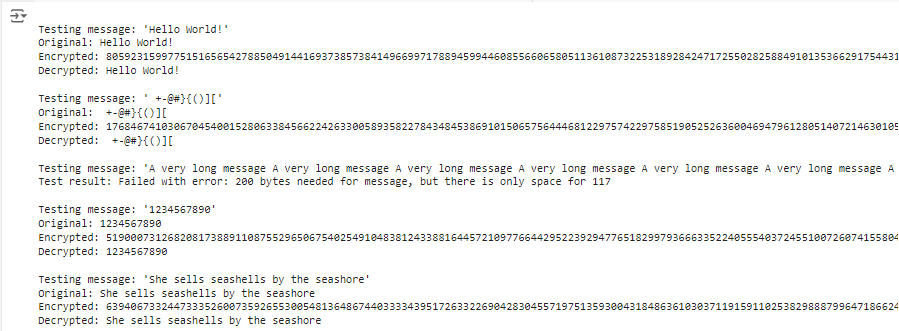
# Question 9:

**Python implementation and result :**





**Explanation of python implementation**

**1. Encryption Function**

The ‘encrypt\_message’ function is responsible for converting a text message into integer encryption, using a public key. This helps ensure that only the owner of the private key can decrypt and read the message

**Operations**

1. **Encoding the Message**: The initial plaintext message is encoded into bytes using the .encode() method, as the RSA algorithm operates on bytes.
2. **Public Key Encryption**: The bytes of the message are then encrypted using the public key. This ensures that the message can only be decrypted by the holder of the corresponding private key.
3. **Converting Bytes to Integer**: The encrypted bytes are converted into a large integer. This transformation is useful for storing or transmitting the encrypted message in a numerical format, facilitating easier handling of the encrypted data.

**2. Decryption Function**

The ‘decrypt\_message’ function is designed to decrypt an encrypted message using a private key, converting the encrypted numeric format back into the original plaintext message. Here’s how it operates:

**Operations**

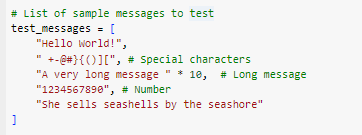
1. **Converting Integer to Bytes**: The encrypted message, now a large integer, is converted back into bytes. This is done using the ‘to\_bytes’ method, calculating the necessary byte length from the bit length of the encrypted integer.
2. **Private Key Decryption**: The bytes are decrypted using the private key. Only the private key has the ability to decrypt these bytes, ensuring that only the intended recipient can read the message.
3. **Decoding to Text**: After decryption, the bytes are decoded back into a string using the ‘.decode()’ method, allowing the recipient to read the original message.

**3.Testing message:**

This function tests the entire encryption and decryption process:

* Each message is encrypted and then decrypted.
* The results are printed and checked to ensure the decrypted message matches the original.

We use a series of test messages, including special characters, numbers, and a long repeating string, used to verify the robustness of the encryption and decryption functions.



**Conclusion**

With the above example the basic application of the RSA encryption mechanism in Python can be adapted for secure communication systems with more complex decryption. The ability to efficiently encrypt and decrypt messages using RSA highlights its role in ensuring the security of encrypted messages.