

Practical No. 1

- A. Develop a secure messaging application where users can exchange messages securely using RSA encryption. Implement a mechanism for generating RSA key pairs and encrypting/decrypting messages.**

Code:

RSA key generation. Or The one given below:

```
#pip install cryptography

from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend

def generate_rsa_key_pair():
    """Generates a new RSA public and private key pair."""
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )
    public_key = private_key.public_key()
    return private_key, public_key

def encrypt_message(public_key, message):
    """Encrypts a message using the recipient's public key."""
    ciphertext = public_key.encrypt(
        message.encode('utf-8'),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return ciphertext

def decrypt_message(private_key, ciphertext):
    """Decrypts a ciphertext using the recipient's private key."""
    plaintext = private_key.decrypt(
```

```

ciphertext,
padding.OAEP(
mgf=padding.MGF1(algorithm=hashes.SHA256()),
algorithm=hashes.SHA256(),
label=None
)
)
return plaintext.decode('utf-8')

if __name__ == "__main__":
# User 1 generates their keys
print("User 1: Generating RSA key pair...")
user1_private_key, user1_public_key = generate_rsa_key_pair()
print("User 1: Key pair generated.")

# User 2 generates their keys
print("\nUser 2: Generating RSA key pair...")
user2_private_key, user2_public_key = generate_rsa_key_pair()
print("User 2: Key pair generated.")

# User 1 sends a message to User 2
original_message_user1 = "Hello User 2, this is a secret message from User 1!"
print(f"\nUser 1: Original message to User 2: '{original_message_user1}'")

# User 1 encrypts the message using User 2's public key
encrypted_message_user1_to_user2 = encrypt_message(user2_public_key, original_message_user1)
print(f"User 1: Encrypted message (ciphertext): {encrypted_message_user1_to_user2}")

# User 2 receives and decrypts the message using their private key
decrypted_message_user2 = decrypt_message(user2_private_key,
encrypted_message_user1_to_user2)
print(f"User 2: Decrypted message: '{decrypted_message_user2}'")

# User 2 sends a reply to User 1

```

```

"IDLE Shell 3.12.2"
File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abdd59, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

= RESTART: F:\MSCIT-BlockChain\BC_Practical_1.py
User 1: Generating RSA key pair...
User 1: Key pair generated.

User 2: Generating RSA key pair...
User 2: Key pair generated.

User 1: Original message to User 2: 'Hello User 2, this is a secret message from User 1!'
User 1: Encrypted message (ciphertext): b"2\x0e9\x03\x59\xfa[\x0c8\x54\x90P\x02\xcdj\x0e\xee\xdd\x85\x4b5\xde\xab\x0f\x12\x24\xebt\x0b0\x08\x03X\x0d9\x07a\x08\xcaai\xed\x0e\xad\x0c8\x0b\x018\x06\x1f\xbb\x0b0\x0e\x0c0c\x08\x0c\x5\x019\x0ea8\x09\x03\x88\x04\x0d\x83\x8b7\xad\x0903E\xaa;8\x03\x0a\x1a\x06\x18a9\xfb;\x0e\x0b\x0827\xce\x083\x0d\x0c3\x057e+ a\x9f\x0f\x0f\x0f\x0f\x0d\x02\x82\x0b8R\x0c1ng0\x0dJ\x0f\x0d\x04\x09\x1\x0d1\x0c\x0f\x03\x0d0Y\x08\x0f\x06\x0c\x06\x08\x08\x03\x0d\x07\x0d\x06\x0ab\x0f5;[5]\x0f\x0c\x0e5\x9c0a\x15I\x0da_2\x1f\x08C\x0ab\x08e\x04\x0c\x01c\xf3\x06\x0b4\x018X\x0c\x09e\x0ad\x0f2\x0e8\x0d7\x0d5\x06\x0e9*\x82\x0fa\x0d\x06\x0e6\x05m\x07\x01\x090\x149\xfb\x0b7\x02\x1d.\x05\x00\x0d\x2j\x06\x0c7\x0f5\x0e7I\x0e1\x06h7\x0c\x01e\x0a0/\x0c4ub9\x0b3\x04g\x0e\x0cd\x0b2F\x0c1\x09\x02-\x1a/\x0f\x0f\x081\x02\x083\x091\x0dc\x006\x0a7"
User 2: Decrypted message: 'Hello User 2, this is a secret message from User 1!'

User 2: Original message to User 1: 'Hi User 1, I received your message securely!'
User 2: Encrypted reply (ciphertext): b'.\x0f\x0b0\x08\x0d\x0c\x9a\r\x0b6\x11kr8\x0af5V\x0ad7=\x96\x96\x05\x0f\x0f90\xfb\x0f\x0f\x0f4!\x0dbv\x0dV-\x0dd\x0f\x095\x02\x12\x0f\x02\x0a5\x0b3\r\x0c9\x0d7\x0ea\x0b\x0f42\x0f19\x0c4\x099\x0db\x0d5[.\x08\x0d\x082X\x080-\x0f6Y\x00\x0b2\x0c76I\x08Nw\x080\x0a3\x0ab\x0e5\\\x0f3\x0f\x084\x0f6\x0c2\x093h4\x0b4u\x0f2h\x0dc-r\x01\x0e5-\x097\x0d4\x08a\x0d2\x0d0\x0b4\x0f0\x0d7\x03\x04\x08btFX\x02\x0e7\x08\x05\x0e1\x0d5\x0b6F\x0a6\x0a8\x093\x0b5\x0ab-\x0db\x09c\x0a3\x0d5\x0b5\x0d5T\x086\x0b0*\x0da\x0a6\x0c\x07\x0d37\x0a8\x0b6:\x12j2\x0f3p8\x0fR\x0b5\x0f5v\x09\x0fa:\x04f\x0eb\x0b9N/B\x093\x0f\x0c7\x0c2\x0c80\x1abV\x0ca\x0bF\x0e3\x0c4\x08f\x0dbaf7\x0d4\x0177\x0bb\x092\x060c\x0c0\x0f2_0\x0f30\x0ab\x01\x0b0\x0d\x080\x0a6\x0a4\x08e\x0f3\x083\x0db\x0f\x0cb\x0949\r\x06\x0a0Lz\x0b\x0f1\x0af\x0bc\x0a3,\x0e70\x0dc\x086\x0e\x07\x0ee\x0a2\x0f\x0ab\x0d9\x0f1:\x0b4'
User 1: Decrypted reply: 'Hi User 1, I received your message securely!'

>>> sni

