



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

CRYPTOGRAPHY AND NETWORK SECURITY

LAB 2 : WRITE A PROGRAM IN SUITABLE PROGRAMMING LANGUAGE FOR THE FOLLOWING:

- **1. VIGENERE CIPHER**
- **2. RAIL FENCE CIPHER**
- **3. EUCLIDEAN ALGORITHM**

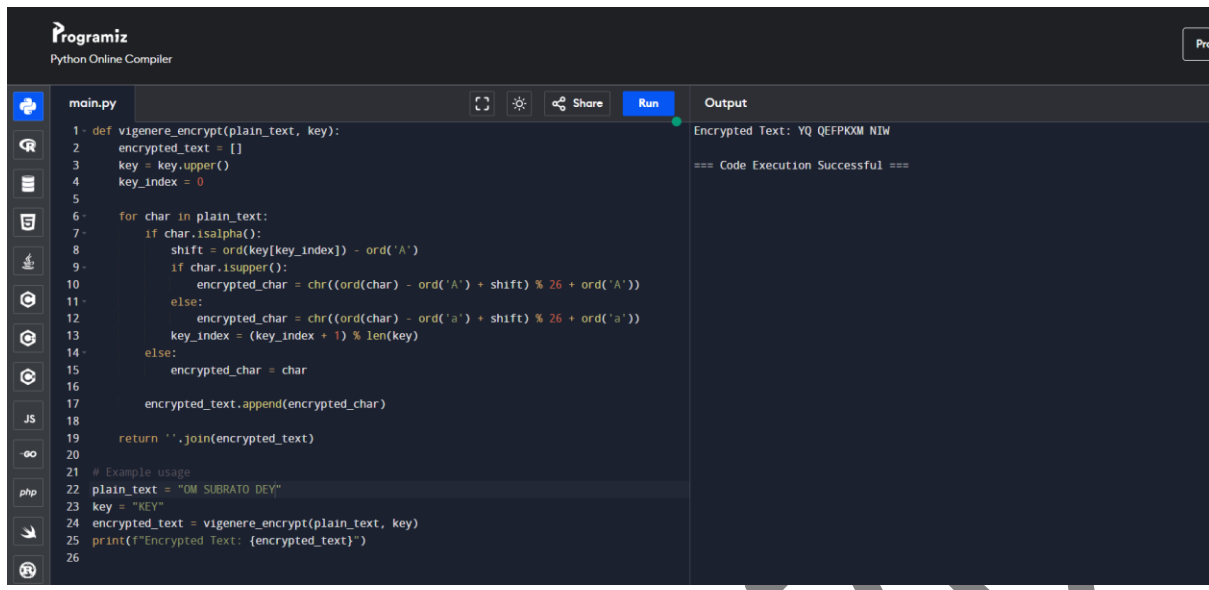
NAME : OM SUBRATO DEY

REGISTER NUMBER : 21BAI1876

1. VIGENERE CIPHER:

ENCRYPTION CODE AND OUTPUT:

```
def vigenere_encrypt(plain_text, key):  
    encrypted_text = []  
    key = key.upper()  
    key_index = 0  
  
    for char in plain_text:  
        if char.isalpha():  
            shift = ord(key[key_index]) - ord('A')  
            if char.isupper():  
                encrypted_char = chr((ord(char) - ord('A') + shift)  
% 26 + ord('A'))  
            else:  
                encrypted_char = chr((ord(char) - ord('a') + shift)  
% 26 + ord('a'))  
            key_index = (key_index + 1) % len(key)  
        else:  
            encrypted_char = char  
        encrypted_text.append(encrypted_char)  
  
    return ''.join(encrypted_text)  
  
# Example usage  
plain_text = "OM SUBRATO DEY"  
key = "KEY"  
encrypted_text = vigenere_encrypt(plain_text, key)  
print(f"Encrypted Text: {encrypted_text}")
```



