Post-Lab:

1. Real-Time Use of Cryptographic Systems

Public-key Cryptography (Asymmetric Cryptography):

- **Use Case: Secure Email (e.g., PGP):** Public-key cryptography is used in secure email systems like PGP (Pretty Good Privacy), where a user's public key encrypts the message, and only the corresponding private key can decrypt it.
- **Example: SSL/TLS:** In web security (HTTPS), asymmetric cryptography is used during the initial handshake between a client and server. The server's public key encrypts a session key, which is then used for symmetric encryption during the session.

Symmetric Key Cryptography (Private Key Cryptography):

- **Use Case: Data Encryption (e.g., AES):** Symmetric key cryptography is widely used for encrypting large amounts of data due to its efficiency. AES (Advanced Encryption Standard) is a common symmetric encryption algorithm.
- **Example: File Encryption:** When encrypting files on a disk, symmetric algorithms are preferred due to their speed and lower computational cost compared to asymmetric algorithms.

2. Testing the Algorithms with Different Inputs

```
# Affine Cipher
print("Affine Cipher Tests")
affine_tests = [
    ("HELLO", 5, 8),
    ("CRYPTOGRAPHY", 7, 3),
    ("AFFINECIPHER", 11, 4),
]
for text, a, b in affine_tests:
    cipher = encrypt_affine(text, a, b)
    decrypted = decrypt_affine(cipher, a, b)
    print(f"Plaintext: {text} | Ciphertext: {cipher} | Decrypted: {decrypted}")


# Caesar Cipher
print("\nCaesar Cipher Tests")
```

```
caesar_tests = [
    ("HELLO", 3),
    ("SECURITY", 5),
    ("CAESARCIPHER", 7),
]
for text, shift in caesar_tests:
    cipher = encrypt_caesar(text, shift)
    decrypted = decrypt_caesar(cipher, shift)
    print(f"Plaintext: {text} | Ciphertext: {cipher} | Decrypted: {decrypted}")

# Miller-Rabin Primality Test
print("\nMiller-Rabin Primality Test")
miller_rabin_tests = [
    (31, 4),  # prime
    (91, 4),  # composite
    (97, 4),  # prime
]
for n, k in miller_rabin_tests:
    result = miller_rabin_test(n, k)
    print(f"Is {n} prime? {result}")

# Euclidean Algorithm for GCD
print("\nEuclidean Algorithm GCD Tests")
gcd_tests = [
    (56, 98),
    (1071, 462),
    (48, 18),
]
for a, b in gcd_tests:
    result = gcd(a, b)
    print(f"GCD of {a} and {b} is {result}")
```

```
# Rabin Cryptosystem
print("\nRabin Cryptosystem Tests")
rabin_tests = [
    (5, 7, 11),  # message, p, q
    (13, 11, 19),
    (20, 17, 23),
]
for m, p, q in rabin_tests:
    n = p * q
    cipher = rabin_encrypt(m, n)
    decrypted = rabin_decrypt(cipher, p, q)
    print(f"Message: {m} | Ciphertext: {cipher} | Decrypted possibilities: {decrypted}")
```