```

B. Allow users to create multiple transactions and display them in an organised format.

Code

```
import Crypto
import binascii
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA

class Client:
    def __init__(self):
        # Creating random number for key
        random = Crypto.Random.new().read
        # Creating new public key and private key
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

        @property
        def identity(self):
            return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

class Transaction:
    def __init__(self, sender, receiver, value):
        self.sender = sender
        self.receiver = receiver
        self.value = value
        self.time = datetime.datetime.now()

    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
        return collections.OrderedDict({
```

```

'sender': identity,

'receiver': self.receiver,

'value': self.value,

'time': self.time

})

def sign_transaction(self):

private_key = self.sender._private_key

signer = PKCS1_v1_5.new(private_key)

h = SHA.new(str(self.to_dict()).encode('utf8'))

return binascii.hexlify(signer.sign(h)).decode('ascii')

Raj = Client()

print("-"*50)

print("Raj Key")

print(Raj.identity)

Vai = Client()

print("-"*50)

print("Vai Key")

print(Vai.identity)

t = Transaction(Raj, Vai.identity, 10.0)

print("-"*50)

print("Transaction Sign")

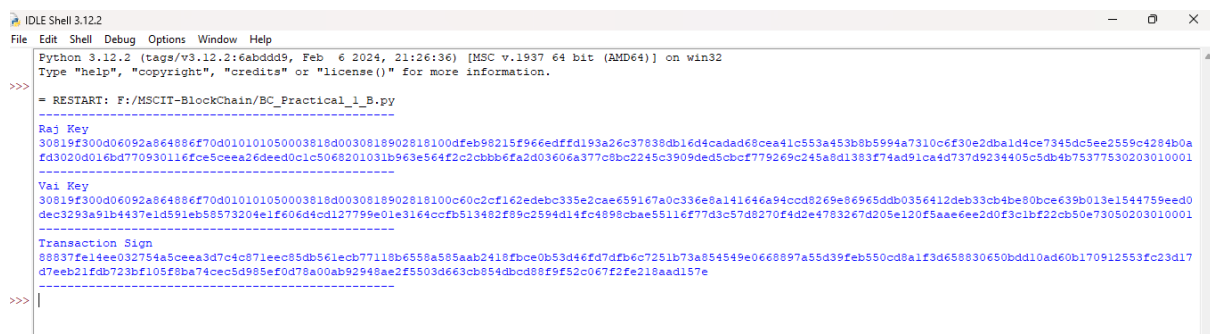
signature = t.sign_transaction()

print(signature)

print("-"*50)

```

Output:



```

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\MSCIT-BlockChain\BC_Practical_1_B.py
-----
Raj Key
30819f300d06092a864886f70d0101050003818d0030818902818100dfef98215f96eedffdf93a26c37838db16d4cadad68cea41c553a453b8b5994a7310c6f30e2dba1d4ce7345dc5ee2559c4284b0a
fd3020d016bd770930116fce5ceea26deed0c1c5068201031b963e564f2c2cbbb6fa2d03606a377c8bc2245c3909ded5cbcf779269c245a8d1383f74ad91ca4d737d9234405c5db4b75377530203010001
-----
Vai Key
30819f300d06092a864886f70d0101050003818d0030818902818100c60c2cf162edebc335e2cae659167a0c336e8a141646a94ccdd8269e86965ddb0356412deb33cb4be80bce639b013e1544759eed0
dec3293a91b4437e1d591eb58573204e1f606d4cd127799e01e3164ccfb513482f89c2594d14fc4898cbacae55116f77d3c57d8270f4d2e4783267d205e120f5aae6ee2d0f3c1bf22cb50e73050203010001
-----
Transaction Sign
88837fe14ee032754a5ceea3d7c4c071e0c85db561ecb77118b6558a585aab2410fbce0b53d46fd7dfb6c7251b73a854549e0668897a55d39feb550cd8a1f3d658830650bdd10ad60b170912553fc23d17
d7eeb21fdb723bf105f8ba74cec5d985ef0d78a00ab92946ae2f5503d663cb854dbcd88f9f52c067f2fe218aad157e
>>>

```

C. Create a Python class named Transaction with attributes for sender, receiver, and amount. Implement a method within the class to transfer money from the sender's account to the receiver's account.

Code:

```
#!/pip install pycryptodome

import Crypto
import binascii

from Crypto.PublicKey import RSA
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
import datetime
import collections
import hashlib

from hashlib import sha256

class Client:
    def __init__(self):
        # Creating random number for key
        random = Crypto.Random.new().read

        # Creating new public key and private key
        self._private_key = RSA.generate(1024, random)

        self._public_key = self._private_key.publickey()

        self._signer = PKCS1_v1_5.new(self._private_key)

        @property
        def identity(self):
            return binascii.hexlify(self._public_key.exportKey(format="DER")).decode(
                "ascii"
            )

class Transaction:
    def __init__(self, sender, receiver, value):
        self.sender = sender

        self.receiver = receiver

        self.value = value

        self.time = datetime.datetime.now()
```

```

def to_dict(self):
    if self.sender == "Genesis":
        identity = "Genesis"
    else:
        identity = self.sender.identity
    return collections.OrderedDict(
        {
            "sender": identity,
            "receiver": self.receiver,
            "value": self.value,
            "time": self.time,
        }
    )

def sign_transaction(self):
    private_key = self.sender._private_key
    signer = PKCS1_v1_5.new(private_key)
    h = SHA.new(str(self.to_dict()).encode("utf8"))
    return binascii.hexlify(signer.sign(h)).decode("ascii")

def sha256(message):
    return hashlib.sha256(message.encode("ascii")).hexdigest

def mine(message, difficulty=1):
    assert difficulty >= 1
    prefix = "1" * difficulty
    for i in range(1000):
        digest = sha256(str(hash(message)) + str(i))
        if digest.startswith(prefix):
            print("after" + str(i) + "iteration found nonce:" + digest)
    return digest

class Block:
    def __init__(self):
        self.verified_transactions = []
        self.previous_block_hash = ""

```

```

self.Nonce = ""

last_block_hash = ""

def display_transaction(transaction):
    dict = transaction.to_dict()
    print("Sender: " + dict["sender"])
    print("-----")
    print("Receiver: " + dict["receiver"])
    print("-----")
    print("Value: " + str(dict["value"]))
    print("-----")
    print("Time: " + str(dict["time"]))
    print("-----")

TPCoins = []

def dump_blockchain(self):
    print("Number of blocks in chain" + str(len(self)))
    for x in range(len(Block.TPCoins)):
        block_temp = Block.TPCoins[x]
        print("block #" + str(x))
        for transaction in block_temp.verified_transactions:
            Block.display_transaction(transaction)
            print("-----")
        last_transaction_index = 0
        transactions = []
        Ninad = Client()
        ks = Client()
        vighnesh = Client()
        sairaj = Client()

        t1 = Transaction(Ninad, ks.identity, 15.0)
        t1.sign_transaction()
        transactions.append(t1)

```



```
t2 = Transaction(Ninad, vighnesh.identity, 6.0)
```

```
t2.sign_transaction()
```

```
transactions.append(t2)
```

```
t3 = Transaction(Ninad, sairaj.identity, 16.0)
```

```
t3.sign_transaction()
```

```
transactions.append(t3)
```

```
t4 = Transaction(vighnesh, Ninad.identity, 8.0)
```

```
t4.sign_transaction()
```

```
transactions.append(t4)
```

```
t5 = Transaction(vighnesh, ks.identity, 19.0)
```

```
t5.sign_transaction()
```

```
transactions.append(t5)
```

```
t6 = Transaction(vighnesh, sairaj.identity, 35.0)
```

```
t6.sign_transaction()
```

```
transactions.append(t6)
```

```
t7 = Transaction(sairaj, vighnesh.identity, 5.0)
```

```
t7.sign_transaction()
```

```
transactions.append(t7)
```

```
t8 = Transaction(sairaj, Ninad.identity, 12.0)
```

```
t8.sign_transaction()
```

```
transactions.append(t8)
```

```
t9 = Transaction(sairaj, ks.identity, 25.0)
```

```
t9.sign_transaction()
```

```
transactions.append(t9)
```

t10 = Transaction(Ninad, ks.identity, 1.0)

t10.sign_transaction()

transactions.append(t10)

for transaction in transactions:

display_transaction(transaction)

print("*" * 50)

