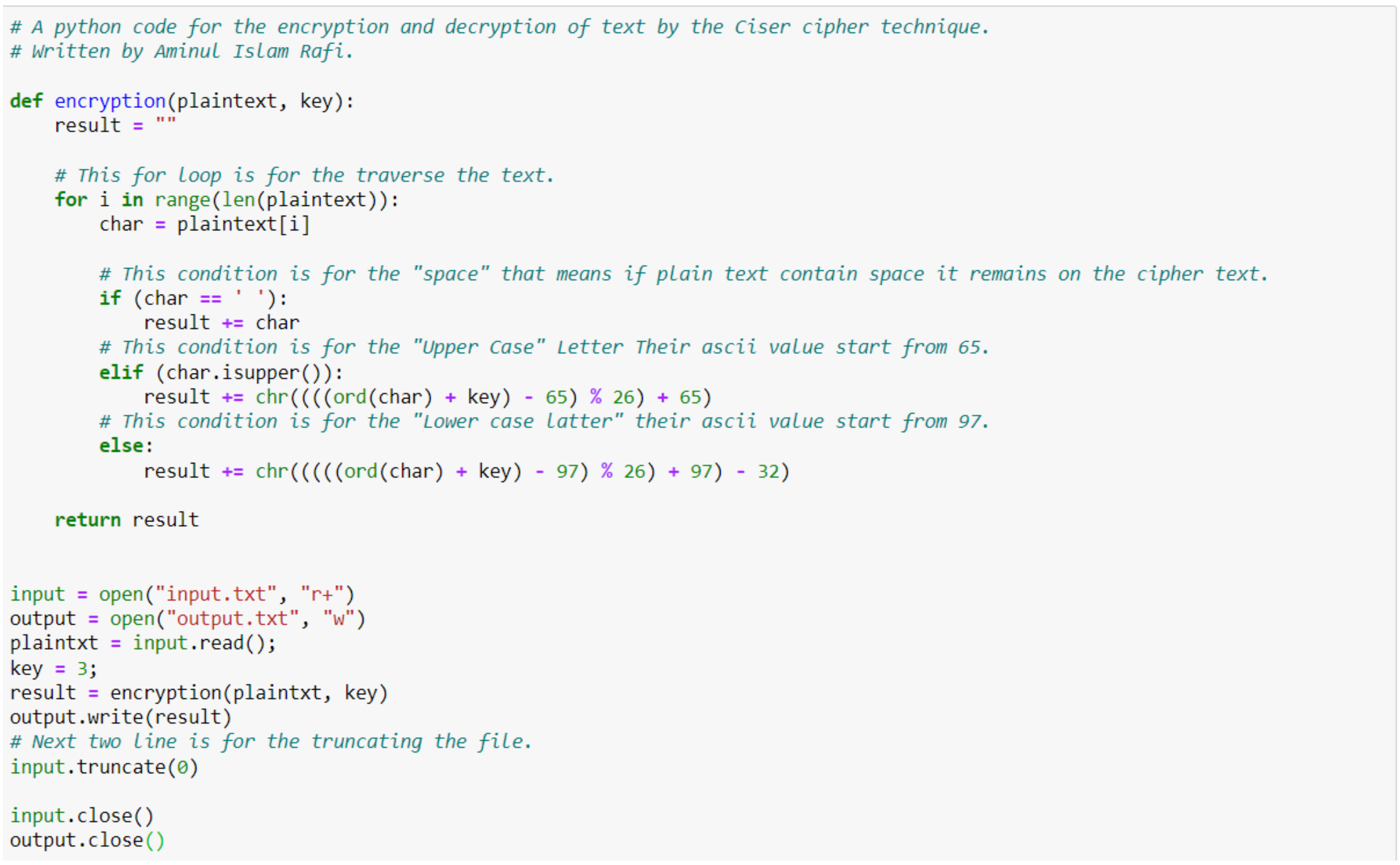
```python
# A python code for the Multiplicative Inverse.
# Written by Aminul Islam Rafi.
def MultiplicativeInverse(a, n):
    r1 = n
    r2 = a
    s1 = 1
    s2 = 0
    t1 = 0
    t2 = 1

    while r2 > 0:
        q = int(r1 / r2)
        r = r1 - q * r2
        r1 = r2
        r2 = r

        s = s1 - q * s2
        s1 = s2
        s2 = s

        t = t1 - q * t2
        t1 = t2
        t2 = t

    t = t1

    if r1 == 1:
        return t
    else:
        return " "
number = 3
divisor = 26

inverse = MultiplicativeInverse(number, divisor)

if inverse != " " and int(inverse) < 0:
    inverse = inverse + number
    print("Multiplicative inverse is :", inverse)
elif inverse != " " and int(inverse) >= 0:
    print("Multiplicative inverse is :", inverse)
else:
    print("Multiplicative inverse is not possible for this pair")
```

Output

```
Multiplicative inverse of 3 under modulo 26 is : 9
```

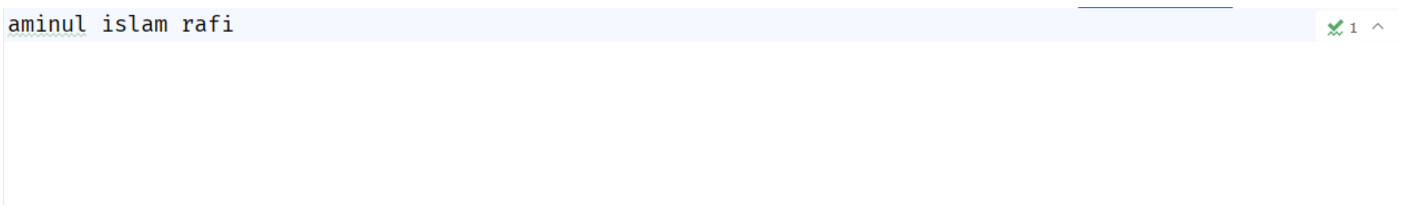
2. Write a python code that encrypt and decrypt the message by using the ceaser cipher take the input from the file and output is saves into another file?

Encryption



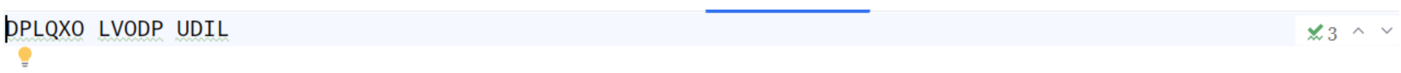
```python
# A python code for the encryption and decryption of text by the Ciser cipher technique.
# Written by Aminul Islam Rafi.

def encryption(plaintext, key):
    result = ""

    # This for loop is for the traverse the text.
    for i in range(len(plaintext)):
        char = plaintext[i]

        # This condition is for the "space" that means if plain text contain space it remains on the cipher text.
        if (char == ' '):
            result += char
        # This condition is for the "Upper Case" Letter Their ascii value start from 65.
        elif (char.isupper()):
            result += chr((((ord(char) + key) - 65) % 26) + 65)
        # This condition is for the "Lower case latter" their ascii value start from 97.
        else:
            result += chr(((((ord(char) + key) - 97) % 26) + 97) - 32)

    return result


input = open("input.txt", "r+")
output = open("output.txt", "w")
plaintxt = input.read();
key = 3;
result = encryption(plaintxt, key)
output.write(result)
# Next two line is for the truncating the file.
input.truncate(0)

input.close()
output.close()
```

Input



```
aminul islam rafi
```

Output



```
DPLQXO LVODP UDIL
```

Decryption

```python
# A python code for the decryption of the text by the Ciser cipher technique.
# Witten by Aminul Islam Rafi.
def decryption(ciphertext, key):
    result = ""

    for i in range(len(ciphertext)):
        char = ciphertext[i]
        # This condition is for the "space" that means if plain text contain space it remains on the cipher text.
        if char == ' ':
            result += char
        # This condition is for the "Upper Case" Letter Their ascii value start from 65.
        elif char.isupper():
            result += chr(((((ord(char) - key) - 65) % 26) + 65)+32)
        # This condition is for the "Lower case latter" their ascii value start from 97.
        else:
            result += chr((((ord(char) - key) - 97) % 26) + 97)
    return result


input = open("output.txt", "r+");
output = open("input.txt", "w");
ciphertext = input.read();
key = 3;
result = decryption(ciphertext, key)
output.write(result)
# Next two line is for the truncating the file.
input.truncate(0)

input.close()
output.close()
```

Input

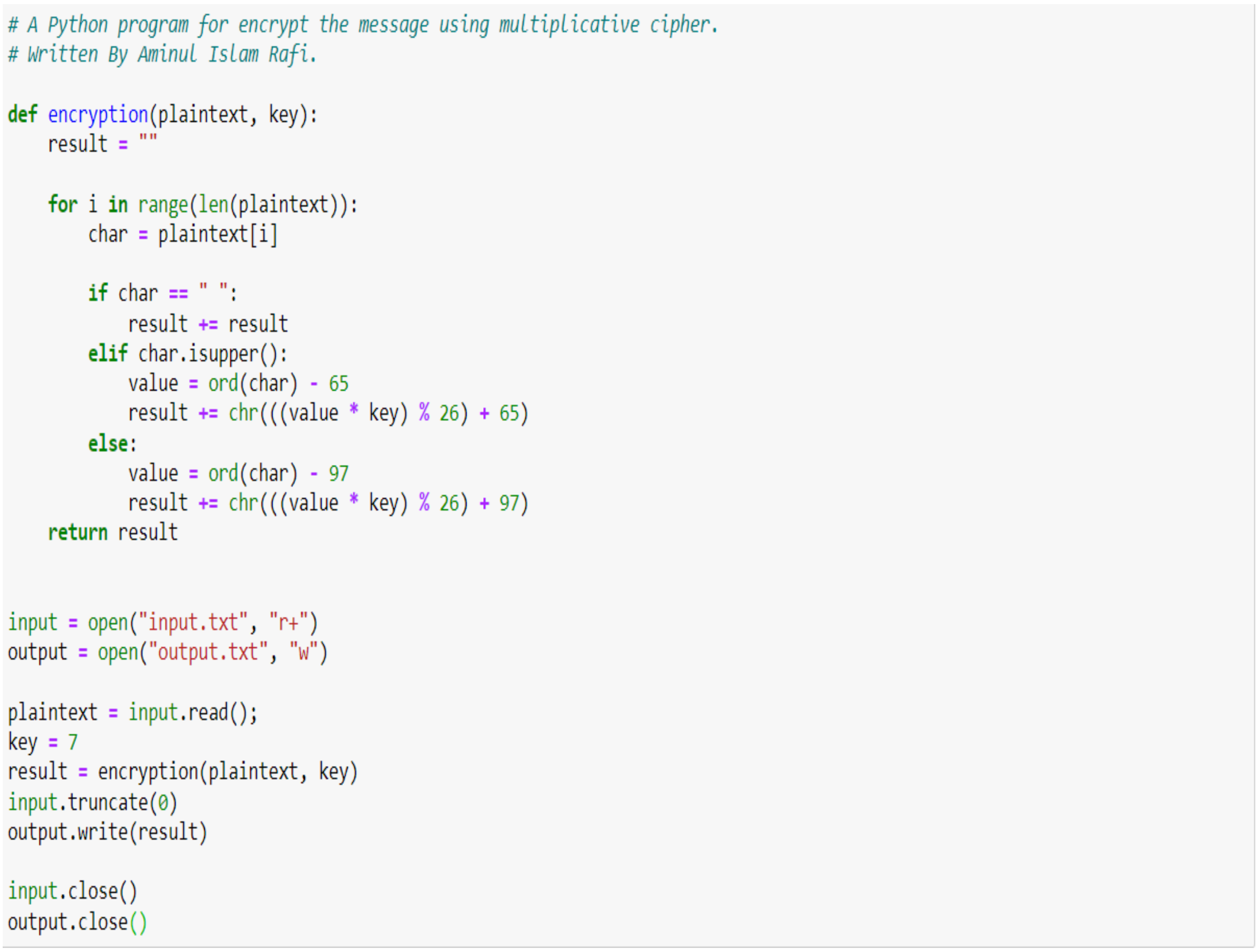
```
DPLQXO LVODP UDIL
```

Output

```
aminul islam rafi
```

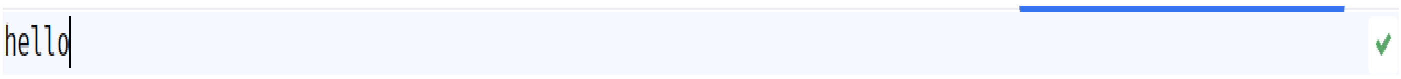
3.Write a python code for finding encryption and decryption of a give text by using multiplicative cipher?

Encryption



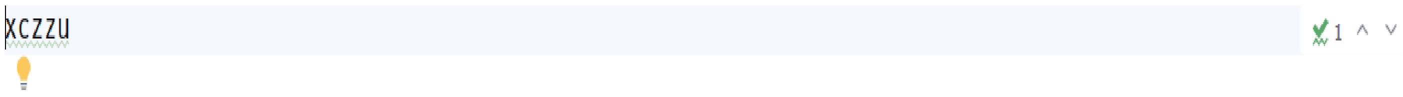
```python
# A Python program for encrypt the message using multiplicative cipher.
# Written By Aminul Islam Rafi.

def encryption(plaintext, key):
    result = ""

    for i in range(len(plaintext)):
        char = plaintext[i]

        if char == " ":
            result += result
        elif char.isupper():
            value = ord(char) - 65
            result += chr(((value * key) % 26) + 65)
        else:
            value = ord(char) - 97
            result += chr(((value * key) % 26) + 97)
    return result


input = open("input.txt", "r+")
output = open("output.txt", "w")

plaintext = input.read();
key = 7
result = encryption(plaintext, key)
input.truncate(0)
output.write(result)

input.close()
output.close()
```