The screenshot shows the Programiz Python Online Compiler interface. The editor contains a Python script for Vigenere encryption. The code defines a function `vigenere_encrypt` that takes `plain_text` and `key` as arguments. It initializes `encrypted_text` as an empty list, converts the key to uppercase, and iterates through the plain text. For each character, it calculates a shift based on the current key character and applies it to the character (wrapping around the alphabet). The resulting encrypted characters are appended to the `encrypted_text` list. Finally, the list is joined into a string and returned. The example usage at the bottom sets `plain_text = "OM SUBRATO DEY"` and `key = "KEY"`, resulting in the encrypted text `YQ QEFPIOM NIW`. The output pane on the right shows the encrypted text and a success message.

```
1 def vigenere_encrypt(plain_text, key):
2     encrypted_text = []
3     key = key.upper()
4     key_index = 0
5
6     for char in plain_text:
7         if char.isalpha():
8             shift = ord(key[key_index]) - ord('A')
9             if char.isupper():
10                encrypted_char = chr((ord(char) - ord('A') + shift) % 26 + ord('A'))
11            else:
12                encrypted_char = chr((ord(char) - ord('a') + shift) % 26 + ord('a'))
13            key_index = (key_index + 1) % len(key)
14        else:
15            encrypted_char = char
16
17        encrypted_text.append(encrypted_char)
18
19    return ''.join(encrypted_text)
20
21 # Example usage
22 plain_text = "OM SUBRATO DEY"
23 key = "KEY"
24 encrypted_text = vigenere_encrypt(plain_text, key)
25 print(f"Encrypted Text: {encrypted_text}")
26
```

Output: Encrypted Text: YQ QEFPIOM NIW
=== Code Execution Successful ===

DECRYPTION CODE AND OUTPUT:

```
def decrypt_vigenere(cipher_text, key):
    plain_text = []
    key = key.upper()
    key_index = 0

    for char in cipher_text:
        if char.isalpha():
            shift = ord(key[key_index]) - ord('A')
            if char.isupper():
                decrypted_char = chr((ord(char) - ord('A') - shift +
26) % 26 + ord('A'))
            else:
                decrypted_char = chr((ord(char) - ord('a') - shift +
26) % 26 + ord('a'))

            key_index = (key_index + 1) % len(key)
        else:
            decrypted_char = char
```

```

        plain_text.append(decrypted_char)

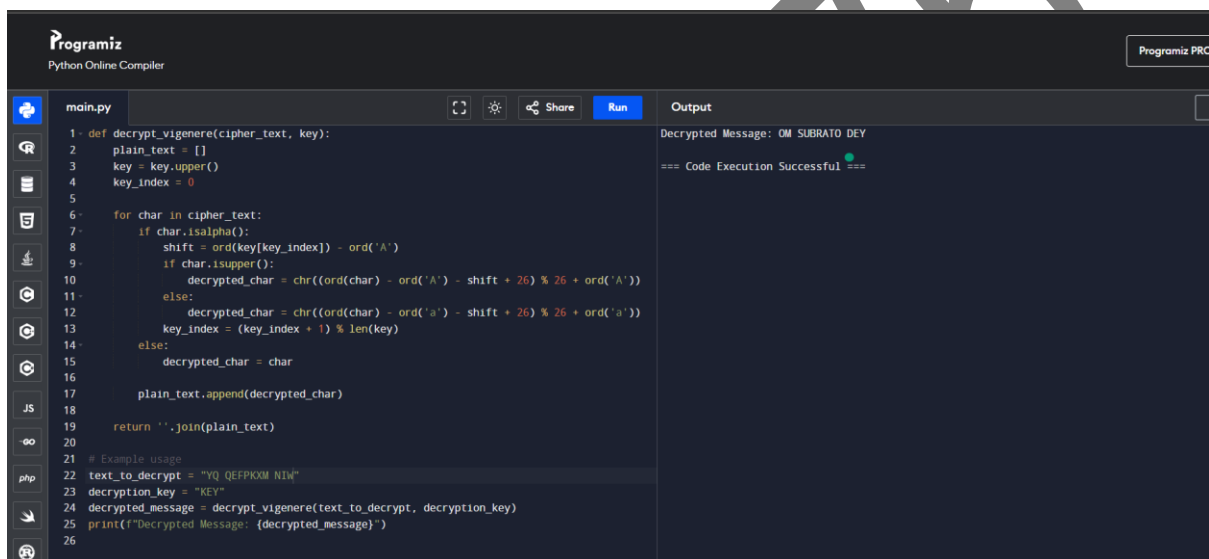
    return ''.join(plain_text)

# Example usage
text_to_decrypt = "YQ QEFPKXM NIW"
decryption_key = "KEY"

decrypted_message = decrypt_vigenere(text_to_decrypt,
decryption_key)

print(f"Decrypted Message: {decrypted_message}")