Output:

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=====
Restart: F:\MSCIT-Blockchain\BC_Practical_1_C.py
Sender: 30819f300d06092a8e4886f70d0101050003818d0030818902818100bba38c1bea7ccf3d52e90cdfdc7b419c909c4b8049cccd5935679c370f3e648efba69440e29c6fb7f715fab38fa71171ad462760b015009d4c074870f32743db5ddd8a18b663fddff5d1ecc8edf1808a75cadd74a92d65b7ccf265d75932e2957c3c96e71de206b360b817d5baf4337abc25f14643ea302c2ba086a037e369d0203010001
-----
Receiver: 30819f300d06092a8e4886f70d0101050003818d0030818902818100c01fb3f1dea5819595ec757a91a20d65511059645c886d9c03d3c7a7f05fed107d0e080d7e02104407b1f98ebd0eb40f2c2e44d48a62b6de2e1308933b41ef48c379fffb50be508cb5a6d3933cee64b120d2630e2ec4851ff76fe5ab51fd22862dab623788953557bab02fbaa5147e10d30409e558b1027ccf38cd0abe672d0203010001
-----
Value: 15.0
-----
Time: 2025-06-26 09:56:13.019031
-----
=====
Sender: 30819f300d06092a8e4886f70d0101050003818d0030818902818100bba38c1bea7ccf3d52e90cdfdc7b419c909c4b8049cccd5935679c370f3e648efba69440e29c6fb7f715fab38fa71171ad462760b015009d4c074870f32743db5ddd8a18b663fddff5d1ecc8edf1808a75cadd74a92d65b7ccf265d75932e2957c3c96e71de206b360b817d5baf4337abc25f14643ea302c2ba086a037e369d0203010001
-----
Receiver: 30819f300d06092a8e4886f70d0101050003818d0030818902818100e590c5f5806153e235f22e63785e16afbf565c5bc580a45529a781a7e06b31d503a0baff407bc9ffef56dc3dfc7c1f42bcf5a22679bf8b75b75ff48a6a8440790ca107c0def06c72e697e64828fa7fb73f326a4ff59278c387c8912aad2dfcc3d3715a449b6066df10c0e7be37dbfacc696969c517ba881e041094f1b89c690170203010001
-----
Value: 6.0
-----
Time: 2025-06-26 09:56:13.019031
-----
=====
Sender: 30819f300d06092a8e4886f70d0101050003818d0030818902818100bba38c1bea7ccf3d52e90cdfdc7b419c909c4b8049cccd5935679c370f3e648efba69440e29c6fb7f715fab38fa71171ad462760b015009d4c074870f32743db5ddd8a18b663fddff5d1ecc8edf1808a75cadd74a92d65b7ccf265d75932e2957c3c96e71de206b360b817d5baf4337abc25f14643ea302c2ba086a037e369d0203010001
-----
Receiver: 30819f300d06092a8e4886f70d0101050003818d0030818902818100a57e041c8257fc98d9a7a90b8f057515a8f23b2b0fe14cbfcae12ae520fea0aaadc5a32b68cea8208416db2857185b52030c45ba49269be5bfa630d16523c4c41313dd92b72f2aa08b3b7371188110920ec6127aa497e0c138e381c356fb96cd342fd5b650d8452efe0cd2da0415418b0926db251b950bd9bd257050c04076d2d0203010001
-----
Value: 16.0
-----
Time: 2025-06-26 09:56:13.019031
-----
=====
Sender: 30819f300d06092a8e4886f70d0101050003818d0030818902818100a57e041c8257fc98d9a7a90b8f057515a8f23b2b0fe14cbfcae12ae520fea0aaadc5a32b68cea8208416db2857185b52030c45ba49269be5bfa630d16523c4c41313dd92b72f2aa08b3b7371188110920ec6127aa497e0c138e381c356fb96cd342fd5b650d8452efe0cd2da0415418b0926db251b950bd9bd257050c04076d2d0203010001
-----
Receiver: 30819f300d06092a8e4886f70d0101050003818d0030818902818100bba38c1bea7ccf3d52e90cdfdc7b419c909c4b8049cccd5935679c370f3e648efba69440e29c6fb7f715fab38fa71171ad462760b015009d4c074870f32743db5ddd8a18b663fddff5d1ecc8edf1808a75cadd74a92d65b7ccf265d75932e2957c3c96e71de206b360b817d5baf4337abc25f14643ea302c2ba086a037e369d0203010001
-----
Value: 12.0
-----
Time: 2025-06-26 09:56:13.034661
-----
=====
Sender: 30819f300d06092a8e4886f70d0101050003818d0030818902818100a57e041c8257fc98d9a7a90b8f057515a8f23b2b0fe14cbfcae12ae520fea0aaadc5a32b68cea8208416db2857185b52030c45ba49269be5bfa630d16523c4c41313dd92b72f2aa08b3b7371188110920ec6127aa497e0c138e381c356fb96cd342fd5b650d8452efe0cd2da0415418b0926db251b950bd9bd257050c04076d2d0203010001
-----
Receiver: 30819f300d06092a8e4886f70d0101050003818d0030818902818100c01fb3f1dea5819595ec757a91a20d65511059645c886d9c03d3c7a7f05fed107d0e080d7e02104407b1f98ebd0eb40f2c2e44d48a62b6de2e1308933b41ef48c379fffb50be508cb5a6d3933cee64b120d2630e2ec4851ff76fe5ab51fd22862dab623788953557bab02fbaa5147e10d30409e558b1027ccf38cd0abe672d0203010001
-----
Value: 25.0
-----
Time: 2025-06-26 09:56:13.034661
-----
=====
Sender: 30819f300d06092a8e4886f70d0101050003818d0030818902818100bba38c1bea7ccf3d52e90cdfdc7b419c909c4b8049cccd5935679c370f3e648efba69440e29c6fb7f715fab38fa71171ad462760b015009d4c074870f32743db5ddd8a18b663fddff5d1ecc8edf1808a75cadd74a92d65b7ccf265d75932e2957c3c96e71de206b360b817d5baf4337abc25f14643ea302c2ba086a037e369d0203010001
-----
Receiver: 30819f300d06092a8e4886f70d0101050003818d0030818902818100c01fb3f1dea5819595ec757a91a20d65511059645c886d9c03d3c7a7f05fed107d0e080d7e02104407b1f98ebd0eb40f2c2e44d48a62b6de2e1308933b41ef48c379fffb50be508cb5a6d3933cee64b120d2630e2ec4851ff76fe5ab51fd22862dab623788953557bab02fbaa5147e10d30409e558b1027ccf38cd0abe672d0203010001
-----
Value: 1.0
-----
Time: 2025-06-26 09:56:13.034661
=====
```

D. Implement a function to add new blocks to the miner and dump the blockchain

Code:

```
import datetime

import hashlib

# Create a class with two functions

class Block:

    def __init__(self, data, previous_hash):

        self.timestamp = datetime.datetime.now(datetime.timezone.utc)

        self.data = data

        self.previous_hash = previous_hash

        self.hash = self.calc_hash()

    def calc_hash(self):

        sha = hashlib.sha256()

        hash_str = self.data.encode("utf-8")

        sha.update(hash_str)

        return sha.hexdigest()

if __name__ == "__main__":