Input



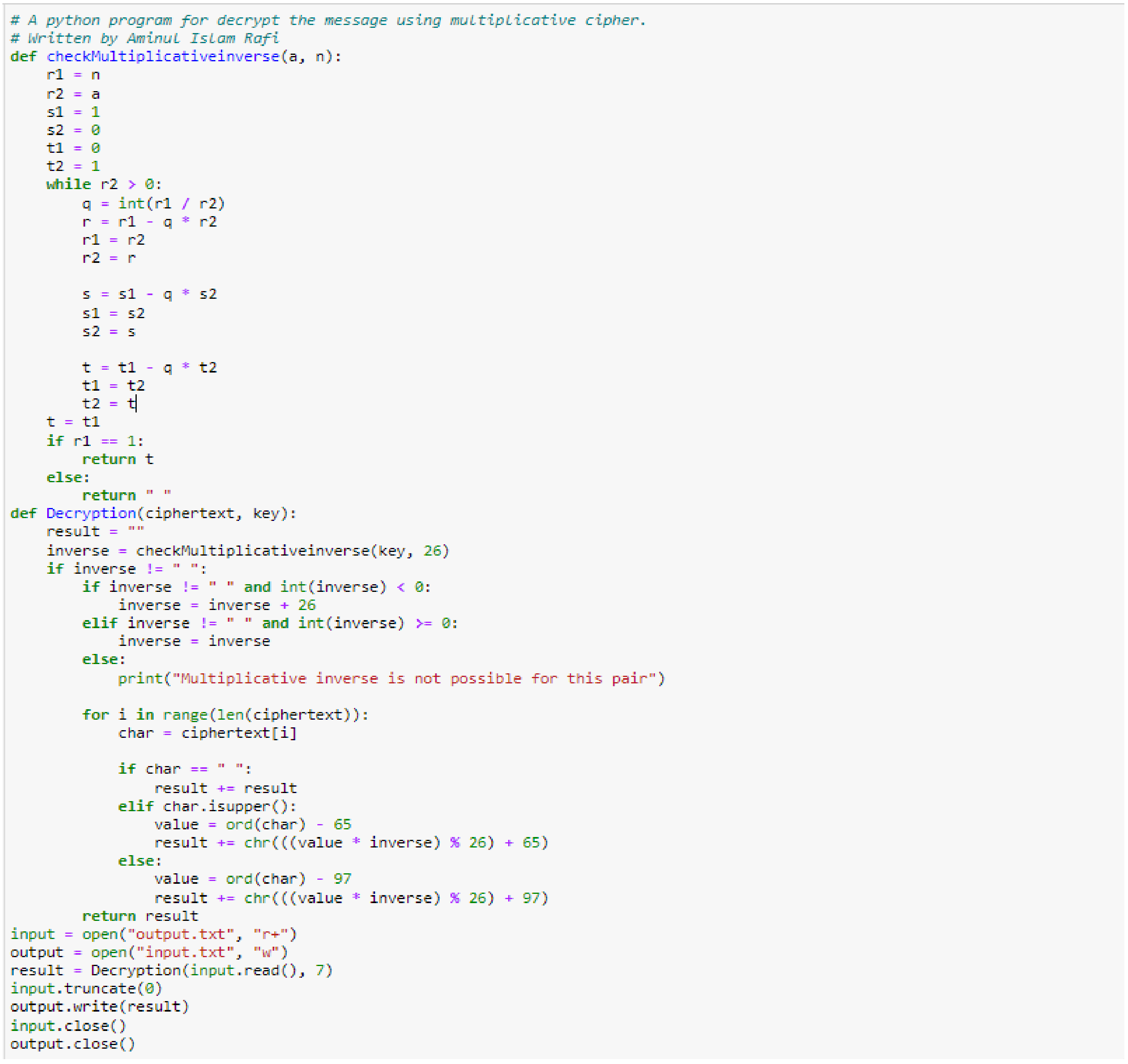
```
hello
```

Output



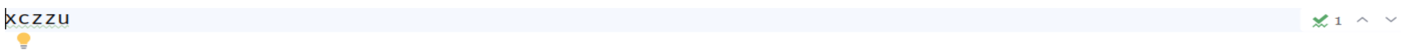
```
xczzu
```

Decryption



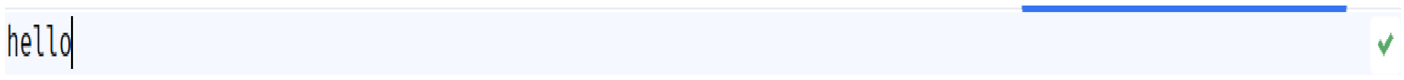
```
# A python program for decrypt the message using multiplicative cipher.
# Written by Aminul Islam Rafi
def checkMultiplicativeinverse(a, n):
    r1 = n
    r2 = a
    s1 = 1
    s2 = 0
    t1 = 0
    t2 = 1
    while r2 > 0:
        q = int(r1 / r2)
        r = r1 - q * r2
        r1 = r2
        r2 = r

        s = s1 - q * s2
        s1 = s2
        s2 = s

        t = t1 - q * t2
        t1 = t2
        t2 = t
    t = t1
    if r1 == 1:
        return t
    else:
        return " "
def Decryption(ciphertext, key):
    result = ""
    inverse = checkMultiplicativeinverse(key, 26)
    if inverse != " ":
        if inverse != " " and int(inverse) < 0:
            inverse = inverse + 26
        elif inverse != " " and int(inverse) >= 0:
            inverse = inverse
        else:
            print("Multiplicative inverse is not possible for this pair")

        for i in range(len(ciphertext)):
            char = ciphertext[i]

            if char == " ":
                result += result
            elif char.isupper():
                value = ord(char) - 65
                result += chr(((value * inverse) % 26) + 65)
            else:
                value = ord(char) - 97
                result += chr(((value * inverse) % 26) + 97)
        return result
input = open("output.txt", "r+")
output = open("input.txt", "w")
result = Decryption(input.read(), 7)
input.truncate(0)
output.write(result)
input.close()
output.close()
```

Input



```
xczzu
```

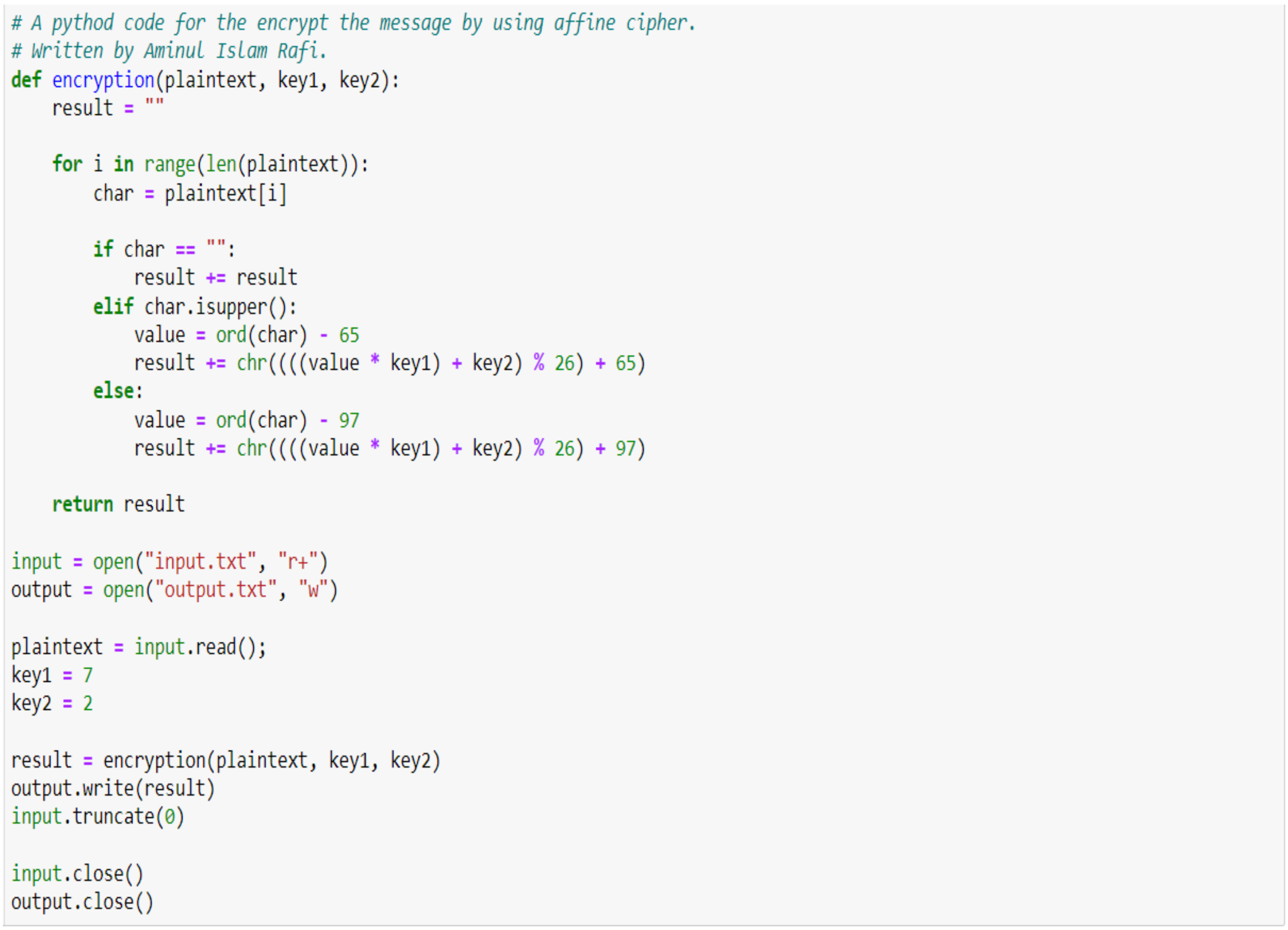
Output



```
hello
```

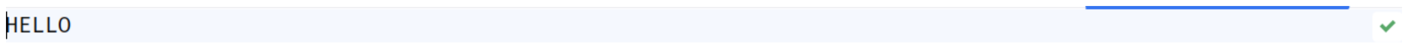
4.Write a python code for encryption and decryption of given text using affine cipher take file input and save the result into another file?

Encryption



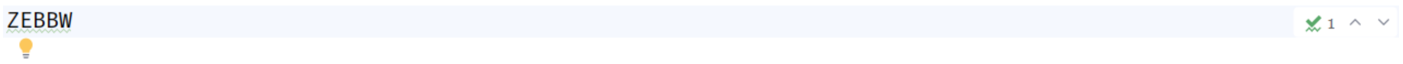
```python
# A pythod code for the encrypt the message by using affine cipher.
# Written by Aminul Islam Rafi.
def encryption(plaintext, key1, key2):
    result = ""

    for i in range(len(plaintext)):
        char = plaintext[i]

        if char == "":
            result += result
        elif char.isupper():
            value = ord(char) - 65
            result += chr((((value * key1) + key2) % 26) + 65)
        else:
            value = ord(char) - 97
            result += chr((((value * key1) + key2) % 26) + 97)

    return result

input = open("input.txt", "r+")
output = open("output.txt", "w")

plaintext = input.read();
key1 = 7
key2 = 2

result = encryption(plaintext, key1, key2)
output.write(result)
input.truncate(0)

input.close()
output.close()
```

Input



```
HELLO
```

Output



```
ZEBBW
```

Decryption

```python
# A pythod code for the decrypt the message by using affine cipher.
# Written by Aminul Islam Rafi.
def MultiplicativeInverse(a, n):
    r1 = n
    r2 = a
    s1 = 1
    s2 = 0
    t1 = 0
    t2 = 1
    while r2 > 0:
        q = int(r1 / r2)
        r = r1 - q * r2
        r1 = r2
        r2 = r

        s = s1 - q * s2
        s1 = s2
        s2 = s

        t = t1 - q * t2
        t1 = t2
        t2 = t
    t = t1
    if r1 == 1:
        return t + n
    else:
        return " "
def encryption(plaintext, key1, key2):
    result = ""
    for i in range(len(plaintext)):
        char = plaintext[i]
        if char == "":
            result += result
        elif char.isupper():
            value = ((ord(char) - 65) - key2)
            result += chr(((value * key1) % 26) + 65)
        else:
            value = ((ord(char) - 97) - key2)
            result += chr(((value * key1) % 26) + 97)

    return result
input = open("output.txt", "r+")
output = open("input.txt", "w")

plaintext = input.read();
key1 = 7
key2 = 2
value = MultiplicativeInverse(key1, 26)
if value != " ":
    result = encryption(plaintext, value, key2)
    output.write(result)
    input.truncate(0)
else:
    print("Decryption is not possible by this key")
input.close()
output.close()
```

Input

```
ZEBBW
```

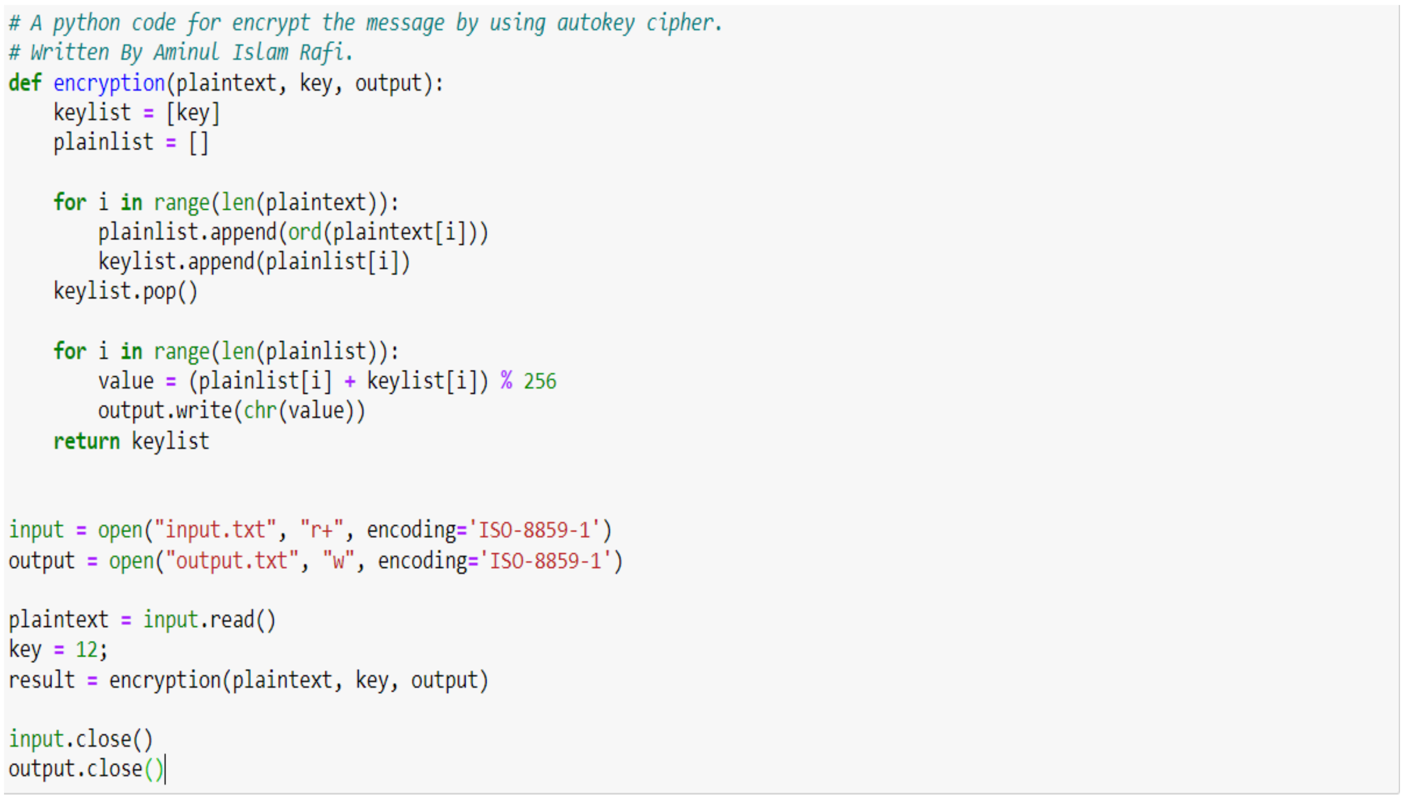
Output

```
HELLO
```

Polyalphabetic Cipher

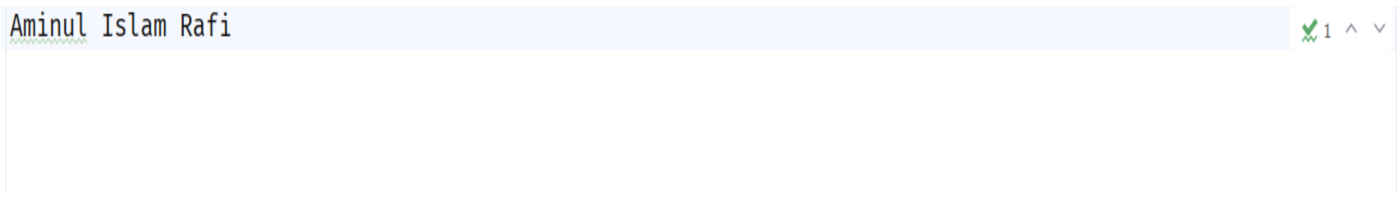
5.Write a python code for encryption and decryption of a given text by using autokey cipher take the file input and save the result into another file?