```



The screenshot shows the Programiz Python Online Compiler interface. The editor contains a Python script for decrypting a Vigenere cipher. The script defines a `decrypt_vigenere` function that takes a cipher text and a key, and returns the decrypted plain text. The example usage section sets `text_to_decrypt = "YQ QEFPKXM NIW"` and `decryption_key = "KEY"`, then calls the function and prints the result. The output pane shows the decrypted message: "OM SUBRATO DEY".

```

main.py
1- def decrypt_vigenere(cipher_text, key):
2-     plain_text = []
3-     key = key.upper()
4-     key_index = 0
5-
6-     for char in cipher_text:
7-         if char.isalpha():
8-             shift = ord(key[key_index]) - ord('A')
9-             if char.isupper():
10-                 decrypted_char = chr((ord(char) - ord('A') - shift + 26) % 26 + ord('A'))
11-             else:
12-                 decrypted_char = chr((ord(char) - ord('a') - shift + 26) % 26 + ord('a'))
13-             key_index = (key_index + 1) % len(key)
14-         else:
15-             decrypted_char = char
16-         plain_text.append(decrypted_char)
17-
18-     return ''.join(plain_text)
19-
20-
21- # Example usage
22- text_to_decrypt = "YQ QEFPKXM NIW"
23- decryption_key = "KEY"
24- decrypted_message = decrypt_vigenere(text_to_decrypt, decryption_key)
25- print(f"Decrypted Message: {decrypted_message}")
26-

```

Output

```

Decrypted Message: OM SUBRATO DEY
=== Code Execution Successful ===

```

2. RAIL FENCE CIPHER:

CODE:

```
def encrypt_text(input_text, key):  
    rails = [[] for _ in range(key)]  
    rail_index = 0  
    direction = 1  
  
    for char in input_text:  
        rails[rail_index].append(char)  
        rail_index += direction  
  
        if rail_index == 0 or rail_index == key - 1:  
            direction *= -1  
  
    cipher = ''  
    for rail in rails:  
        cipher += ''.join(rail)  
  
    return cipher  
  
# Example usage  
plain_text = "OM SUBRATO DEY"  
key = 3  
encrypted_text = encrypt_text(plain_text, key)  
print(f"Encrypted Text: {encrypted_text}")
```

OUTPUT:

The screenshot shows the Programiz Python Online Compiler interface. The editor contains a Python script for a Rail Fence cipher. The script defines a function `encrypt_text` that takes `input_text` and `key` as arguments. It initializes `rails` as a list of empty lists, `rail_index` as 0, and `direction` as 1. It then iterates over each character in `input_text`, appending it to the current rail and updating the `rail_index` and `direction` based on the current rail's position. After processing all characters, it joins the rails into a single string and returns it. An example usage is provided at the bottom, with `plain_text = "OM SUBRATO DEY"` and `key = 3`, resulting in the encrypted text `"OUTEMSBAODY R"`.

```
1 def encrypt_text(input_text, key):
2     rails = [[] for _ in range(key)]
3     rail_index = 0
4     direction = 1
5
6     for char in input_text:
7         rails[rail_index].append(char)
8         rail_index += direction
9
10        if rail_index == 0 or rail_index == key - 1:
11            direction *= -1
12
13    cipher = ''
14    for rail in rails:
15        cipher += ''.join(rail)
16
17    return cipher
18
19 # Example usage
20 plain_text = "OM SUBRATO DEY"
21 key = 3
22 encrypted_text = encrypt_text(plain_text, key)
23 print(f'Encrypted Text: {encrypted_text}')
24
```

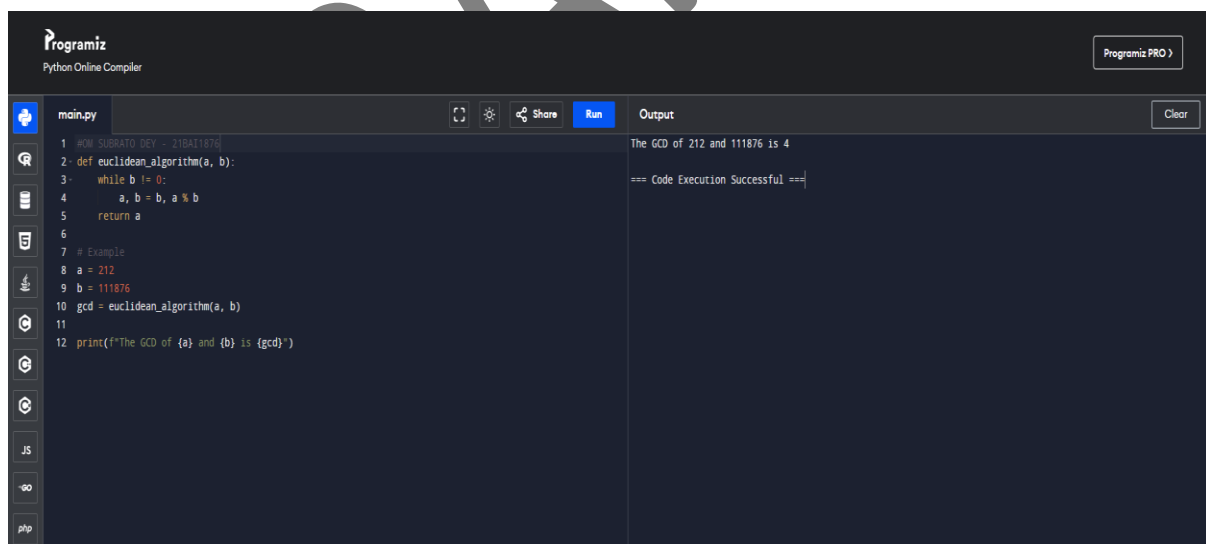
The output panel shows the result: "Encrypted Text: OUTEMSBAODY R" and "=== Code Execution Successful ===".

3. EUCLIDEAN TECHNIQUE:

CODE:

```
def euclidean_algorithm(a, b):  
    while b != 0:  
        a, b = b, a % b  
    return a  
  
# Example  
a = 212  
b = 111876  
gcd = euclidean_algorithm(a, b)  
  
print(f"The GCD of {a} and {b} is {gcd}")
```

OUTPUT:



The screenshot displays the Programiz Python Online Compiler interface. On the left, a sidebar contains icons for file management and a list of languages including Python, JS, Go, and PHP. The main editor area shows a file named 'main.py' with the following code:

```
1 #GCM SUBRATO DEVI - 21BA11876  
2 def euclidean_algorithm(a, b):  
3     while b != 0:  
4         a, b = b, a % b  
5     return a  
6  
7 # Example  
8 a = 212  
9 b = 111876  
10 gcd = euclidean_algorithm(a, b)  
11  
12 print(f"The GCD of {a} and {b} is {gcd}")
```

On the right, the 'Output' tab is active, displaying the result of the code execution:

```
The GCD of 212 and 111876 is 4  
== Code Execution Successful ==
```

The interface also includes a 'Run' button and a 'Clear' button for the output.