    # Instantiate the class

    blockchain = [Block("First block", "0")]

    blockchain.append(Block("Second block", blockchain[0].hash))

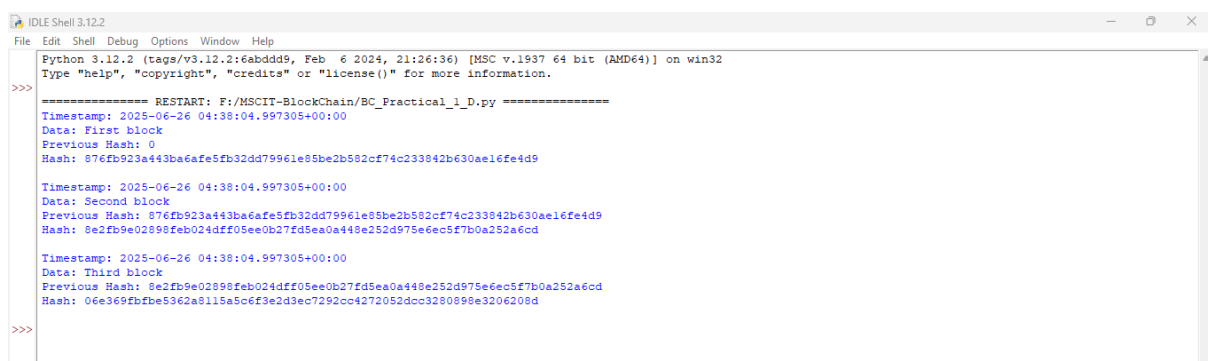
    blockchain.append(Block("Third block", blockchain[1].hash))

    # Dumping the blockchain

    for block in blockchain:

        print(f"Timestamp: {block.timestamp}\nData: {block.data}\nPrevious Hash: {block.previous_hash}\nHash: {block.hash}\n")
```

Output:



```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/MSCIT-Blockchain/BC_Practical_1_D.py =====
Timestamp: 2025-06-26 04:38:04.997305+00:00
Data: First block
Previous Hash: 0
Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9

Timestamp: 2025-06-26 04:38:04.997305+00:00
Data: Second block
Previous Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9
Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd

Timestamp: 2025-06-26 04:38:04.997305+00:00
Data: Third block
Previous Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd
Hash: 06e369fbf5e5362a8115a5c6f3e2d3ec7292cc4272052d0c3280898e3206208d

>>>
```

Practical No. 2

A. Write a python program to demonstrate mining

Code:

```
//npm install web3
const {Web3} = require('web3');
const web3=new Web3(new Web3.providers.HttpProvider('http://127.0.0.1:7545'));
async function mine(){
const accounts=await web3.eth.getAccounts();
const coinbaseacc1=accounts[0];
const coinbaseacc2=accounts[1];
console.log('Mining etheron Ganache with coinbase address:${coinbaseacc1}');
while(true)
{ try{
await web3.eth.sendTransaction({
from:coinbaseacc1, to:coinbaseacc2,
value:50,
});
console.log('Hii Shiva Mined a new block!');
}catch(err){ console.error(err);
} }
} mine();
```

Output:

[illegible]

B. Demonstrate the use of the Bitcoin Core API to interact with a Bitcoin Core node.

Bitcoin Core Api

```
# pip install requests

import requests

# Task 1: Get information regarding the current block

def get_current_block_info():

    response = requests.get("https://blockchain.info/latestblock")

    block_info = response.json()

    print("Current block information:")

    print("Block height:", block_info['height'])

    print("Block hash:", block_info['hash'])

    print("Block index:", block_info['block_index'])

    print("Timestamp:", block_info['time'])


# Task 3: Get balance of an address

def get_address_balance(address):

    response = requests.get(f"https://blockchain.info/q/addressbalance/{address}")

    balance = float(response.text) / 10**8

    print("Balance of address", address, ":", balance, "BTC")


# Example usage

if __name__ == "__main__":

    # Task 1: Get information regarding the current block

    get_current_block_info()


    # Task 3: Get balance of an address

    address = "3Dh2ft6UsqjbTNzs5zrp7uK17Gqg1Pg5u5"

    get_address_balance(address)
```

Output:

```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rajdeep> pip install requests
Defaulting to user installation because normal site-packages is not writeable
Collecting requests
  Downloading requests-2.32.4-py3-none-any.whl.metadata (4.9 kB)
Collecting charset_normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.2-cp312-cp312-win_amd64.whl.metadata (36 kB)
Collecting idna<4,>=2.5 (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.5.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2025.6.15-py3-none-any.whl.metadata (2.4 kB)
Downloading requests-2.32.4-py3-none-any.whl (64 kB)
 64.8/64.8 kB ? eta 0:00:00
Downloading certifi-2025.6.15-py3-none-any.whl (157 kB)
 157.7/157.7 kB 9.8 MB/s eta 0:00:00
Downloading charset_normalizer-3.4.2-cp312-cp312-win_amd64.whl (105 kB)
 105.8/105.8 kB 6.0 MB/s eta 0:00:00
Downloading idna-3.10-py3-none-any.whl (70 kB)
 70.4/70.4 kB 3.8 MB/s eta 0:00:00
Downloading urllib3-2.5.0-py3-none-any.whl (129 kB)
 129.8/129.8 kB ? eta 0:00:00
Installing collected packages: urllib3, idna, charset_normalizer, certifi, requests
WARNING: The script normalizer.exe is installed in 'C:\Users\Rajdeep\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed certifi-2025.6.15 charset_normalizer-3.4.2 idna-3.10 requests-2.32.4 urllib3-2.5.0
```

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: F:\MSCIT-Blockchain\BC_Fract_9.py =====
Current block information:
Block height: 902744
Block hash: 000000000000000000000000aecdcaed79d656ad2302a316a9a9f67cda1896c8a2b5
Block index: 902744
Timestamp: 1750908980
Balance of address 3Dh2ft6UsqjbTNzs5zrp7uKl7GqglPg5u5 : 0.0 BTC
>>>
```

C. Demonstrating the process of running a blockchain node on your local machine.

Make Sure you have Installed node.js in their System.