Encryption



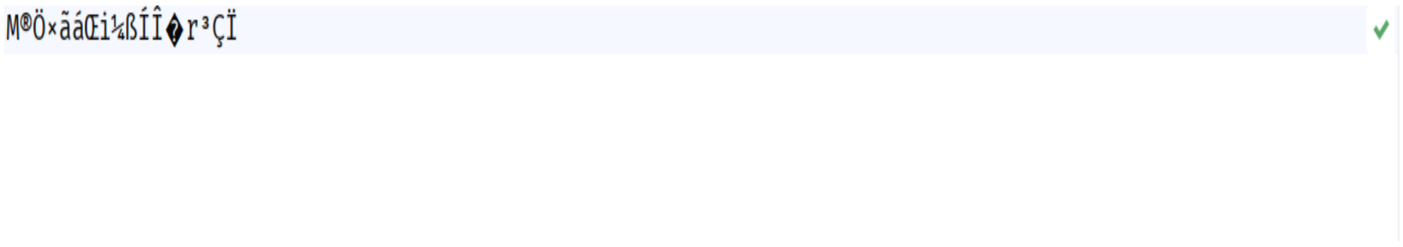
```python
# A python code for encrypt the message by using autokey cipher.
# Written By Aminul Islam Rafi.
def encryption(plaintext, key, output):
    keylist = [key]
    plainlist = []

    for i in range(len(plaintext)):
        plainlist.append(ord(plaintext[i]))
        keylist.append(plainlist[i])
    keylist.pop()

    for i in range(len(plainlist)):
        value = (plainlist[i] + keylist[i]) % 256
        output.write(chr(value))
    return keylist


input = open("input.txt", "r+", encoding='ISO-8859-1')
output = open("output.txt", "w", encoding='ISO-8859-1')

plaintext = input.read()
key = 12;
result = encryption(plaintext, key, output)

input.close()
output.close()
```

Input



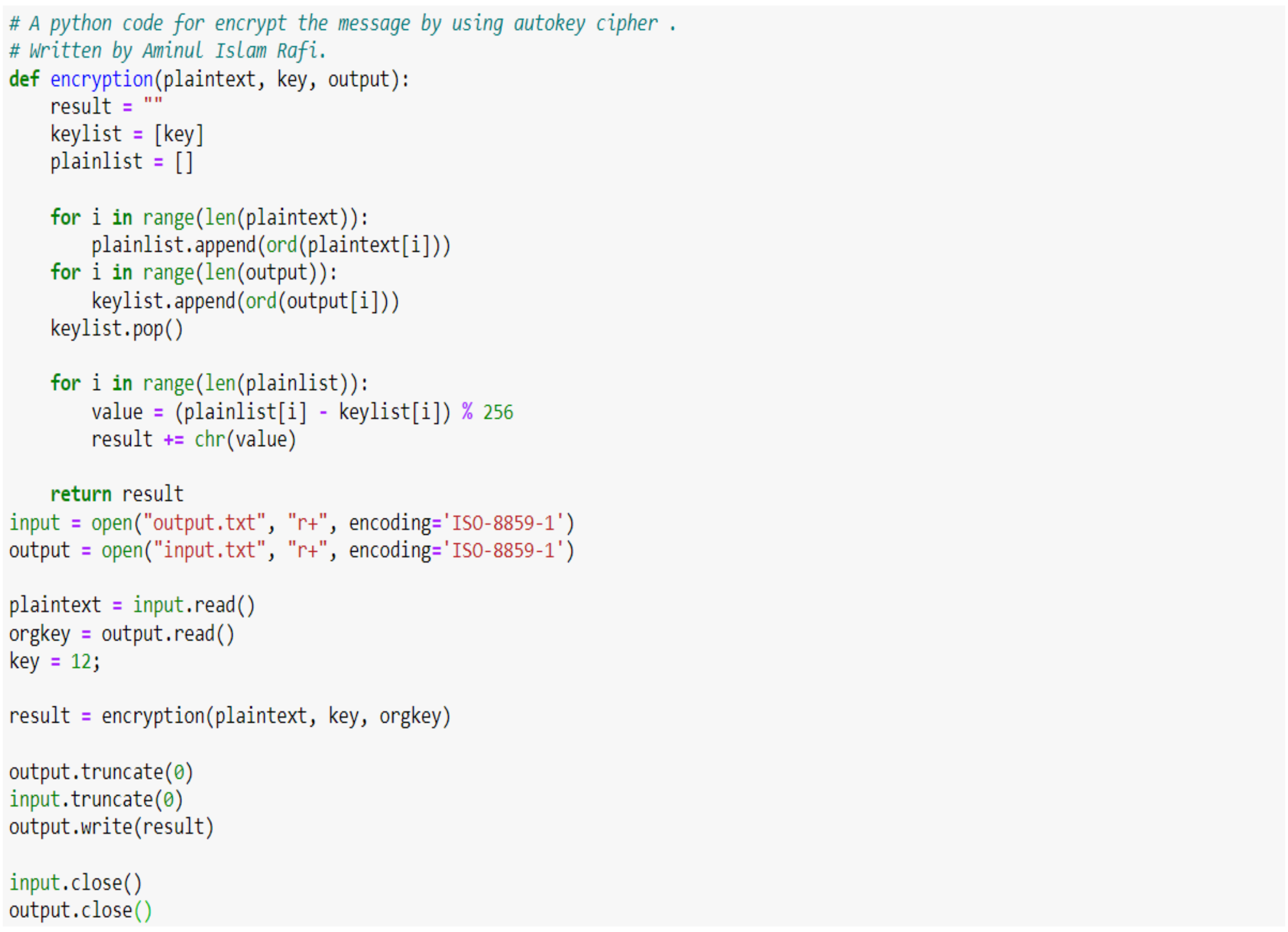
```
Aminul Islam Rafi
```

Output



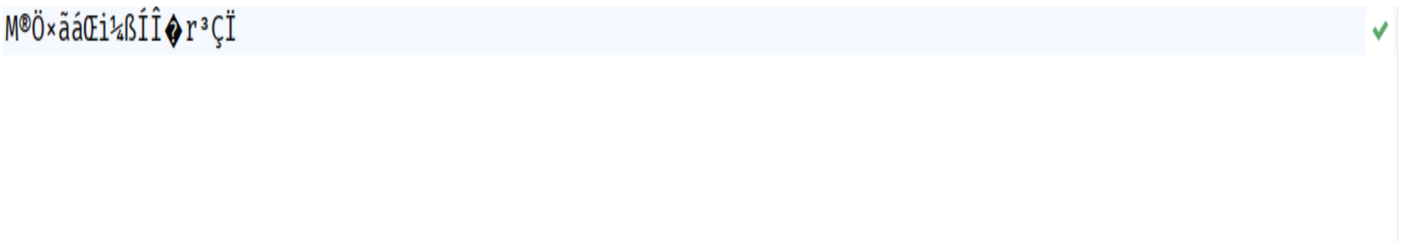
```
M®Ö×ãáŒi¼ßÍÎ�r³ÇÏ
```

Decryption



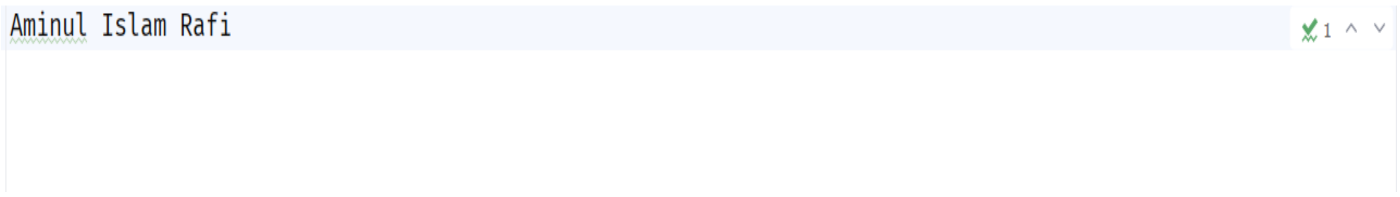
```python
# A python code for encrypt the message by using autokey cipher .
# Written by Aminul Islam Rafi.
def encryption(plaintext, key, output):
    result = ""
    keylist = [key]
    plainlist = []

    for i in range(len(plaintext)):
        plainlist.append(ord(plaintext[i]))
    for i in range(len(output)):
        keylist.append(ord(output[i]))
    keylist.pop()

    for i in range(len(plainlist)):
        value = (plainlist[i] - keylist[i]) % 256
        result += chr(value)

    return result
input = open("output.txt", "r+", encoding='ISO-8859-1')
output = open("input.txt", "r+", encoding='ISO-8859-1')

plaintext = input.read()
orgkey = output.read()
key = 12;

result = encryption(plaintext, key, orgkey)

output.truncate(0)
input.truncate(0)
output.write(result)

input.close()
output.close()
```

Input



```
M®Ö×ãáŒi¼ßÍÎ�r³ÇÏ
```

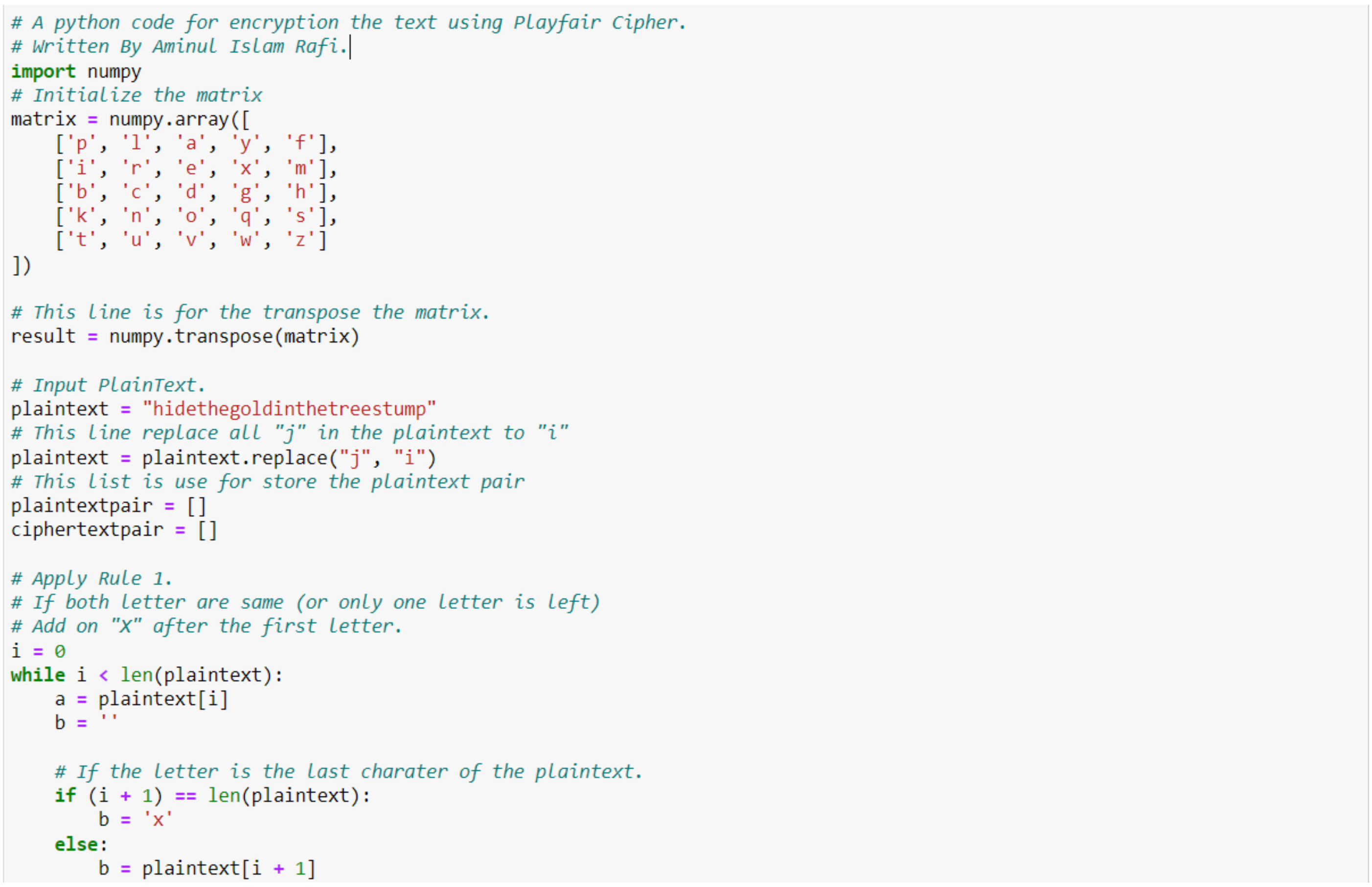
Output



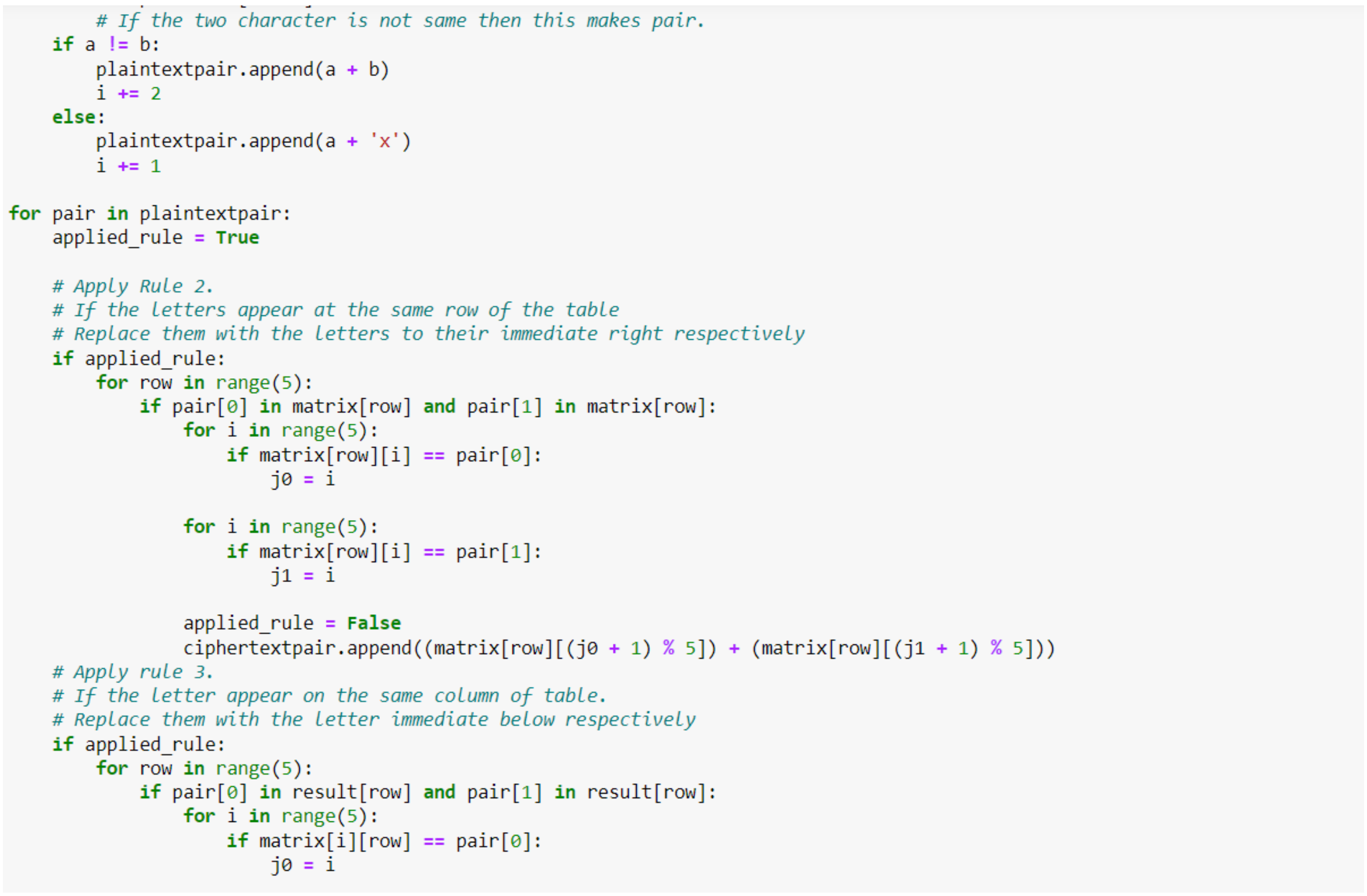
```
Aminul Islam Rafi
```

6.Write a python code for finding the encryption and decryption of a given text by using playfair cipher take the input from the file and show the output in console?

