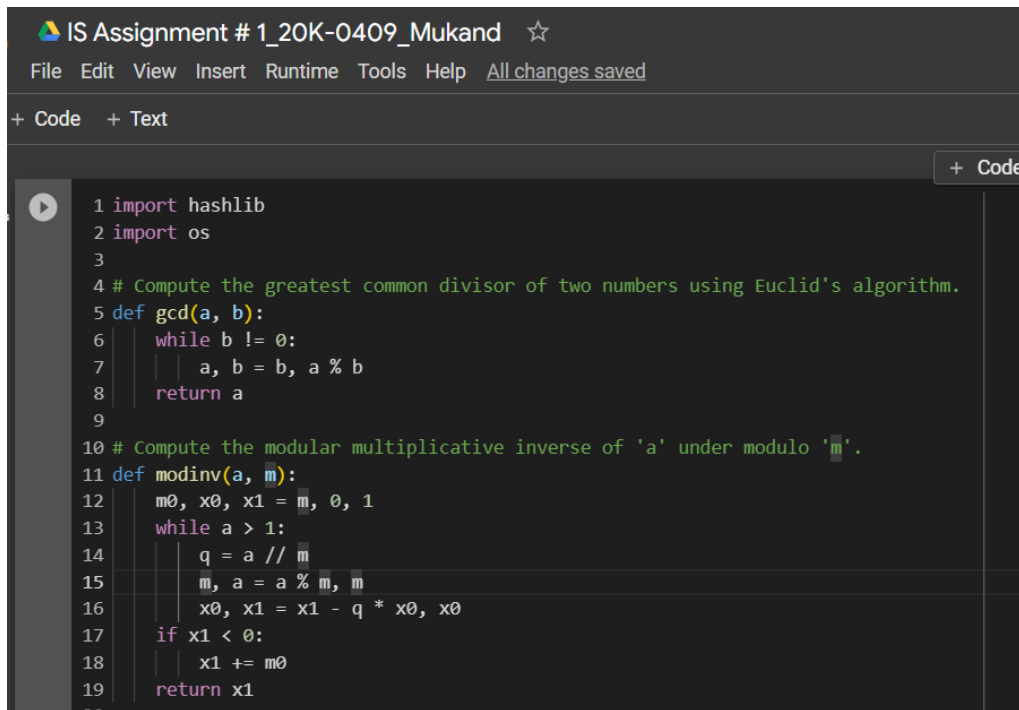

IS Assignment # 1

Roll no: 20K-0409

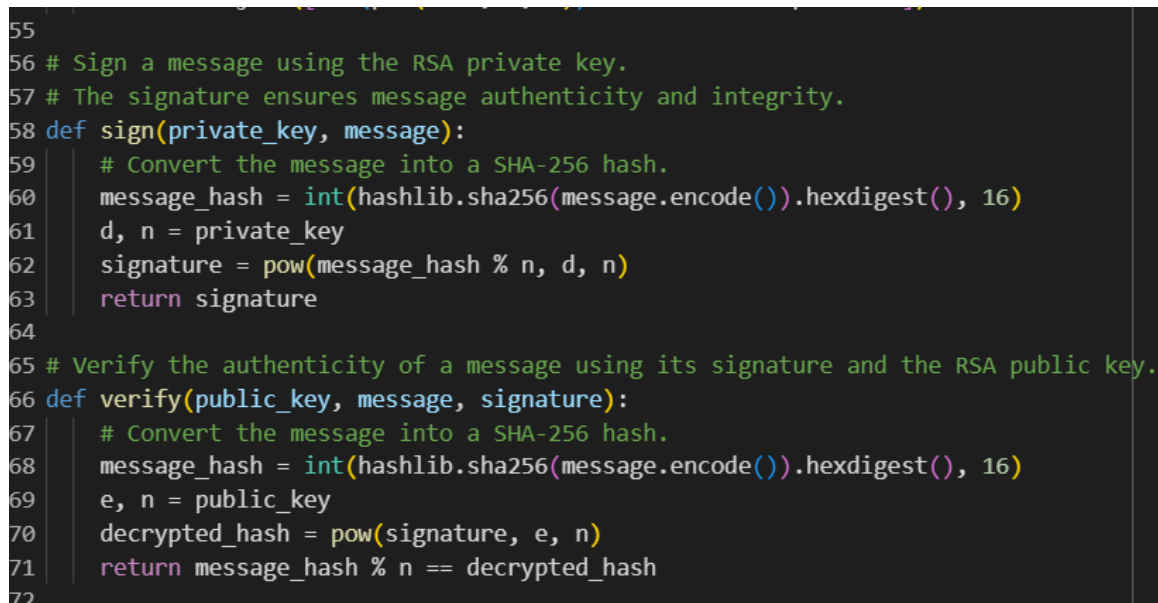
Screen shots:



```
IS Assignment # 1_20K-0409_Mukand ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
+ Code

1 import hashlib
2 import os
3
4 # Compute the greatest common divisor of two numbers using Euclid's algorithm.
5 def gcd(a, b):
6     while b != 0:
7         a, b = b, a % b
8     return a
9
10 # Compute the modular multiplicative inverse of 'a' under modulo 'm'.
11 def modinv(a, m):
12     m0, x0, x1 = m, 0, 1
13     while a > 1:
14         q = a // m
15         m, a = a % m, m
16         x0, x1 = x1 - q * x0, x0
17     if x1 < 0:
18         x1 += m0
19     return x1
20
```

RSA Signature:



```
55
56 # Sign a message using the RSA private key.
57 # The signature ensures message authenticity and integrity.
58 def sign(private_key, message):
59     # Convert the message into a SHA-256 hash.
60     message_hash = int(hashlib.sha256(message.encode()).hexdigest(), 16)
61     d, n = private_key
62     signature = pow(message_hash % n, d, n)
63     return signature
64
65 # Verify the authenticity of a message using its signature and the RSA public key.
66 def verify(public_key, message, signature):
67     # Convert the message into a SHA-256 hash.
68     message_hash = int(hashlib.sha256(message.encode()).hexdigest(), 16)
69     e, n = public_key
70     decrypted_hash = pow(signature, e, n)
71     return message_hash % n == decrypted_hash
72
```

Encrypt and Decrypt:

```
45
46 # Encrypt a plaintext using the RSA public key.