Code:

```
// npm install crypto-js

const SHA256=require("crypto-js/sha256");

class Block{

constructor(index,timestamp,data,previousHash=""){ this.index=index;

this.timestamp=timestamp; this.data=data; this.previousHash=previousHash;
this.hash=this.calculateHash();

}

calculateHash(){ return SHA256(

this.index+ this.previousHash+ this.timestamp+ JSON.stringify(this.data)

).toString();

} }

class Blockchain{

constructor(){ this.chain=[this.createGenesisBlock()];

} createGenesisBlock(){

return new Block(0,"09/06/2024","GenesisBlock","0");

} getLatestBlock(){

return this.chain[this.chain.length-1]; }

addBlock(newBlock){ newBlock.previousHash=this.getLatestBlock().hash;

newBlock.hash=newBlock.calculateHash(); this.chain.push(newBlock);

} isChainValid(){

for(let i=1;i<this.chain.length;i++){ constcurrentBlock = this.chain[i];

constpreviousBlock = this.chain[i-1];

if(currentBlock.hash != currentBlock.calculateHash()){ returnfalse;

}

if(currentBlock.previousHash != previousBlock.hash){ return false;

}

}

return true;

} }

//BlockchainImplementation
```

```
let myCoin=new Blockchain();  
myCoin.addBlock(new Block(1,"09/06/2024",{amount:4}));  
myCoin.addBlock(new Block(2,"09/06/2024",{amount:8}));  
// console.log('Isblockchainvalid?'+myCoin.isChainValid());  
console.log(JSON.stringify(myCoin,null,4))
```

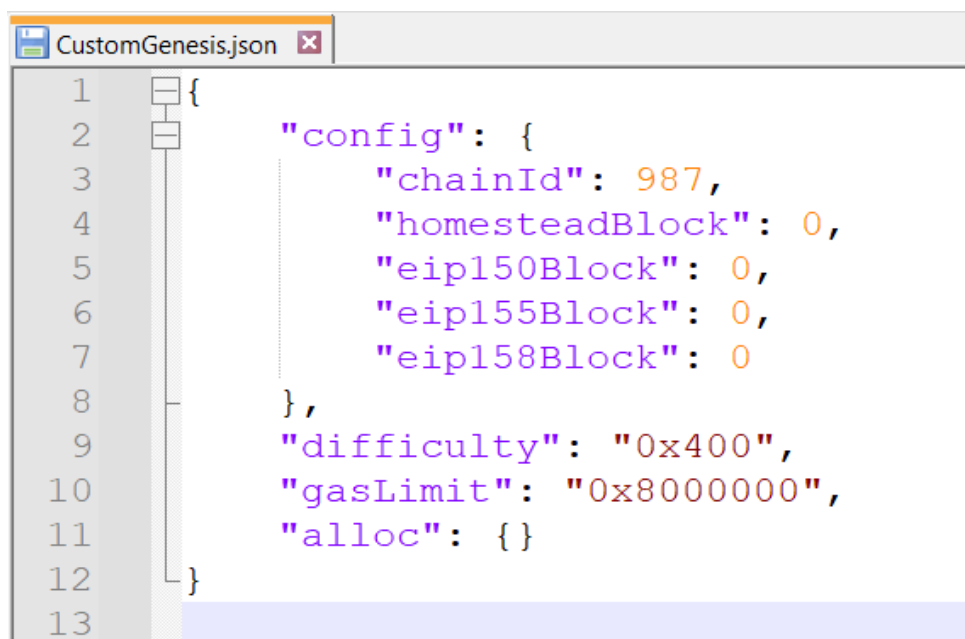
Output:

```
C:\Users\schau\Music\Blockchain Practicals>node main.js  
{  
  "chain": [  
    {  
      "index": 0,  
      "timestamp": "09/06/2024",  
      "data": "GenesisBlock",  
      "previousHash": "0",  
      "hash": "aa9262d28a2dd660edee1f21c813d95cfb5ef420da4776b2f1ad2454d1848895"  
    },  
    {  
      "index": 1,  
      "timestamp": "09/06/2024",  
      "data": {  
        "amount": 4  
      },  
      "previousHash": "aa9262d28a2dd660edee1f21c813d95cfb5ef420da4776b2f1ad2454d1848895",  
      "hash": "05a1db13df9faa0e3d2e845e177538e310a68c32d6edcb45740fedcfabe1db54"  
    },  
    {  
      "index": 2,  
      "timestamp": "09/06/2024",  
      "data": {  
        "amount": 8  
      },  
      "previousHash": "05a1db13df9faa0e3d2e845e177538e310a68c32d6edcb45740fedcfabe1db54",  
      "hash": "57f583ebe09d59c17d73892ec244180afa91d0e23afdcc853c2f338ca267d7af"  
    }  
  ]  
}
```


D. Demonstrate mining using geth on your private network.

Step 1-> Create a folder named ethermine and a JSON file named genesis.json and write the following lines in it.

```
Genesis.json
{
"config": {
"chainId": 987, "homesteadBlock": 0,
"eip150Block": 0, "eip155Block": 0, "eip158Block": 0
},
"difficulty": "0x400",
"gasLimit": "0x8000000",
"alloc": {}
}
```



```
CustomGenesis.json
1 {
2   "config": {
3     "chainId": 987,
4     "homesteadBlock": 0,
5     "eip150Block": 0,
6     "eip155Block": 0,
7     "eip158Block": 0
8   },
9   "difficulty": "0x400",
10  "gasLimit": "0x8000000",
11  "alloc": {}
12 }
13
```

Step 2-> Run command geth account new --datadir

C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine

testnet-blockchain.

```
C:\Users\Achsah>geth account new --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine
INFO [04-20|20:03:09.337] Maximum peer count          ETH=50 LES=0 total=50
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:  0x77CB2BdBC0f1743bC73E92f1a8b1AB80BEDB35AE
Path of the secret key file: C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\key
store\UTC--2023-04-20T14-33-26.959134300Z--77cb2bdbc0f1743bc73e92f1a8blab80bedb35ae

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Step 3-> Run command `geth account new --datadir`

`C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine`

```
C:\Users\Achsah>geth --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine i
nit C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\genesis.json
Fatal: invalid genesis file: math/big: cannot unmarshal "\"3792\"" into a *big.Int

C:\Users\Achsah>geth --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine i
nit C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\genesis.json
INFO [04-20|20:23:47.707] Maximum peer count                ETH=50 LES=0 total=50
INFO [04-20|20:23:47.717] Set global gas cap                 cap=50,000,000
INFO [04-20|20:23:47.720] Using leveledb as the backing database
INFO [04-20|20:23:47.720] Allocated cache and file handles   database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata cache=16.00MiB handles=16
INFO [04-20|20:23:47.741] Using LevelDB as the backing database
INFO [04-20|20:23:47.765] Opened ancient database            database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata\ancient\chain readonly=false
INFO [04-20|20:23:47.767] Writing custom genesis block
INFO [04-20|20:23:47.773] Persisted trie from memory database nodes=1 size=147.00B time="636.4µs"
```

Step 4-> Run command `geth --identity "localB" --http --http.port "8280" --http.corsdomain "*" --http.api "db,eth,net,web3" --datadir`

`"C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine"`

`--port "30303" --nodiscover --networkid 5777 console`. This command will enable geth console

```
C:\Users\Achsah>geth --identity "localB" --http --http.port "8280" --http.corsdomain "*" --http.api
"db,eth,net,web3" --datadir "C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine" --
port "30303" --nodiscover --networkid 5777 console
INFO [04-20|20:29:41.383] Maximum peer count                ETH=50 LES=0 total=50
INFO [04-20|20:29:41.389] Set global gas cap                 cap=50,000,000
INFO [04-20|20:29:41.392] Allocated trie memory caches       clean=154.00MiB dirty=256.00MiB
INFO [04-20|20:29:41.396] Using leveledb as the backing database
INFO [04-20|20:29:41.396] Allocated cache and file handles   database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata cache=512.00MiB handles=8192
INFO [04-20|20:29:41.412] Using LevelDB as the backing database
INFO [04-20|20:29:41.420] Opened ancient database            database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata\ancient\chain readonly=false
INFO [04-20|20:29:41.423] Disk storage enabled for ethash caches dir=C:\Users\Achsah\Documents\MSc
IT\sem4\blockchain_practical\ethermine\geth\ethash count=3
INFO [04-20|20:29:41.424] Disk storage enabled for ethash DAGs dir=C:\Users\Achsah\AppData\Local
\Ethash count=2
INFO [04-20|20:29:41.426] Initialising Ethereum protocol     network=5777 dbversion=<nil>
INFO [04-20|20:29:41.427]
INFO [04-20|20:29:41.430] -----
```

Step 5-> Run the command

`miner.setEtherbase('0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0')`

in the geth console

Step 6-> Run the command `miner.start()` to start mining