Encryption



```python
# A python code for encryption the text using Playfair Cipher.
# Written By Aminul Islam Rafi.
import numpy
# Initialize the matrix
matrix = numpy.array([
    ['p', 'l', 'a', 'y', 'f'],
    ['i', 'r', 'e', 'x', 'm'],
    ['b', 'c', 'd', 'g', 'h'],
    ['k', 'n', 'o', 'q', 's'],
    ['t', 'u', 'v', 'w', 'z']
])

# This line is for the transpose the matrix.
result = numpy.transpose(matrix)

# Input PlainText.
plaintext = "hidethegoldinthetreestump"
# This line replace all "j" in the plaintext to "i"
plaintext = plaintext.replace("j", "i")
# This list is use for store the plaintext pair
plaintextpair = []
ciphertextpair = []

# Apply Rule 1.
# If both letter are same (or only one letter is left)
# Add on "X" after the first letter.
i = 0
while i < len(plaintext):
    a = plaintext[i]
    b = ''

    # If the letter is the last charater of the plaintext.
    if (i + 1) == len(plaintext):
        b = 'x'
    else:
        b = plaintext[i + 1]
```



```python
        # If the two character is not same then this makes pair.
    if a != b:
        plaintextpair.append(a + b)
        i += 2
    else:
        plaintextpair.append(a + 'x')
        i += 1

for pair in plaintextpair:
    applied_rule = True

    # Apply Rule 2.
    # If the letters appear at the same row of the table
    # Replace them with the letters to their immediate right respectively
    if applied_rule:
        for row in range(5):
            if pair[0] in matrix[row] and pair[1] in matrix[row]:
                for i in range(5):
                    if matrix[row][i] == pair[0]:
                        j0 = i

                for i in range(5):
                    if matrix[row][i] == pair[1]:
                        j1 = i

                applied_rule = False
                ciphertextpair.append((matrix[row][(j0 + 1) % 5]) + (matrix[row][(j1 + 1) % 5]))
    # Apply rule 3.
    # If the letter appear on the same column of table.
    # Replace them with the letter immediate below respectively
    if applied_rule:
        for row in range(5):
            if pair[0] in result[row] and pair[1] in result[row]:
                for i in range(5):
                    if matrix[i][row] == pair[0]:
                        j0 = i
                    for i in range(5):
                        if matrix[i][row] == pair[1]:
                            j1 = i

                    applied_rule = False
                ciphertextpair.append((matrix[(j0 + 1) % 5][row]) + (matrix[(j1 + 1) % 5][row]))
    # Apply rule 4
    # If the letters are not in the same row or same column
    # replace them with the letters on the same row respectively but at the
    # other pair of the corners of the rectangle define by the orginal pair.
    if applied_rule:
        for row in range(5):
            for col in range(5):
                if matrix[row][col] == pair[0]:
                    x0 = row
                    y0 = col
            for row1 in range(5):
                for col1 in range(5):
                    if matrix[row1][col1] == pair[1]:
                        x1 = row1
                        y1 = col1
        ciphertextpair.append((matrix[x0][y1]) + (matrix[x1][y0]))

print(plaintextpair)
print(ciphertextpair)
```

Input & Output

Top row corresponds to the input pair and the second row corresponding to the ciphertext pair.

```
['hi', 'de', 'th', 'eg', 'ol', 'di', 'nt', 'he', 'tr', 'ex', 'es', 'tu', 'mp']
['bm', 'od', 'zb', 'xd', 'na', 'be', 'ku', 'dm', 'ui', 'xm', 'mo', 'uv', 'if']
```

Decryption

```
# A python code for decrypting the ciphertext into the plaintext.
# Written by Aminul Islam Rafi.
import numpy
# Initialize the matrix
matrix = numpy.array([
    ['p', 'l', 'a', 'y', 'f'],
    ['i', 'r', 'e', 'x', 'm'],
    ['b', 'c', 'd', 'g', 'h'],
    ['k', 'n', 'o', 'q', 's'],
    ['t', 'u', 'v', 'w', 'z']
])

# This line is for the transpose the matrix.
result = numpy.transpose(matrix)

# Input PlainText.
plaintext = "bmodzbxdnabekudmuixmmouvif"
# This line replace all "j" in the plaintext to "i"
plaintext = plaintext.replace("j", "i")
# This list is use for store the plaintext pair
plaintextpair = []
ciphertextpair = []

# Apply Rule 1.
# make the pair of two letter.
i = 0
while i < len(plaintext):
    a = plaintext[i]
    b = plaintext[i + 1]

    plaintextpair.append(a + b)
    i += 2

for pair in plaintextpair:
    applied_rule = True
```

```python
    # Apply Rule 2.
    # If the letters appear at the same row of the table
    # Replace them with the letters to their immediate left respectively
    if applied_rule:
        for row in range(5):
            if pair[0] in matrix[row] and pair[1] in matrix[row]:
                for i in range(5):
                    if matrix[row][i] == pair[0]:
                        j0 = i

                for i in range(5):
                    if matrix[row][i] == pair[1]:
                        j1 = i

                applied_rule = False
                ciphertextpair.append((matrix[row][(j0 - 1) % 5]) + (matrix[row][(j1 - 1) % 5]))
    # Apply rule 3.
    # If the letter appear on the same column of table.
    # Replace them with the letter immediate upper respectively
    if applied_rule:
        for row in range(5):
            if pair[0] in result[row] and pair[1] in result[row]:
                for i in range(5):
                    if matrix[i][row] == pair[0]:
                        j0 = i

                    for i in range(5):
                        if matrix[i][row] == pair[1]:
                            j1 = i

                    applied_rule = False
                ciphertextpair.append((matrix[(j0 - 1) % 5][row]) + (matrix[(j1 - 1) % 5][row]))
    # Apply rule 4.
    # If the letters are not in the same row or same column
    # replace them with the letters on the same row respectively but at the
    # other pair of the corners of the rectangle define by the orginal pair.
```

```python
    if applied_rule:
        for row in range(5):
            for col in range(5):
                if matrix[row][col] == pair[0]:
                    x0 = row
                    y0 = col
            for row1 in range(5):
                for col1 in range(5):
                    if matrix[row1][col1] == pair[1]:
                        x1 = row1
                        y1 = col1
        ciphertextpair.append((matrix[x0][y1]) + (matrix[x1][y0]))

print(plaintextpair)
print(ciphertextpair)
```

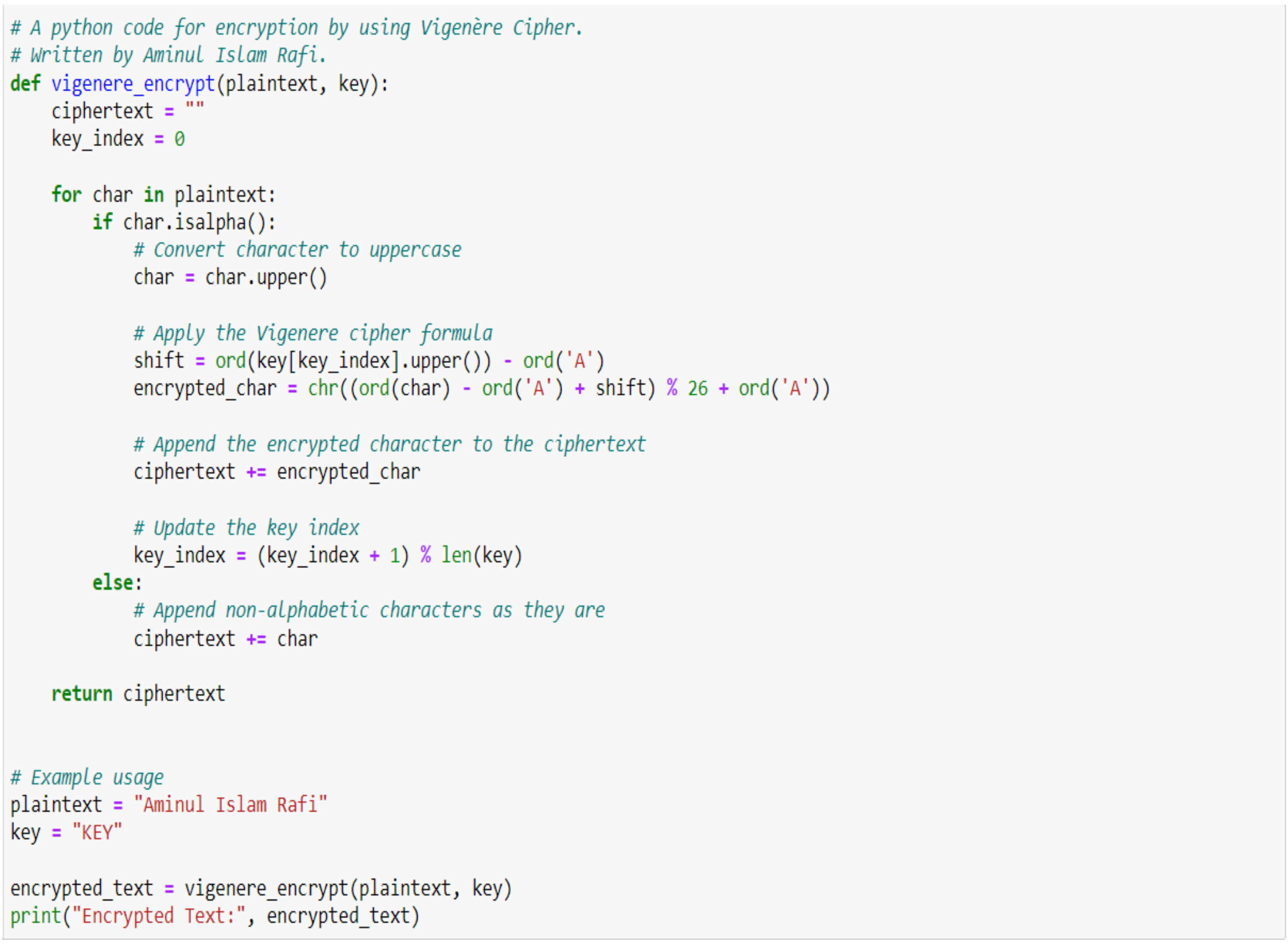
Input & Output

Top row contain the ciphertext pair and the second row contain the plaintext pair corresponding to this ciphertext.

```
['bm', 'od', 'zb', 'xd', 'na', 'be', 'ku', 'dm', 'ui', 'xm', 'mo', 'uv', 'if']
['hi', 'de', 'th', 'eg', 'ol', 'di', 'nt', 'he', 'tr', 'ex', 'es', 'tu', 'mp']
```

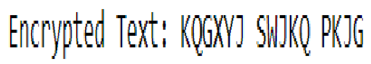
7. Write a python code for finding the encryption and decryption of a given text by using Vigenère cipher?

Encryption



```python
# A python code for encryption by using Vigenère Cipher.
# Written by Aminul Islam Rafi.
def vigenere_encrypt(plaintext, key):
    ciphertext = ""
    key_index = 0

    for char in plaintext:
        if char.isalpha():
            # Convert character to uppercase
            char = char.upper()

            # Apply the Vigenere cipher formula
            shift = ord(key[key_index].upper()) - ord('A')
            encrypted_char = chr((ord(char) - ord('A') + shift) % 26 + ord('A'))

            # Append the encrypted character to the ciphertext
            ciphertext += encrypted_char

            # Update the key index
            key_index = (key_index + 1) % len(key)
        else:
            # Append non-alphabetic characters as they are
            ciphertext += char

    return ciphertext


# Example usage
plaintext = "Aminul Islam Rafi"
key = "KEY"

encrypted_text = vigenere_encrypt(plaintext, key)
print("Encrypted Text:", encrypted_text)
```

Output



```
Encrypted Text: KQGXYJ SWJKQ PKJG
```

Decryption

```python
# A python code for decryption by using Vigenère Cipher.
# Written by Aminul Islam Rafi.
def vigenere_decrypt(ciphertext, key):
    plaintext = ""
    key_index = 0

    for char in ciphertext:
        if char.isalpha():
            # Convert character to uppercase
            char = char.upper()

            # Apply the Vigenere cipher formula to decrypt the character
            shift = ord(key[key_index].upper()) - ord('A')
            decrypted_char = chr((ord(char) - ord('A') - shift) % 26 + ord('A'))

            # Append the decrypted character to the plaintext
            plaintext += decrypted_char

            # Update the key index
            key_index = (key_index + 1) % len(key)
        else:
            # Append non-alphabetic characters as they are
            plaintext += char

    return plaintext


# Example usage
ciphertext = "RIJVS UYVJN"
key = "KEY"

decrypted_text = vigenere_decrypt(ciphertext, key)
print("Decrypted Text:", decrypted_text)
```

Output

```
Decrypted Text: HELLO WORLD
```

8. Write a python code for finding the encryption and decryption of a given text by using Hill cipher take the input from the file and save the output on to another file?

Encryption

```python
import numpy

# This is the key for encrypt the message.
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
            'v', 'w', 'x', 'y', 'z']
key = numpy.array([[17, 17, 5], [21, 18, 21], [2, 2, 19]])
plain = numpy.array([], dtype=int)
input = open("input.txt", "r+")
output = open("output.txt", "w")

plaintext = input.read();
# This line find the modulus.
check = len(plaintext) % len(key)
if check != 0:
    # This is used for find how many character left in plaintext for multiple of keylength.
    num=len(key)-check
    for i in range(num):
        plaintext = plaintext + 'z'
j = 0
result = ""
for i in range(len(plaintext)):
    plain = numpy.append(plain, alphabet.index(plaintext[i]))
    j = j + 1
    # If the array is fill for the multiplication then multiply it and find the cipher text for this group it is repeated un
    if j >= len(key):
        value = numpy.matmul(plain, key)
        for i in range(len(value)):
            m = value[i] % 26
            result = result + chr(m + 65)
        for i in range(len(key)):
            plain = numpy.delete(plain, 0)
        j = 0
output.write(result)
input.truncate(0)
input.close()
output.close()
```

Input

```
paymoremoney
```

Output

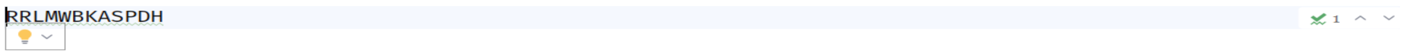
```
RRLMWBKASPDH
```

Decryption

```python
# Written by Aminul Islam Rafi.
import numpy
def MultiplicativeInverse(a, n):
    r1 = n
    r2 = a
    s1 = 1
    s2 = 0
    t1 = 0
    t2 = 1

    while r2 > 0:
        q = int(r1 / r2)
        r = r1 - q * r2
        r1 = r2
        r2 = r

        s = s1 - q * s2
        s1 = s2
        s2 = s

        t = t1 - q * t2
        t1 = t2
        t2 = t

    t = t1

    if r1 == 1:
        return t + n
    else:
        return " "


alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
            'V', 'W', 'X', 'Y', 'Z']
key = numpy.array([[17, 17, 5], [21, 18, 21], [2, 2, 19]])
plain = numpy.array([], dtype=int)
```

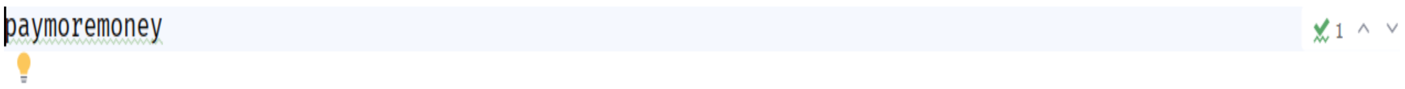
```python
# Find the inverse of the matrix.
inverse = numpy.linalg.inv(key)
# Find the determined of the matrix.
determine = numpy.linalg.det(key)
# Find the multiplicative inverse of the inverse determinded.
invdet = MultiplicativeInverse(determine % 26, 26)
# This two loop are used for the find the inverse of the key matrix.
for i in range(len(inverse)):
    for j in range(len(inverse)):
        # inverse[i][j] * determine is used to find the adjoint of the matrix so that we can calculate .
        value = numpy.round(inverse[i][j] * determine)
        key[i][j] = (invdet * (value % 26)) % 26

input = open("output.txt", "r+")
output = open("input.txt", "w")
ciphertext = input.read()
j = 0
result = ""
for i in range(len(ciphertext)):
    plain = numpy.append(plain, alphabet.index(ciphertext[i]))
    j = j + 1
    if j >= len(key):
        value = numpy.matmul(plain, key)
        for i in range(len(value)):
            m = value[i] % 26
            result = result + chr(m + 97)
        for i in range(3):
            plain = numpy.delete(plain, 0)
        j = 0

output.write(result)
input.truncate(0)
input.close()
output.close()
```

Input



```
RRLMWBKASPDH
```

Output



```
paymoremoney
```

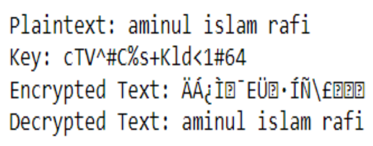
9.Write a python code for finding the encryption and decryption of a given text by using One Time Pad Cipher?