47 def encrypt(pk, plaintext):
48     e, n = pk
49     return [pow(ord(char), e, n) for char in plaintext]
50
51 # Decrypt a ciphertext using the RSA private key.
52 def decrypt(pk, ciphertext):
53     d, n = pk
54     return ''.join([chr(pow(char, d, n)) for char in ciphertext])
55
```

Output:

```
RSA Encryption/Decryption
Enter a prime number (e.g., 17, 19, 23): 109
Enter another prime number (different from the first): 103

Public key: (5, 11227)
Private key: (8813, 11227)

Want to input the message directly or from a .txt file? (m/f): m
Enter a message to encrypt (small data): hi, 102 , # mukand

Encrypted message: 87566933282181563929720376828156282181561969815665401812792510043218110057
Decrypted message: hi, 102 , # mukand

Digital Signature: 3107
The signature is valid!
```

Input from file:

```
108 is_verified = verify(public, message, signature)
109 if is_verified:
110     print("The signature is valid!")
111 else:
112     print("The signature is invalid!")
113
```

```
RSA Encryption/Decryption
Enter a prime number (e.g., 17, 19, 23): 103
Enter another prime number (different from the first): 97

Public key: (5, 9991)
Private key: (3917, 9991)

Want to input the message directly or from a .txt file? (m/f): f
Enter the path to the .txt file: /content/I_S.txt

Encrypted message: 671687562504396465425043964654963016416425820764081004654164216525043968756764058204654219316416539616425047640360546541313896
Decrypted message: This is mukand krishna pursuing CS.

Digital Signature: 3652
The signature is valid!
```