```
To exit, press ctrl-d or type exit
> INFO [04-20|20:29:45.021] Mapped network port                proto=tcp extport=30303 intport=30303
NP IGDv1-IP1"
>
> miner.setEtherbase('0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0')
true
> miner.start()
INFO [04-20|20:34:45.673] Updated mining threads              threads=4
INFO [04-20|20:34:45.674] Transaction pool price threshold updated price=1,000,000,000
null
> INFO [04-20|20:34:45.683] Commit new sealing work             number=1 sealhash=2e6f57..6db9c6 unde
=0 fees=0 elapsed=7.571ms
INFO [04-20|20:34:45.686] Commit new sealing work             number=1 sealhash=2e6f57..6db9c6 unde
fees=0 elapsed=9.940ms
INFO [04-20|20:34:47.975] Generating DAG in progress          epoch=0 percentage=0 elapsed=1.636s
INFO [04-20|20:34:49.873] Generating DAG in progress          epoch=0 percentage=1 elapsed=3.534s
```

Step 7-> Below screenshots are the mining processes running on your local machine.

```
INFO [04-20|20:38:42.556] Generating DAG in progress      epoch=0 percentage=98 elapsed=3m5
6.216s
INFO [04-20|20:38:46.897] Generating DAG in progress      epoch=0 percentage=99 elapsed=4m0
.557s
INFO [04-20|20:38:46.901] Generated ethash verification cache  epoch=0 elapsed=4m0.561s
INFO [04-20|20:38:48.755] Successfully sealed new block    number=1 sealhash=2e6f57..6db9c6
hash=ccf3e9..10adff elapsed=4m3.071s
INFO [04-20|20:38:48.765] "⚡ mined potential block"      number=1 hash=ccf3e9..10adff
INFO [04-20|20:38:48.756] Commit new sealing work        number=2 sealhash=cb4ba0..84e1dd
uncles=0 txs=0 gas=0 fees=0 elapsed="504.9us"
INFO [04-20|20:38:48.770] Commit new sealing work        number=2 sealhash=cb4ba0..84e1dd
uncles=0 txs=0 gas=0 fees=0 elapsed=14.488ms
INFO [04-20|20:38:49.389] Successfully sealed new block    number=2 sealhash=cb4ba0..84e1dd
hash=4c7137..a04b67 elapsed=632.526ms
```

Step 8-> To stop the mining press Ctrl+D

```
INFO [04-20|20:39:21.980] Commit new sealing work        number=17 sealhash=923697..cb5b4d
uncles=0 txs=0 gas=0 fees=0 elapsed=117.201ms
INFO [04-20|20:39:21.984] Ethereum protocol stopped
INFO [04-20|20:39:22.046] Transaction pool stopped
INFO [04-20|20:39:22.047] Writing cached state to disk    block=16 hash=f09f60..c23237 root
=0c083a..cddeff
INFO [04-20|20:39:22.081] Persisted trie from memory database  nodes=3 size=408.00B time=1.5741m
s gcnodes=0 gcsiz=0.00B gctime=0s livenodes=31 livesize=3.83KiB
INFO [04-20|20:39:22.087] Writing cached state to disk    block=15 hash=d73b6d..f4a2cf root
=903c8d..6038c0
INFO [04-20|20:39:22.089] Persisted trie from memory database  nodes=2 size=262.00B time=0s
gcnodes=0 gcsiz=0.00B gctime=0s livenodes=29 livesize=3.58KiB
INFO [04-20|20:39:22.098] Writing snapshot state to disk    root=d56154..abe42a
INFO [04-20|20:39:22.130] Persisted trie from memory database  nodes=0 size=0.00B time=0s
gcnodes=0 gcsiz=0.00B gctime=0s livenodes=29 livesize=3.58KiB
INFO [04-20|20:39:22.135] Writing clean trie cache to disk  path=C:\Users\Achsah\Documents\MS
cIT\sem4\blockchain_practical\ethermine\geth\triecache threads=4
INFO [04-20|20:39:22.323] Persisted the clean trie cache    path=C:\Users\Achsah\Documents\MS
cIT\sem4\blockchain_practical\ethermine\geth\triecache elapsed=143.729ms
INFO [04-20|20:39:22.490] Blockchain stopped
```

Practical No. 3

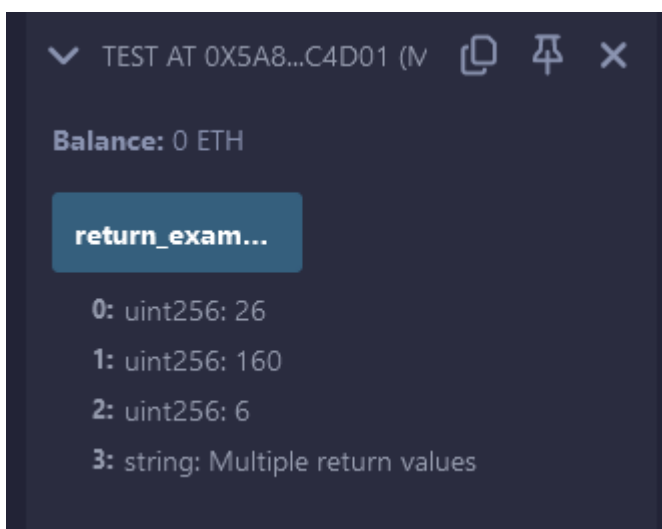
A. Write a Solidity program that demonstrates various types of functions including regular functions, view functions, pure functions, and the fallback function.

1. Functions

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

contract Test {
    function return_example()
        public
        pure
        returns (
            uint256,
            uint256,
            uint256,
            string memory
        )
    {
        uint256 num1 = 10;
        uint256 num2 = 16;
        uint256 sum = num1 + num2;
        uint256 prod = num1 * num2;
        uint256 diff = num2 - num1;
        string memory message = "Multiple return values";
        return (sum, prod, diff, message);
    }
}
```

Output:



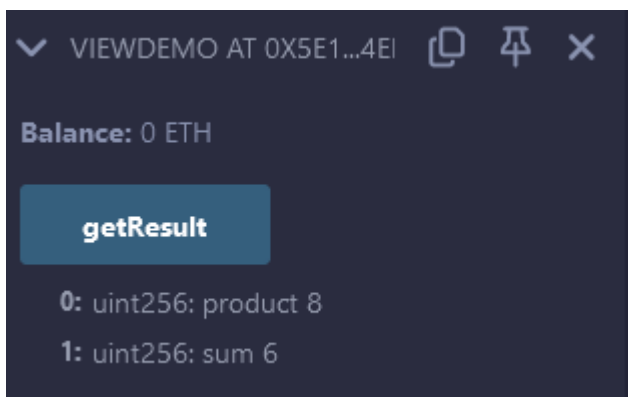
2. View Function

```
pragma solidity ^0.5.0;

contract ViewDemo
{
    uint256 num1 = 2;
    uint256 num2 = 4;

    function getResult() public view returns (uint256 product, uint256 sum) {
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

Output:



VIEWDEMO AT 0X5E1...4EI

Balance: 0 ETH

getResult

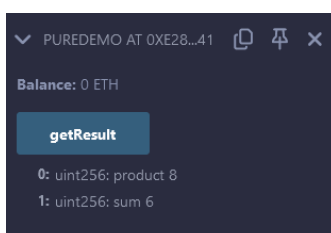
0: uint256: product 8
1: uint256: sum 6

3. Pure Function:

```
pragma solidity ^0.5.0;

contract PureDemo {
    function getResult() public pure returns (uint256 product, uint256 sum) {
        uint256 num1 = 2;
        uint256 num2 = 4;
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

Output:



PUREDEMO AT 0XE28...41

Balance: 0 ETH

getResult

0: uint256: product 8
1: uint256: sum 6

B. Write a Solidity program that demonstrates function overloading, mathematical functions, and cryptographic functions.

1. Function Overloading

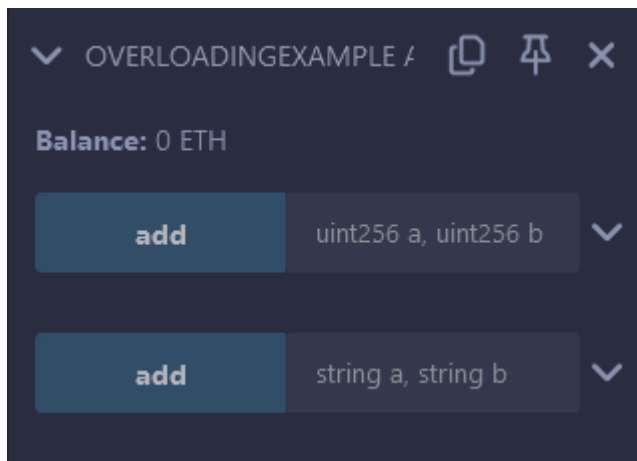
```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract OverloadingExample {

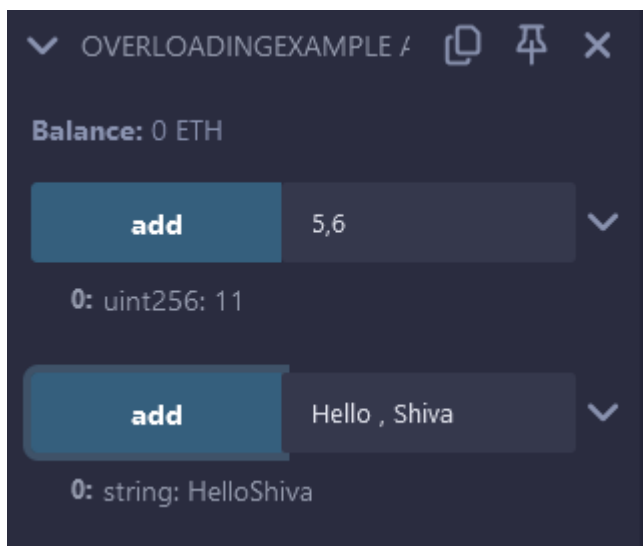
    function add(uint256 a, uint256 b) public pure returns (uint256) {
        return a + b;
    }

    function add(string memory a, string memory b) public pure returns (string memory) {
        return string(abi.encodePacked(a, b));
    }
}
```

Output:



Give integer and string values to both add functions as below.



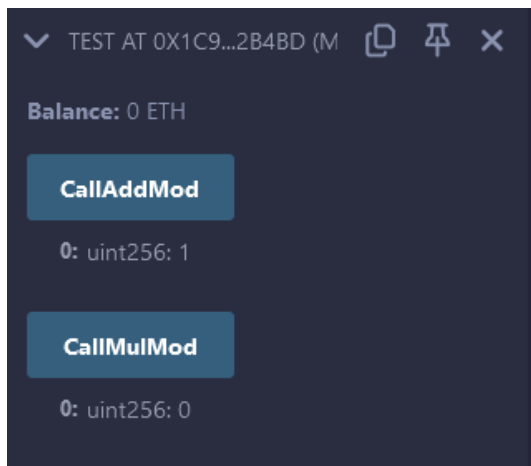
2. Mathematical Function

```
pragma solidity ^0.5.0;

contract Test {
    function CallAddMod() public pure returns(uint) {
        return addmod(7, 3, 3);
    }

    function CallMulMod() public pure returns(uint) {
        return mulmod(7, 3, 3);
    }
}
```

Output:



3. Cryptographic Functions.





```
pragma solidity ^0.5.0;

contract Test {
    function callKeccak256() public pure returns (bytes32 result) {
        return keccak256(abi.encodePacked("BLOCKCHAIN"));
    }

    function callsha256() public pure returns (bytes32 result) {
        return sha256(abi.encodePacked("BLOCKCHAIN"));
    }

    function callripemd() public pure returns (bytes20 result) {
        return ripemd160(abi.encodePacked("BLOCKCHAIN"));
    }
}
```

Output:

 TEST AT 0X93F...C96CC (M)   

Balance: 0 ETH

callKeccak256

0: bytes32: result 0xedb146af3dfbb7f9
95ec65b249dd88d2d54ca0707a84f
08c25e4cc08f5168aea

callripemd

0: bytes20: result 0x638cf4481022e8b
e8fab43fa5f76ccffc62f2a09

callsha256

0: bytes32: result 0xdffdca1f7dd5c94af
ea2936253a2463a26aad06fa9b5f36
b5affc8851e8c8d42

C. Write a Solidity program that demonstrates various features including contracts, inheritance, constructors, abstract contracts, interfaces.

1. Contracts

```
pragma solidity ^0.5.0;

contract ContractDemo {
    string message = "Hello Shivam";

    function dispMsg() public view returns (string memory) {
        return message;
    }
}
```

Output:



2. Inheritance

```
pragma solidity >=0.4.22 <0.6.0;

contract Parent {
    uint256 internal sum;

    function setValue() external {
        uint256 a = 10;
        uint256 b = 20;
        sum = a + b;
    }
}

contract Child is Parent {
    function getValue() external view returns (uint256) {
        return sum;
    }
}
```

```

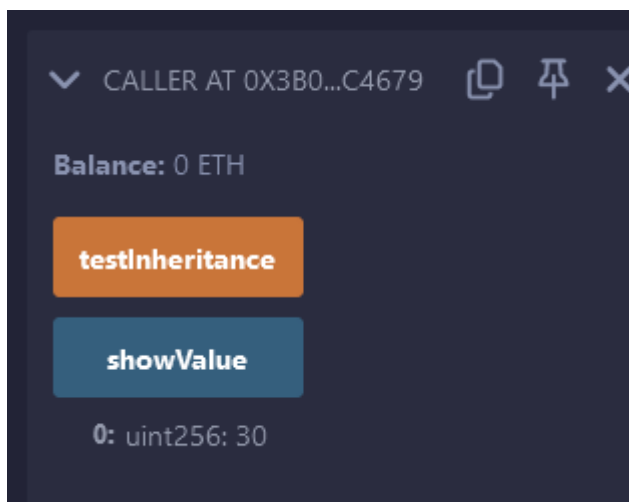
contract Caller {
    Child cc = new Child();

    function testInheritance() public returns (uint256) {
        cc.setValue();
        return cc.getValue();
    }

    function showValue() public view returns (uint256) {
        return cc.getValue();
    }
}