Encryption

```python
# A python code for encryption and decryption a text by using one time pad.
# Written by Aminul Islam Rafi.
import random

def generate_key(length):
    """Generate a random key of the specified length."""
    key = ""
    for _ in range(length):
        key += chr(random.randint(32, 126))  # Generates a random ASCII character in the printable range
    return key

def encrypt(plaintext, key):
    """Encrypt the plaintext using the provided key."""
    ciphertext = ""
    for i in range(len(plaintext)):
        char = plaintext[i]
        key_char = key[i]
        encrypted_char = chr((ord(char) + ord(key_char)) % 256)  # Modulo 256 for all ASCII characters
        ciphertext += encrypted_char
    return ciphertext

def decrypt(ciphertext, key):
    """Decrypt the ciphertext using the provided key."""
    plaintext = ""
    for i in range(len(ciphertext)):
        char = ciphertext[i]
        key_char = key[i]
        decrypted_char = chr((ord(char) - ord(key_char)) % 256)  # Modulo 256 for all ASCII characters
        plaintext += decrypted_char
    return plaintext

# Example usage:
plaintext = "aminul islam rafi"
key = generate_key(len(plaintext))
```

```python
print("Plaintext:", plaintext)
print("Key:", key)

encrypted_text = encrypt(plaintext, key)
print("Encrypted Text:", encrypted_text)

decrypted_text = decrypt(encrypted_text, key)
print("Decrypted Text:", decrypted_text)
```

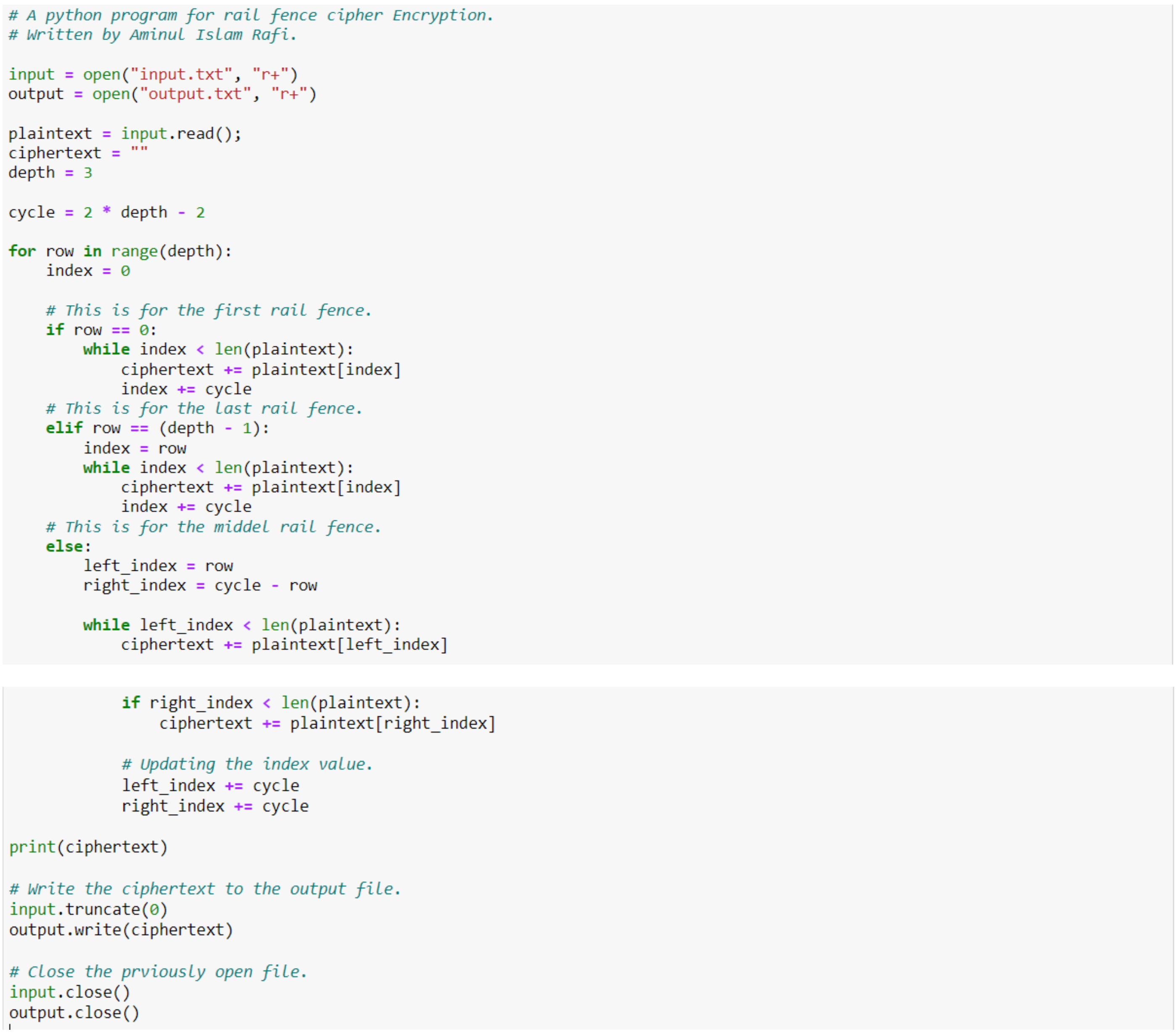
Output



```
Plaintext: aminul islam rafi
Key: cTV^#C%s+Kld<1#64
Encrypted Text: ÄÁ¿Ì□¯EÜ□·ÍÑ\£□□□
Decrypted Text: aminul islam rafi
```

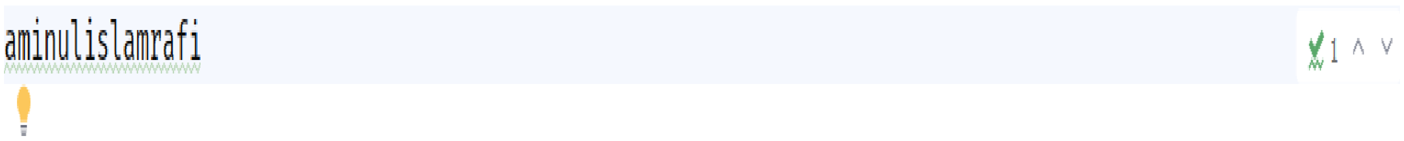
10.Write a python code for finding the encryption and decryption of a given text by using Rail fence cipher take the input from the file and save the output on to another file?

Encryption

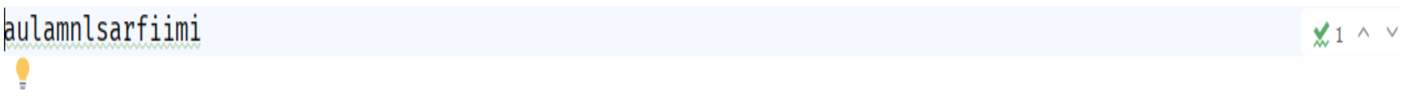


```python
# A python program for rail fence cipher Encryption.
# Written by Aminul Islam Rafi.

input = open("input.txt", "r+")
output = open("output.txt", "r+")

plaintext = input.read();
ciphertext = ""
depth = 3

cycle = 2 * depth - 2

for row in range(depth):
    index = 0

    # This is for the first rail fence.
    if row == 0:
        while index < len(plaintext):
            ciphertext += plaintext[index]
            index += cycle
    # This is for the last rail fence.
    elif row == (depth - 1):
        index = row
        while index < len(plaintext):
            ciphertext += plaintext[index]
            index += cycle
    # This is for the middel rail fence.
    else:
        left_index = row
        right_index = cycle - row

        while left_index < len(plaintext):
            ciphertext += plaintext[left_index]

            if right_index < len(plaintext):
                ciphertext += plaintext[right_index]

            # Updating the index value.
            left_index += cycle
            right_index += cycle

print(ciphertext)

# Write the ciphertext to the output file.
input.truncate(0)
output.write(ciphertext)

# Close the prviously open file.
input.close()
output.close()
```

Input

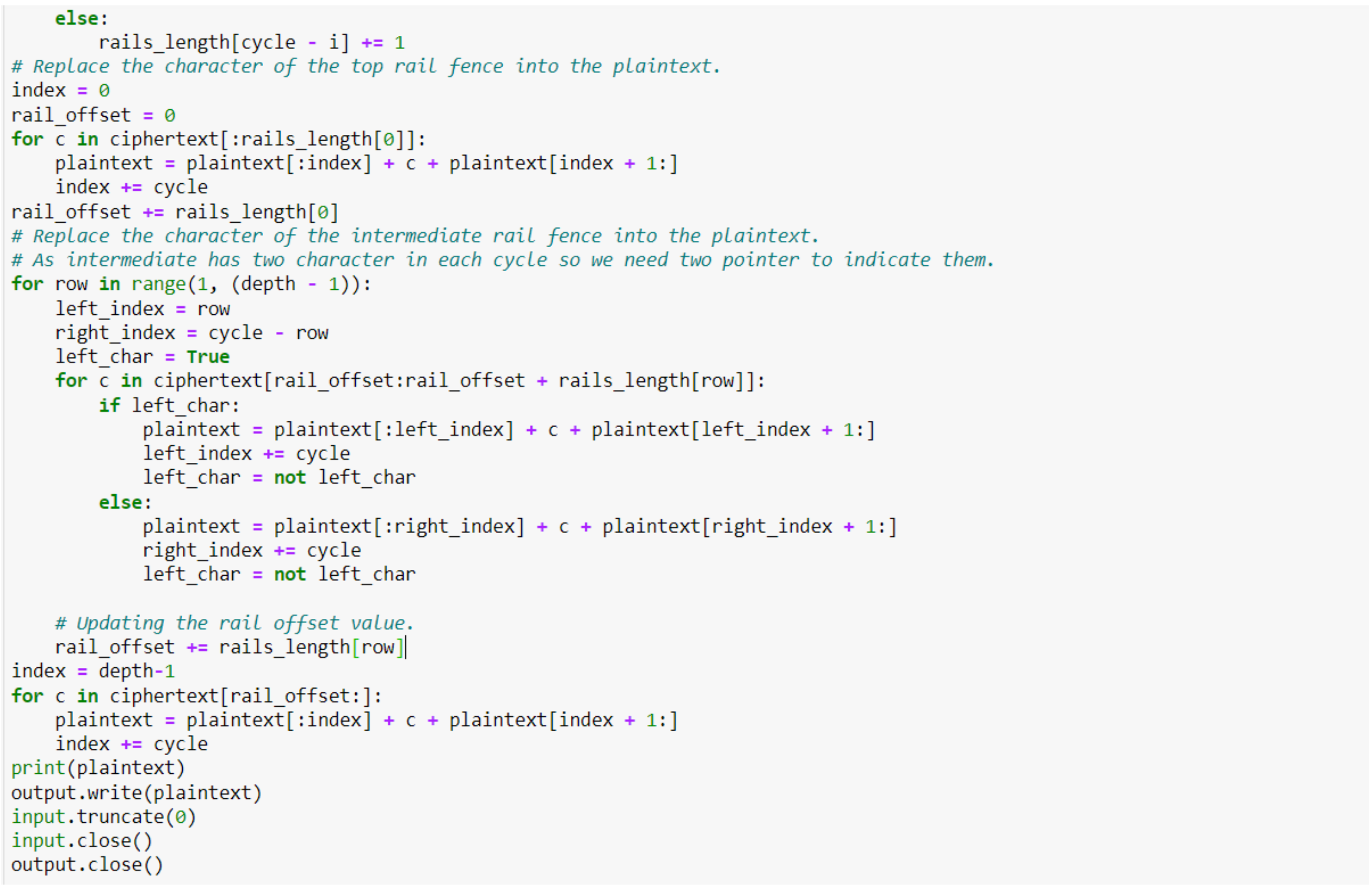

aminulislamrafi

Output


aulamnlsarfiimi

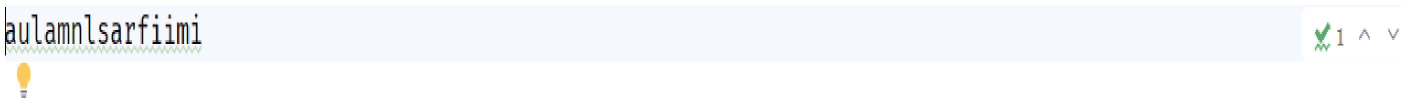
Decryption

```python
# A python program for rail fence cipher decryption.
# Written by Aminul Islam Rafi.

input = open("output.txt", "r+")
output = open("input.txt", "r+")

ciphertext = input.read()
depth = 3
cycle = 2 * depth - 2
length = len(ciphertext)

# This makes the plaintext length as the same length of ciphertext.
plaintext = "." * length

# Integer division gives the result same as(Math.floor()).
units = length // cycle

# Make a list of size equal to key.
rails_length = [0] * depth

# Top Rail Length.
rails_length[0] = units

# Intermediate Rail Length.
for i in range(1, depth - 1):
    rails_length[i] = 2 * units
# Bottom Rail Length.
rails_length[depth - 1] = units

# If the length of the ciphertext is not completely divisable by cycle then
# This loop work.
for i in range(length % cycle):
    if i < depth:
        rails_length[i] += 1
```



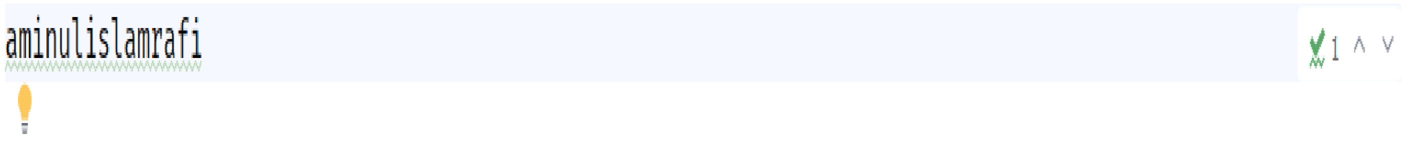
```python
    else:
        rails_length[cycle - i] += 1
# Replace the character of the top rail fence into the plaintext.
index = 0
rail_offset = 0
for c in ciphertext[:rails_length[0]]:
    plaintext = plaintext[:index] + c + plaintext[index + 1:]
    index += cycle
rail_offset += rails_length[0]
# Replace the character of the intermediate rail fence into the plaintext.
# As intermediate has two character in each cycle so we need two pointer to indicate them.
for row in range(1, (depth - 1)):
    left_index = row
    right_index = cycle - row
    left_char = True
    for c in ciphertext[rail_offset:rail_offset + rails_length[row]]:
        if left_char:
            plaintext = plaintext[:left_index] + c + plaintext[left_index + 1:]
            left_index += cycle
            left_char = not left_char
        else:
            plaintext = plaintext[:right_index] + c + plaintext[right_index + 1:]
            right_index += cycle
            left_char = not left_char

    # Updating the rail offset value.
    rail_offset += rails_length[row]
index = depth-1
for c in ciphertext[rail_offset:]:
    plaintext = plaintext[:index] + c + plaintext[index + 1:]
    index += cycle
print(plaintext)
output.write(plaintext)
input.truncate(0)
input.close()
output.close()
```

Input



```
aulamnlsarfiimi
```

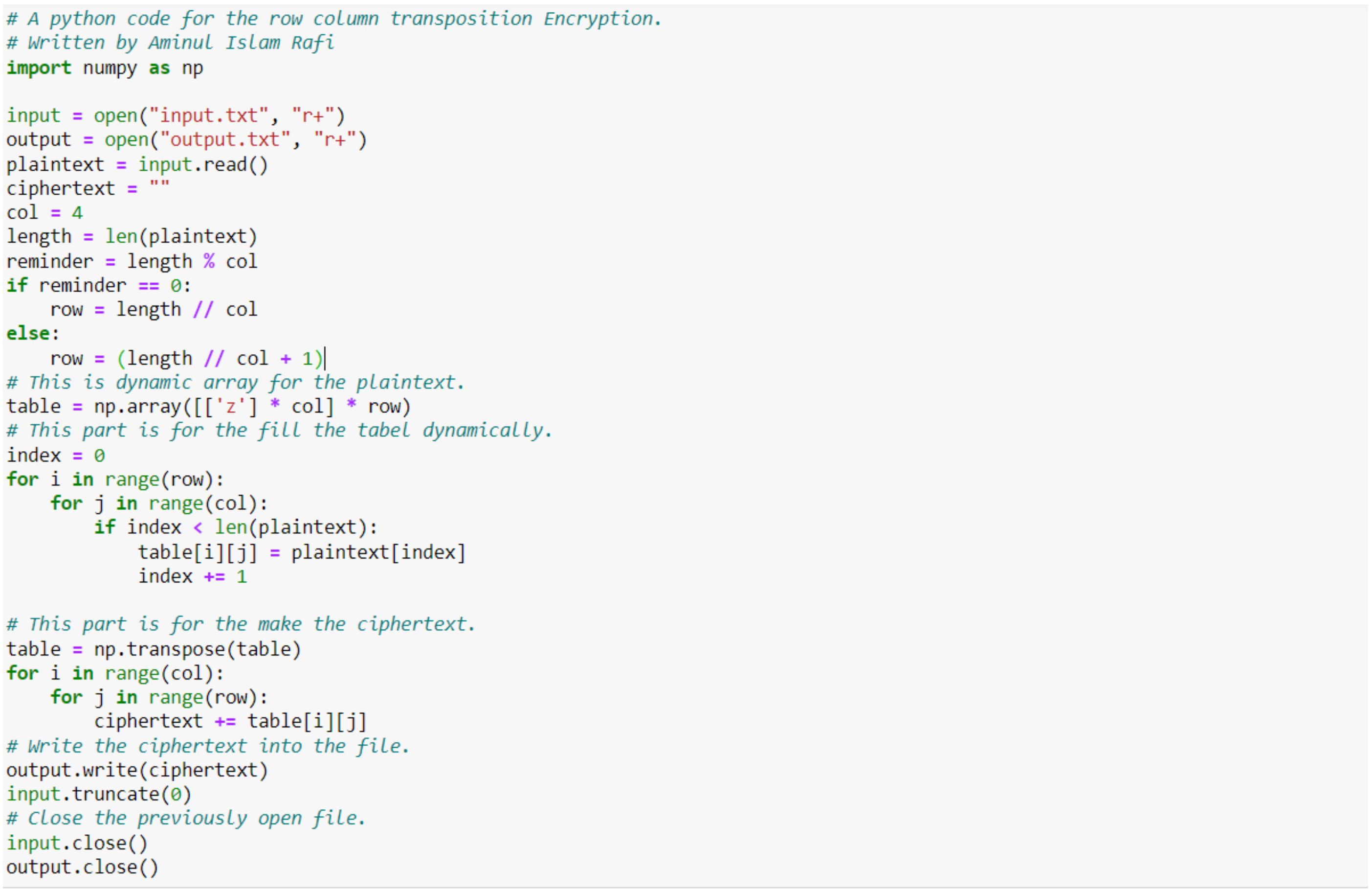
Output



```
aminulislamrafi
```

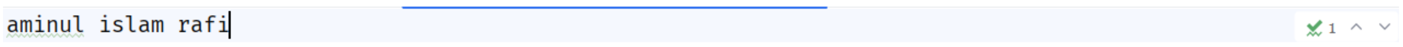
11.Write a python code for finding the encryption and decryption of a given text by using keyless row column transposition cipher take the input from the file and save the output on to another file?

Encryption



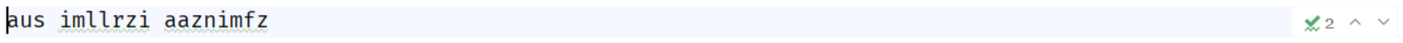
```python
# A python code for the row column transposition Encryption.
# Written by Aminul Islam Rafi
import numpy as np

input = open("input.txt", "r+")
output = open("output.txt", "r+")
plaintext = input.read()
ciphertext = ""
col = 4
length = len(plaintext)
reminder = length % col
if reminder == 0:
    row = length // col
else:
    row = (length // col + 1)
# This is dynamic array for the plaintext.
table = np.array([['z'] * col] * row)
# This part is for the fill the tabel dynamically.
index = 0
for i in range(row):
    for j in range(col):
        if index < len(plaintext):
            table[i][j] = plaintext[index]
            index += 1

# This part is for the make the ciphertext.
table = np.transpose(table)
for i in range(col):
    for j in range(row):
        ciphertext += table[i][j]
# Write the ciphertext into the file.
output.write(ciphertext)
input.truncate(0)
# Close the previously open file.
input.close()
output.close()
```

Input



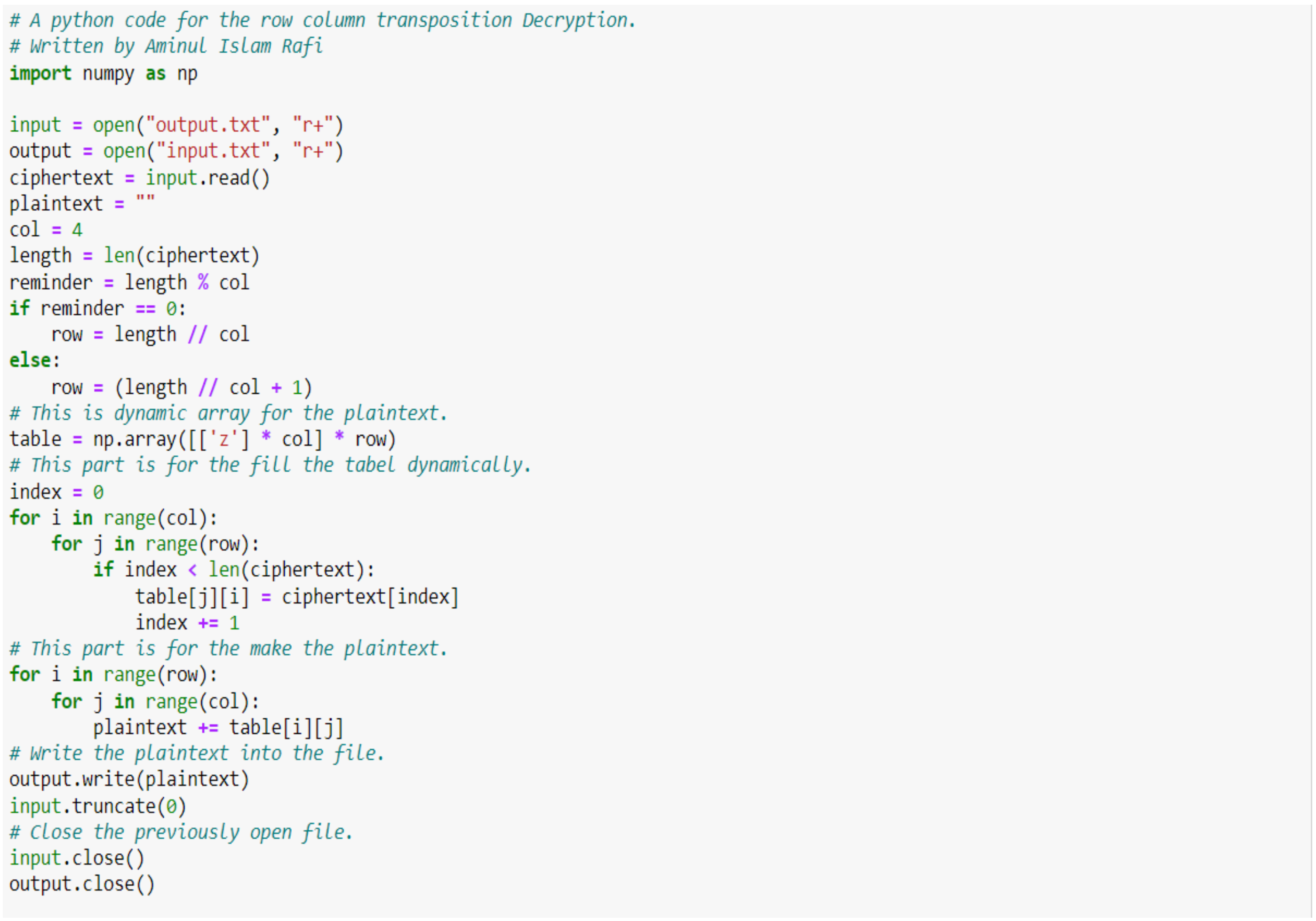
```
aminul islam rafi
```

Output



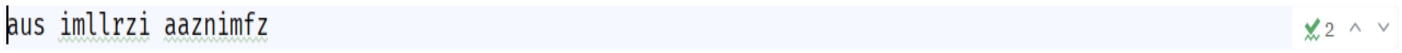
```
aus imllrzi aaznimfz
```

Decryption



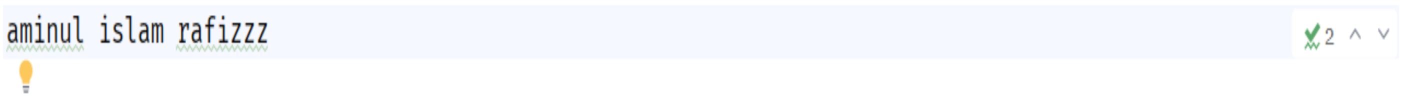
```python
# A python code for the row column transposition Decryption.
# Written by Aminul Islam Rafi
import numpy as np

input = open("output.txt", "r+")
output = open("input.txt", "r+")
ciphertext = input.read()
plaintext = ""
col = 4
length = len(ciphertext)
reminder = length % col
if reminder == 0:
    row = length // col
else:
    row = (length // col + 1)
# This is dynamic array for the plaintext.
table = np.array([['z'] * col] * row)
# This part is for the fill the tabel dynamically.
index = 0
for i in range(col):
    for j in range(row):
        if index < len(ciphertext):
            table[j][i] = ciphertext[index]
            index += 1
# This part is for the make the plaintext.
for i in range(row):
    for j in range(col):
        plaintext += table[i][j]
# Write the plaintext into the file.
output.write(plaintext)
input.truncate(0)
# Close the previously open file.
input.close()
output.close()
```

Input



```
aus imllrzi aaznimfz
```

Output



```
aminul islam rafizzz
```

12. Write a python code for finding the encryption and decryption of a given text by using keyed row transposition cipher take the input from the file and save the output on to another file?

Encryption

```python
# A python code for encryption keyed transposition cipher.
# Written by Aminul Islam Rafi.
import numpy as np
input = open("input.txt", "r+")
output = open("output.txt", "r+")
plaintext = input.read()
ciphertext = ""
# This block size is dynamic you can give any value.
blockSize = 4
value = len(plaintext) % blockSize
# This is used for the adding extra dummy character in the plaintext.
if value != 0:
    for i in range(blockSize - value):
        plaintext += "z"
# This line used for find how many block needed for represent the plaintext.
value = int(len(plaintext) / blockSize)
print(value)
key = [2, 0, 3, 4, 1]
list = np.array([['z'] * value] * blockSize)
list1 = np.array([['z'] * value] * blockSize)
index = 0
for i in range(blockSize):
    for j in range(value):
        list[i][j] = plaintext[index]
        index += 1
# Perform the transposition.
for i in range(blockSize):
    index = 0
    for j in range(value):
        list1[i][j] = list[i][key[index]]
        ciphertext += list1[i][j]
        index += 1
print("The required ciphertext is : ", ciphertext)
output.write(ciphertext)
input.truncate(0)
input.close()
output.close()
```

Input

```
enemyattackstonight
```

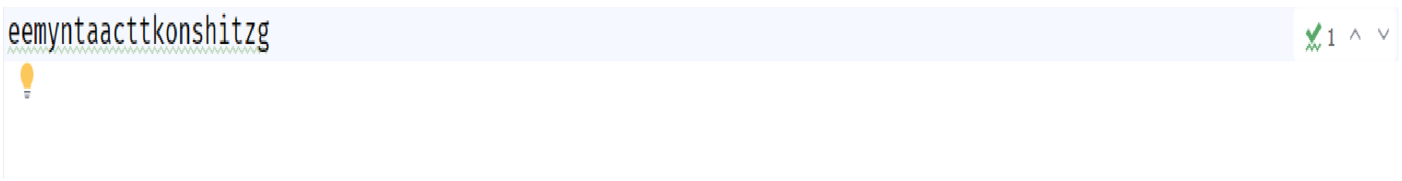
Output

```
eemyntaacttkonshitzg
```

Decryption

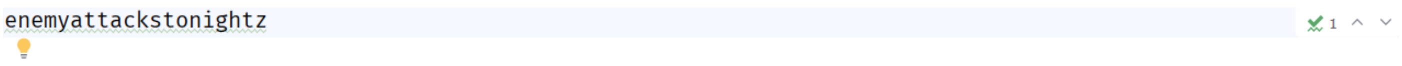
```python
# A python code for encryption keyed transposition cipher.
# Written by Aminul Islam Rafi.
import numpy as np
input = open("output.txt", "r+")
output = open("input.txt", "r+")
plaintext = input.read()
ciphertext = ""
# This block size is dynamic you can give any value.
blockSize = 4
# This line used for find how many block needed for represent the plaintext.
value = int(len(plaintext) / blockSize)
print(value)
# This is key need for transfer the plaintext block.
key = [1, 4, 0, 2, 3]
list = np.array([['z'] * value] * blockSize)
list1 = np.array([['z'] * value] * blockSize)
index = 0
# Represent the plaintext into the matrix so that we can transpose the character .
for i in range(blockSize):
    for j in range(value):
        list[i][j] = plaintext[index]
        index += 1
# Perform the transposition.
for i in range(blockSize):
    index = 0
    for j in range(value):
        list1[i][j] = list[i][key[index]]
        ciphertext += list1[i][j]
        index += 1
print("The required ciphertext is : ", ciphertext)
output.write(ciphertext)
input.truncate(0)
input.close()
output.close()
```

Input



```
eemyntaacttkonshitzg
```

Output



```
enemyattackstonightz
```

13.Write a python code for finding the encryption and decryption of a given text by using keyed row column transposition cipher (Combining two approach)?

Encryption

```python
# A python code for the combine transposition cipher.
# Written by Aminul Islam Rafi.
import numpy as np
plaintext = "enemyattackstonight"
ciphertext=""
key = [2, 0, 3, 4, 1]
value = len(key)
length = len(plaintext)

if length % value == 0:
    col = int(length / value)
else:
    col = int((length / value) + 1)

plainMatrix=np.array([[25]*value]*col)
keyMatrix = np.array([[0]*value]*value)

index=0
for i in range(col):
  for j in range(value):
    if index < length:
      num=ord(plaintext[index])-97
      plainMatrix[i][j] = num
      index += 1
j=0
for i in range(len(key)):
  keyMatrix[key[i]][j]=1
  j+=1

cipherMatrix=np.matmul(plainMatrix,keyMatrix)

for i in range(value):
  for j in range(col):
    value = (cipherMatrix[j][i]+65)
    ciphertext += chr(value)
print("Ciphertext is:",ciphertext)
```

Output

```
Ciphertext is: ETTHEAKIMAOTYCNZNTSG
```

Decryption

```
# A python code for the combine transposition cipher.
# Written by Aminul Islam Rafi.
import numpy as np
ciphertext = "ETTHEAKIMAOTYCNZNTSG"
plaintext = ""
key = [2, 0, 3, 4, 1]
value = int(len(key))
length = len(ciphertext)
col = int(length / value)
cipherMatrix = np.array([[0] * col] * value)
keyMatrix = np.array([[0] * value] * value)

index = 0
for i in range(value):
    for j in range(col):
        if index < length:
            num = ord(ciphertext[index]) - 65
            cipherMatrix[i][j] = num
            index += 1
j = 0
for i in range(len(key)):
    keyMatrix[key[i]][j] = 1
    j += 1
# Perform this transposition operation on cipherMatrix and keyMatrix.
cipherMatrix = np.transpose(cipherMatrix)
keyMatrix = np.linalg.inv(keyMatrix)
plainMatrix = np.matmul(cipherMatrix, keyMatrix)
for i in range(col):
    for j in range(value):
        result = (int(plainMatrix[i][j]) + 97)
        plaintext += chr(result)
print(plaintext)
```

Output

```
enemyattackstonightz
```

Modern Block Cipher

14.Write a python code for finding the encryption and decryption of a given text by using Data Encryption Standard Cipher?

Encryption

```python
# A python code for encryption by using the data encryption standard.
# Written by Aminul Islam Rafi

# Import the des function from the Crypto Module.
import base64
from Crypto.Cipher import DES

# This is the key for the encryption.
key = 'hello123'

# This function is for the added the extra text to the plaintext if it is not fill
# The block of 64 bit's size.
def pad(text):
    # DES Algorithm works with 8 byte thats why we divided it by the 8.
    # If the given text is multiple of 8 then exit form the while loop.
    while len(text) % 8 != 0:
        text += ' '
    return text

# Create the object of the DES and pass the appropriate parameter.
# Second parameter is the encryption mode . Several encryption mode are available.
des = DES.new(key.encode('utf-8'),DES.MODE_ECB)  # 'utf-8' encode the string to bytes.As Des require the byte as parameter.

text = 'aminul islam rafi'
padded_text = pad(text)

# This encrypt function is uesed to encrypt the message.
encrypted_text = des.encrypt(padded_text.encode('utf-8'))

print("The Encrypted text is :",encrypted_text)
print("The ecnrypted text in character formate :",base64.b64encode(encrypted_text))
```

Output

```
The Encrypted text is : b';\x9e\xc9\xac\x9d\xdb\t}\xe6\xf2\x13h\xc5\xdf\xf9\xc2\xbf\x14\xfeqT\xa2Ag'
The ecnrypted text in character formate : b'O57JrJ3bCX3m8hNoxd/5wr8U/nFUokFn'
```

Decryption

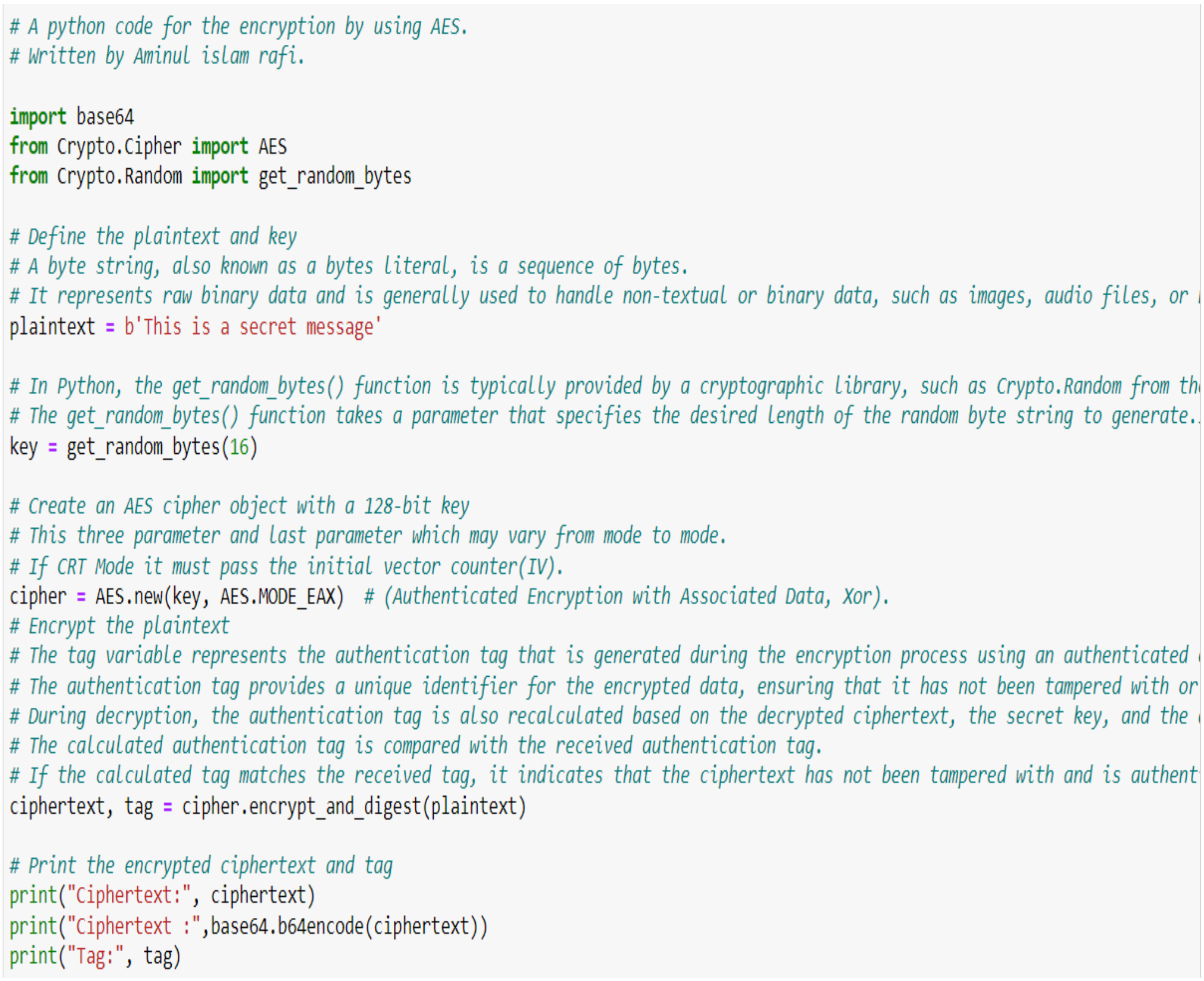
```python
# Create the object of the DES and pass the appropriate parameter.
# Second parameter is the encryption mode . Several encryption mode are available.
des = DES.new(key.encode('utf-8'),DES.MODE_ECB)  # 'utf-8' encode the string to bytes.As Des require the byte as parameter.
# This encrypt function is uesed to encrypt the message.
decrypted_text = des.decrypt(encrypted_text)
print("The Decrypted text is :",decrypted_text.decode())
```

Output

```
The Decrypted text is : aminul islam rafi
```

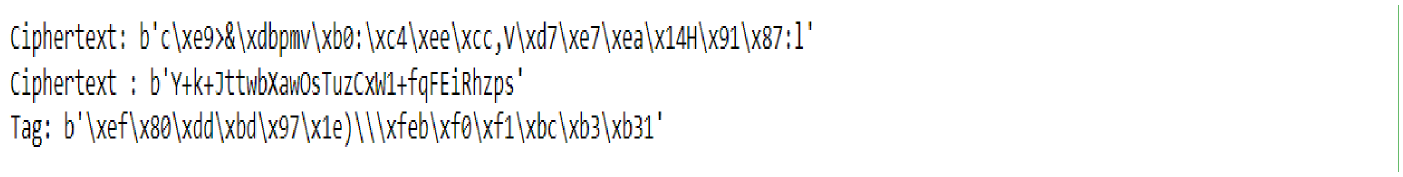
15.Write a python code for finding the encryption and decryption of a given text by using Advance Encryption Standard Cipher?

Encryption



```python
# A python code for the encryption by using AES.
# Written by Aminul islam rafi.

import base64
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

# Define the plaintext and key
# A byte string, also known as a bytes literal, is a sequence of bytes.
# It represents raw binary data and is generally used to handle non-textual or binary data, such as images, audio files, or
plaintext = b'This is a secret message'

# In Python, the get_random_bytes() function is typically provided by a cryptographic library, such as Crypto.Random from th
# The get_random_bytes() function takes a parameter that specifies the desired length of the random byte string to generate.
key = get_random_bytes(16)

# Create an AES cipher object with a 128-bit key
# This three parameter and last parameter which may vary from mode to mode.
# If CRT Mode it must pass the initial vector counter(IV).
cipher = AES.new(key, AES.MODE_EAX)  # (Authenticated Encryption with Associated Data, Xor).
# Encrypt the plaintext
# The tag variable represents the authentication tag that is generated during the encryption process using an authenticated
# The authentication tag provides a unique identifier for the encrypted data, ensuring that it has not been tampered with or
# During decryption, the authentication tag is also recalculated based on the decrypted ciphertext, the secret key, and the
# The calculated authentication tag is compared with the received authentication tag.
# If the calculated tag matches the received tag, it indicates that the ciphertext has not been tampered with and is authent
ciphertext, tag = cipher.encrypt_and_digest(plaintext)

# Print the encrypted ciphertext and tag
print("Ciphertext:", ciphertext)
print("Ciphertext :",base64.b64encode(ciphertext))
print("Tag:", tag)
```

Output



```
Ciphertext: b'c\xe9>&\xdbpmv\xb0:\xc4\xee\xcc,V\xd7\xe7\xea\x14H\x91\x87:l'
Ciphertext : b'Y+k+JttwbXawOsTuzCxW1+fqFEiRhzps'
Tag: b'\xef\x80\xdd\xbd\x97\x1e)\\\xfeb\xf0\xf1\xbc\xb3\xb31'
```

Decryption

```python
# Create a new AES cipher object with the same key
# The nonce (number used once) is a crucial component in symmetric key encryption algorithms, particularly in modes like AES
# It serves as an additional input to the encryption algorithm and helps ensure the uniqueness and security of the ciphertex
decrypt_cipher = AES.new(key, AES.MODE_EAX, nonce=cipher.nonce)

# Decrypt the ciphertext
decrypted_plaintext = decrypt_cipher.decrypt_and_verify(ciphertext, tag)

# Print the decrypted plaintext
print("Decrypted plaintext:", decrypted_plaintext.decode())
```

Output

```
Decrypted plaintext: This is a secret message
```

16.Write a python code for finding the encryption and decryption of a given text by using ElGamal Cryptography?

Encryption

```python
# A python code for encrpting the text by using the ElGamal Cryptosystem.
# Written by Aminul Islam Rafi.
# Sympy is a Python library for symbolic mathematics.
import random
from sympy import primitive_root,randprime
# The number for which you want to find the primitive root
prime = randprime(124,10**3)
root = primitive_root(prime)
d=random.randint(1,(prime-2)) # It is private key.
e=(pow(root,d)%prime)  # It is public key.
r=random.randint(1,10)  # Select a random integer.
#Define the plaintext.
plaintext = "Aminul Islam Rafi"
# Encryption Algorithm.
ciphertext=[]
for char in plaintext:
  ciphertext1=(pow(root,r)%prime)
  ciphertext2=((ord(char)*pow(e,r))%prime)
  ciphertext.append((ciphertext1,ciphertext2))
print(ciphertext)
```

Output

```
[(3, 537), (3, 423), (3, 299), (3, 454), (3, 671), (3, 392), (3, 253), (3, 46), (3, 609), (3, 392), (3, 51), (3, 423), (3, 25
3), (3, 325), (3, 51), (3, 206), (3, 299)]
```

Decryption

```python
#Decryption Algorithm
plaintext=""
for pair in ciphertext:
  ciphertext1,ciphertext2=pair
  value=pow(ciphertext1,d)
  multinv = pow(value,-1,prime)
  decrypt_char = (ciphertext2*multinv) % prime
  plaintext += chr(decrypt_char)
print(plaintext)
```

Output

```
Aminul Islam Rafi
```

17.Write a python code for finding the encryption and decryption of a given text by using RSA Cryptography?

Encryption

```python
# A python code for encrpting the text by using the ElGamal Cryptosystem.
# Written by Aminul Islam Rafi.
from sympy import randprime
import random
import math

# Generate two random prime numbers
prime1 = randprime(2, 10**2)  # Adjust the range as needed
prime2 = randprime(2, 10**2)  # Adjust the range as needed
n = prime1 * prime2
phin = (prime1 - 1) * (prime2 - 1)

# Generating the public key
e = randprime(1, phin)
while math.gcd(phin, e) != 1:
    e = random.randint(1, phin)

# Generating the private key.
d = pow(e, -1, phin)

plaintext = "Aminul Islam Rafi"

# Encryption Algorithm
i = 0
ciphertext = ""
length = len(plaintext)
while i < length:
    value = ord(plaintext[i])
    num = pow(value, e) % n
    ciphertext += chr(num + 32)  # Add 32 to get the correct ASCII character
    i += 1
print("The cipher text of the plaintext :",plaintext," is: ",ciphertext)
```

Output

```
The cipher text of the plaintext : Aminul Islam Rafi  is:  WVzĆΘhiL·¢hłViKł7z
```

Decryption

```python
# Decryption Algorithm
i = 0
decrypted_text = ""
length = len(ciphertext)
while i < length:
    value = ord(ciphertext[i]) - 32  # Subtract 32 to get the original numerical value
    num = pow(value, d) % n
    decrypted_text += chr(num)
    i += 1
print("The plaintext of the ciphertext :",ciphertext," is: ",decrypted_text)
```

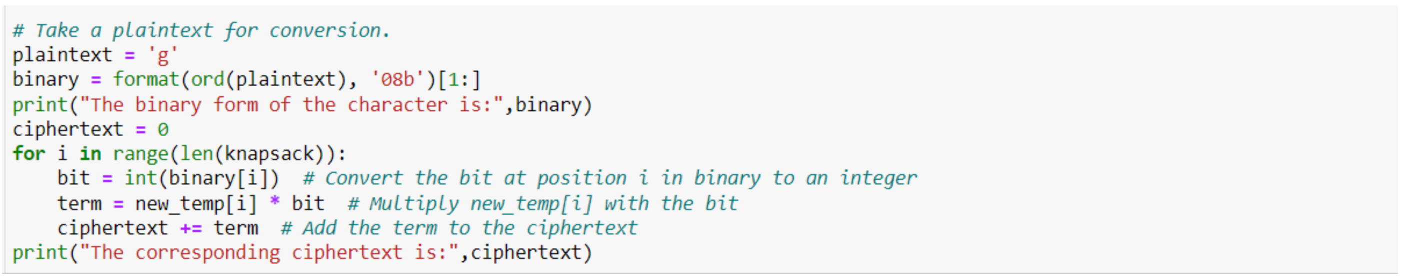
Output

```
The plaintext of the ciphertext : ШVzĆΘhiL₵hɨViKɨ7z  is:  Aminul Islam Rafi
```

18.Write a python code for finding the encryption and decryption of a given text by using Knapsack Cryptography?

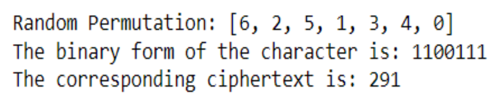
Encryption

```python
# A python code for encrypting the text using knapsack cryptosystem.
# Written by Aminul Islam Rafi.
import math
import random
from sympy import randprime

# Generate a superincreasing knapsack
def generate_superincreasing_knapsack(n):
    knapsack = [1]
    for _ in range(n - 1):
        knapsack.append(sum(knapsack) + 1)
    return knapsack

# Generate a superincreasing knapsack
knapsack = generate_superincreasing_knapsack(7)

# Encrypt a message
n = sum(knapsack) + 5

r = randprime(1, n - 1)
while math.gcd(n, r) != 1:
    r = randprime(2, n - 1)
temp = [(r * value) % n for value in knapsack]

random_permutation = random.sample(range(7), 7)
print("Random Permutation:", random_permutation)

new_temp = [0] * 7
for i in range(len(temp)):
    new_temp[i] = temp[random_permutation[i]]
```



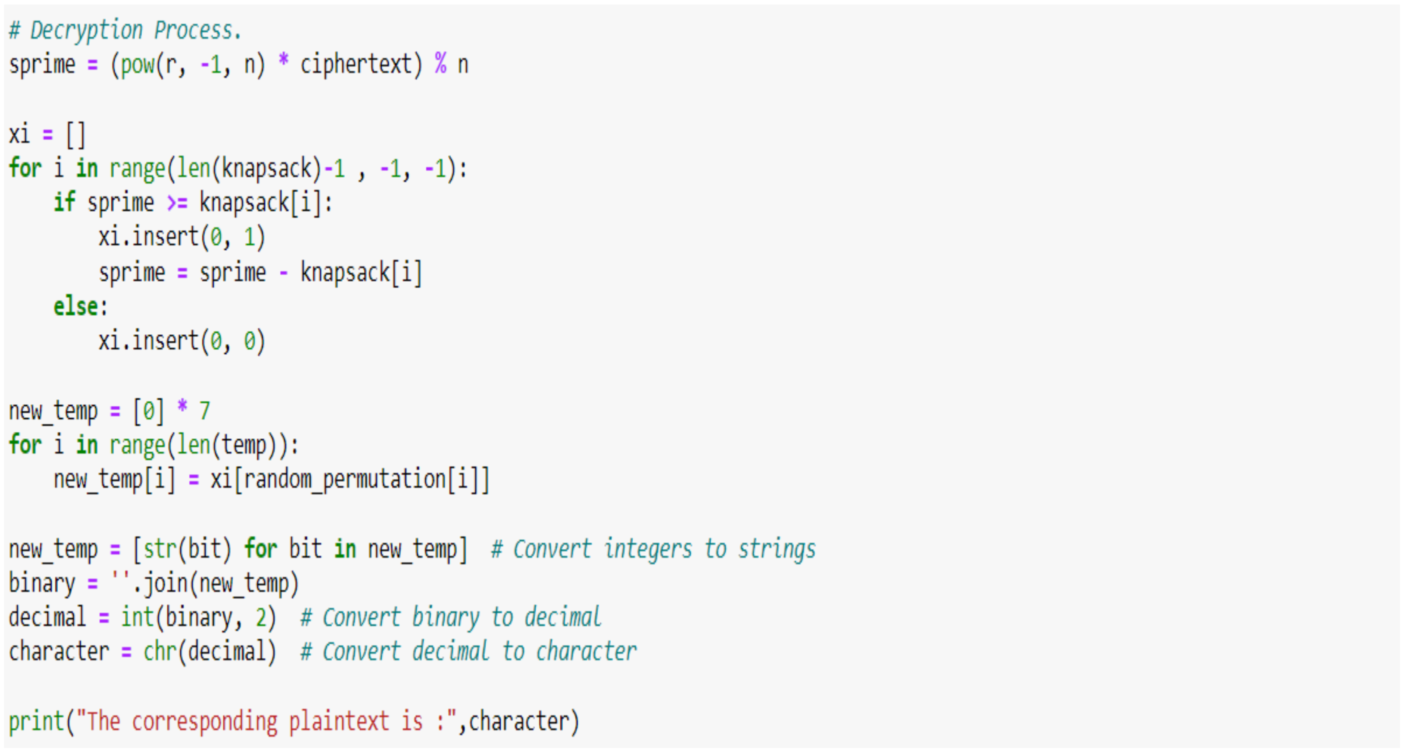
```python
# Take a plaintext for conversion.
plaintext = 'g'
binary = format(ord(plaintext), '08b')[1:]
print("The binary form of the character is:",binary)
ciphertext = 0
for i in range(len(knapsack)):
    bit = int(binary[i])  # Convert the bit at position i in binary to an integer
    term = new_temp[i] * bit  # Multiply new_temp[i] with the bit
    ciphertext += term  # Add the term to the ciphertext
print("The corresponding ciphertext is:",ciphertext)
```

Output



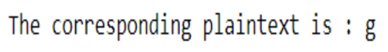
```
Random Permutation: [6, 2, 5, 1, 3, 4, 0]
The binary form of the character is: 1100111
The corresponding ciphertext is: 291
```

Decryption



```python
# Decryption Process.
sprime = (pow(r, -1, n) * ciphertext) % n

xi = []
for i in range(len(knapsack)-1 , -1, -1):
    if sprime >= knapsack[i]:
        xi.insert(0, 1)
        sprime = sprime - knapsack[i]
    else:
        xi.insert(0, 0)

new_temp = [0] * 7
for i in range(len(temp)):
    new_temp[i] = xi[random_permutation[i]]

new_temp = [str(bit) for bit in new_temp]  # Convert integers to strings
binary = ''.join(new_temp)
decimal = int(binary, 2)  # Convert binary to decimal
character = chr(decimal)  # Convert decimal to character

print("The corresponding plaintext is :",character)
```

Output

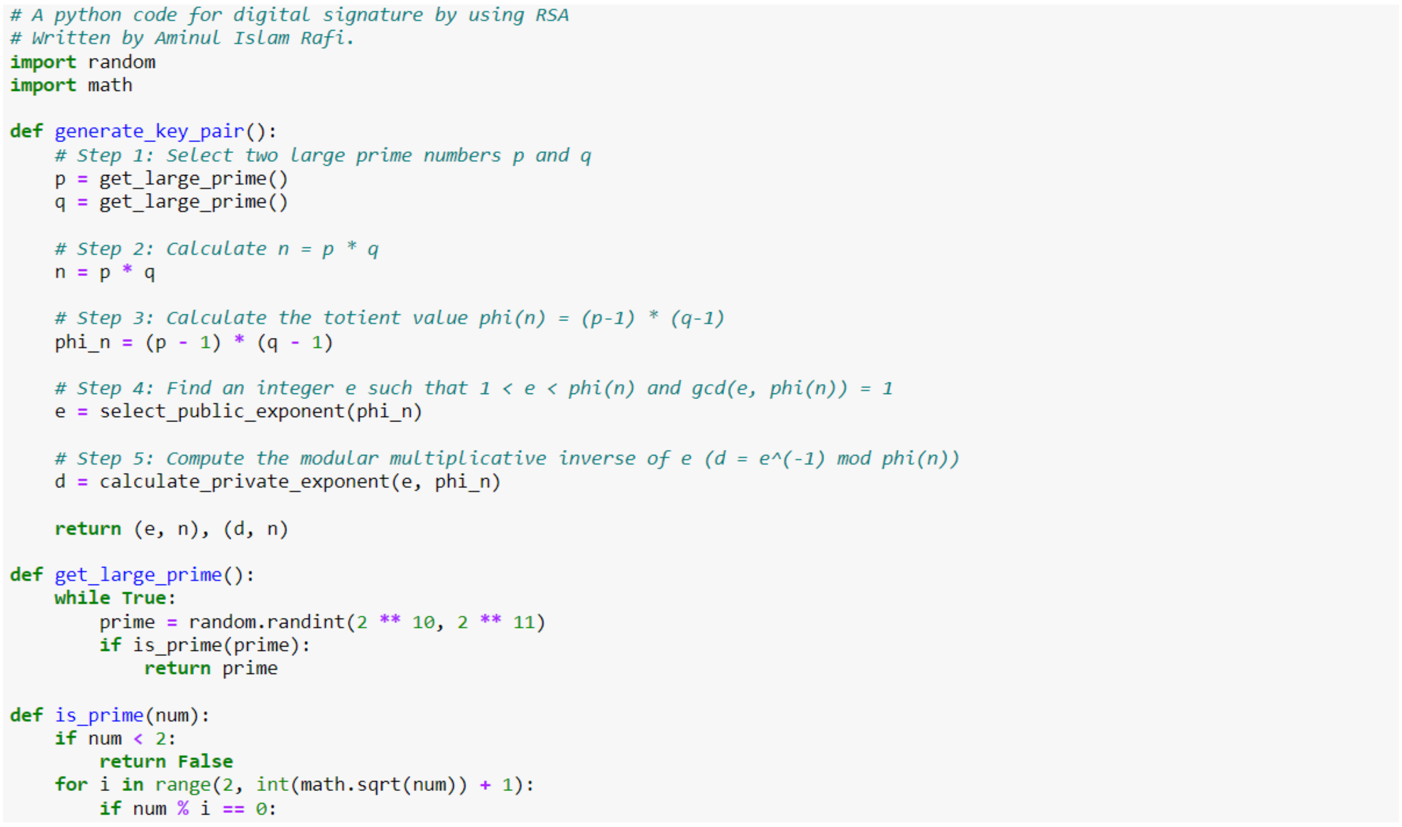


```
The corresponding plaintext is : g
```

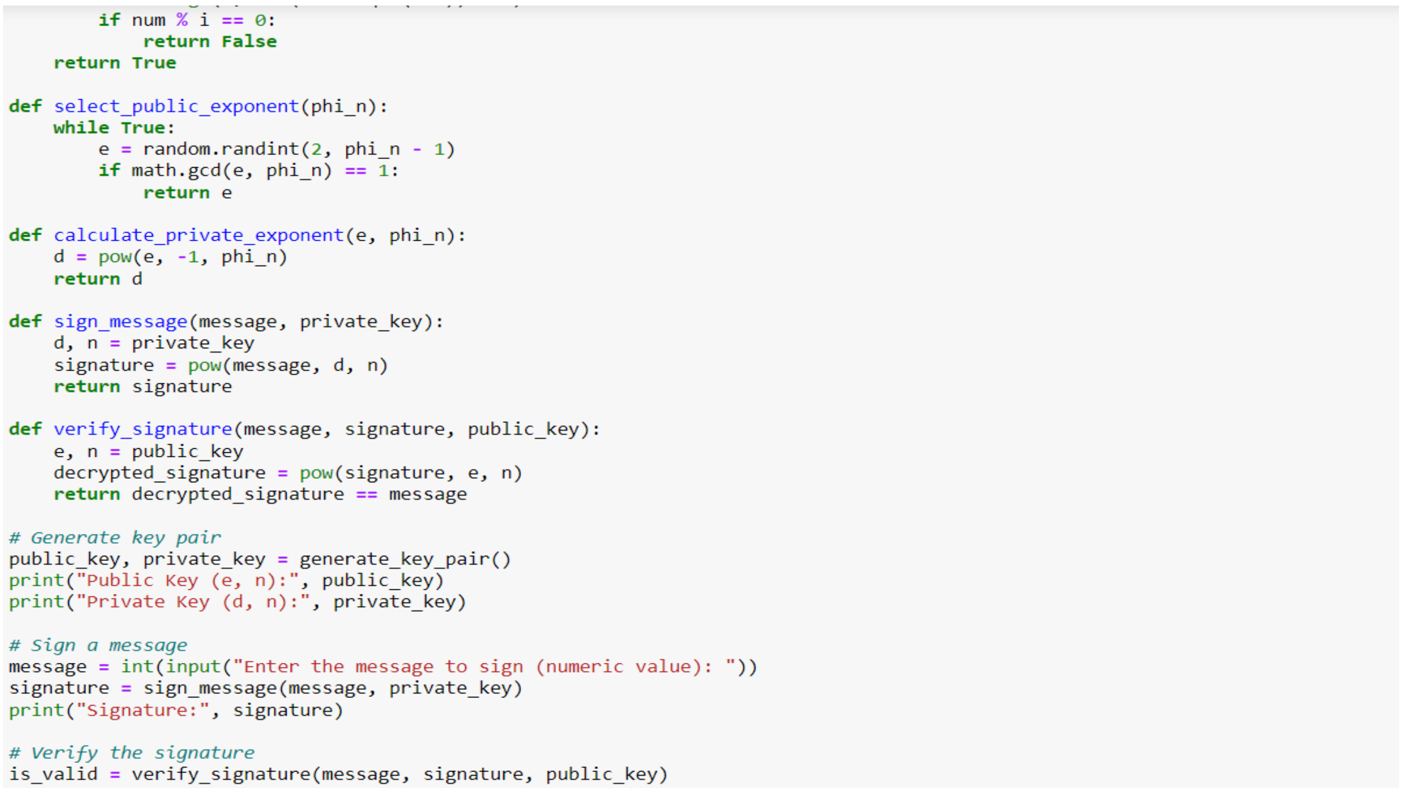
Digital Signature

19.Write a python code for signing the message by using RSA digital signature scheme?

Code



```python
# A python code for digital signature by using RSA
# Written by Aminul Islam Rafi.
import random
import math

def generate_key_pair():
    # Step 1: Select two large prime numbers p and q
    p = get_large_prime()
    q = get_large_prime()

    # Step 2: Calculate n = p * q
    n = p * q

    # Step 3: Calculate the totient value phi(n) = (p-1) * (q-1)
    phi_n = (p - 1) * (q - 1)

    # Step 4: Find an integer e such that 1 < e < phi(n) and gcd(e, phi(n)) = 1
    e = select_public_exponent(phi_n)

    # Step 5: Compute the modular multiplicative inverse of e (d = e^(-1) mod phi(n))
    d = calculate_private_exponent(e, phi_n)

    return (e, n), (d, n)

def get_large_prime():
    while True:
        prime = random.randint(2 ** 10, 2 ** 11)
        if is_prime(prime):
            return prime

def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
```



```python
        if num % i == 0:
            return False
    return True

def select_public_exponent(phi_n):
    while True:
        e = random.randint(2, phi_n - 1)
        if math.gcd(e, phi_n) == 1:
            return e

def calculate_private_exponent(e, phi_n):
    d = pow(e, -1, phi_n)
    return d

def sign_message(message, private_key):
    d, n = private_key
    signature = pow(message, d, n)
    return signature

def verify_signature(message, signature, public_key):
    e, n = public_key
    decrypted_signature = pow(signature, e, n)
    return decrypted_signature == message

# Generate key pair
public_key, private_key = generate_key_pair()
print("Public Key (e, n):", public_key)
print("Private Key (d, n):", private_key)

# Sign a message
message = int(input("Enter the message to sign (numeric value): "))
signature = sign_message(message, private_key)
print("Signature:", signature)

# Verify the signature
is_valid = verify_signature(message, signature, public_key)
```

```
if is_valid:
    print("Signature is valid.")
else:
    print("Signature is not valid.")
```

Output

```
Public Key (e, n): (1246455, 1804147)
Private Key (d, n): (345431, 1804147)
Enter the message to sign (numeric value): 12
Signature: 414530
Signature is valid.
```

End