```

Output:



3. Abstract Contracts

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.5.17;

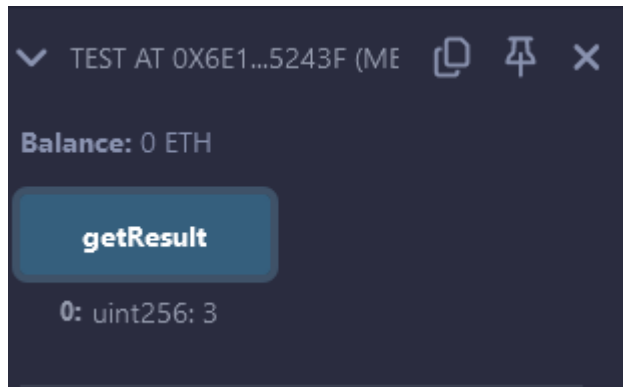
contract Calculator {
    function getResult() external view returns (uint256);
}

contract Test is Calculator {
    constructor() public {}

    function getResult() external view returns (uint256) {
        uint256 a = 1;
        uint256 b = 2;
        uint256 result = a + b;
        return result;
    }
}

```

Output:



4. Constructor

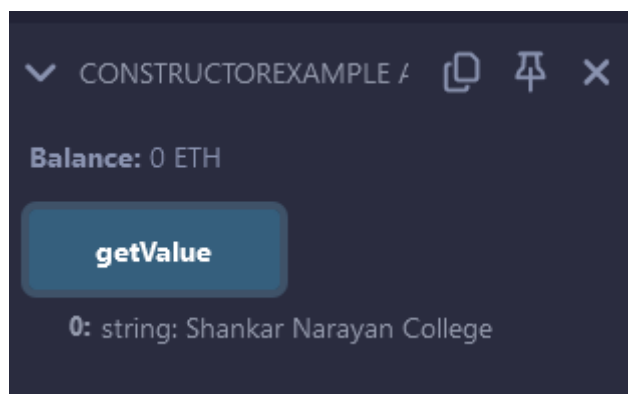
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

// Creating a contract
contract constructorExample {
    string str;

    constructor() public {
        str = "Shankar Narayan College";
    }

    function getValue() public view returns (string memory) {
        return str;
    }
}
```

Output



5. Interfaces:

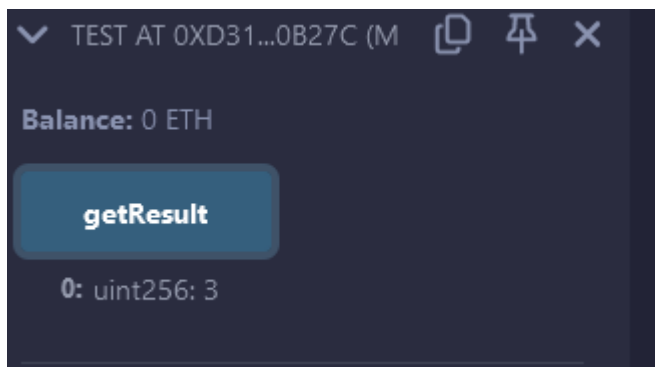
```
pragma solidity ^0.5.0;

interface Calculator {
    function getResult() external view returns (uint);
}

contract Test is Calculator {
    constructor() public {}

    function getResult() external view returns (uint) {
        uint a = 1;
        uint b = 2;
        uint result = a + b;
        return result;
    }
}
```

Output:



D. Write a Solidity program that demonstrates use of libraries, assembly, events, and error handling.

1. Libraries

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

library myMathLib {
    function sum(uint256 a, uint256 b) public pure returns (uint256) {
        return a + b;
    }

    function exponent(uint256 a, uint256 b) public pure returns (uint256) {
        return a ** b;
    }
}
```

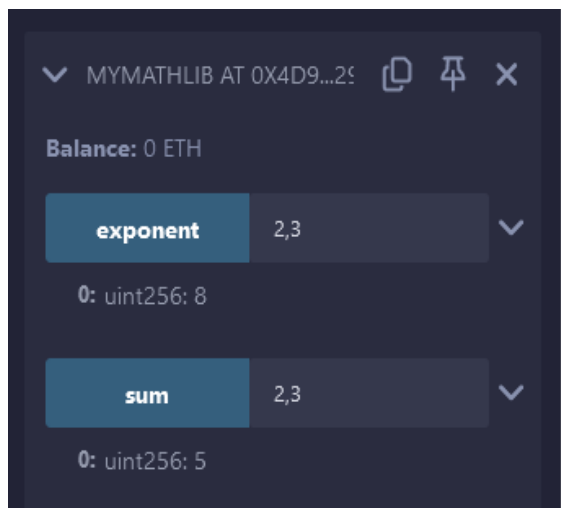
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

import "contracts/myLIB.sol";

contract UseLib {
    function getsum(uint256 x, uint256 y) public pure returns (uint256) {
        return myMathLib.sum(x, y);
    }

    function getexponent(uint256 x, uint256 y) public pure returns (uint256) {
        return myMathLib.exponent(x, y);
    }
}
```

Output



2. Assembly

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract InlineAssembly {
    // Defining function
    function add(uint256 a) public view returns (uint256 b) {
        assembly {
            let c := add(a, 16)
            mstore(0x80, c)
            {
                let d := add(sload(c), 12)
                b := d
            }

            b := add(b, c)
        }
    }
}
```

Output



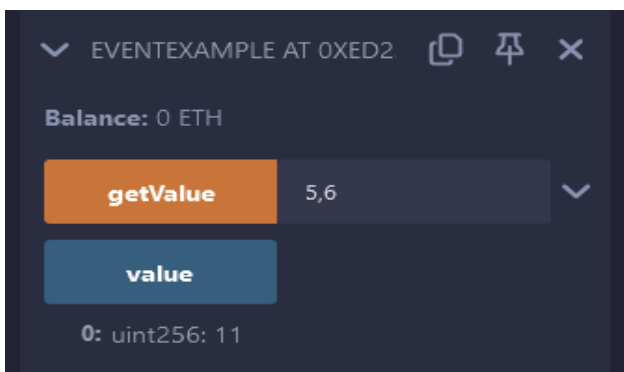
3. Events

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

// Creating a contract
contract eventExample {
    // Declaring state variables
    uint256 public value = 0;
    // Declaring an event
    event Increment(address owner);
}
```

```
// Defining a function for logging event
function getValue(uint256 _a, uint256 _b) public {
    emit Increment(msg.sender); // Emitting the Increment event with the caller's address
    value = _a + _b; // Updating the value state variable
}
}
```

Output:



4. Error Handling

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.5.17;

contract ErrorDemo {
    function getSum(uint256 a, uint256 b) public pure returns (uint256) {
        uint256 sum = a + b;
        // require(sum < 255, "Invalid");
        assert(sum < 255);
        return sum;
    }
}
```

Output:

