PRACTICAL REPORT ON
Blockchain
Submitted by:
ESHAAN RAKESH SHUKLA

Under the esteemed guidance of
Mrs. Bimal Kumbhar
Designation: Professor

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)
SEM-IV
2O24-25

UNIVERSITY OF MUMBAI

MASTER OF SCIENCE IN
INFORMATION TECHNOLOGY

Abhinav College of Arts, Commerce & Science
(*Affiliated to University of Mumbai*)
Mumbai, PIN CODE 401105
MAHARASHTRA

ABHINAV COLLEGE OF ARTS, COMMERCE & SCIENCE

(Affiliated to University of Mumbai)

THANE – MAHARASHTRA – 401105

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is Certify that Mr.____

Studying in M.Sc.I.T. Part – II in our college having Roll no. _________________________

and Exam Seat No.___________________________________ has successfully completed

the Practical of **Blockchain** according to the prescribed University of Mumbai list

of practical’ s in the academic year 2024 - 2025.

Date:-

Professor in-Charge **Co-Ordinator** **Principal**

External in-Charge

Sr No:		Practical Description	Sign
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.	
	b	Allow users to create multiple transactions and display them in an organized format.	
	c	Implement a function to add new blocks to the miner and dump the blockchain.	
2	a	Write a Python program to demonstrate mining.	
	b	Demonstrate the use of the Bitcoin Core API to interact with a Bitcoin Core node.	
	c	Demonstrating the process of running a blockchain node on your local machine.	
3	a	Write a Solidity program that demonstrates various types of functions including regular functions, view functions, pure functions, and the fallback function.	
	b	Write a Solidity program that demonstrates function overloading, mathematical functions, and cryptographic functions.	
	c	Write a Solidity program using libraries, assembly, events, and error handling.	

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.

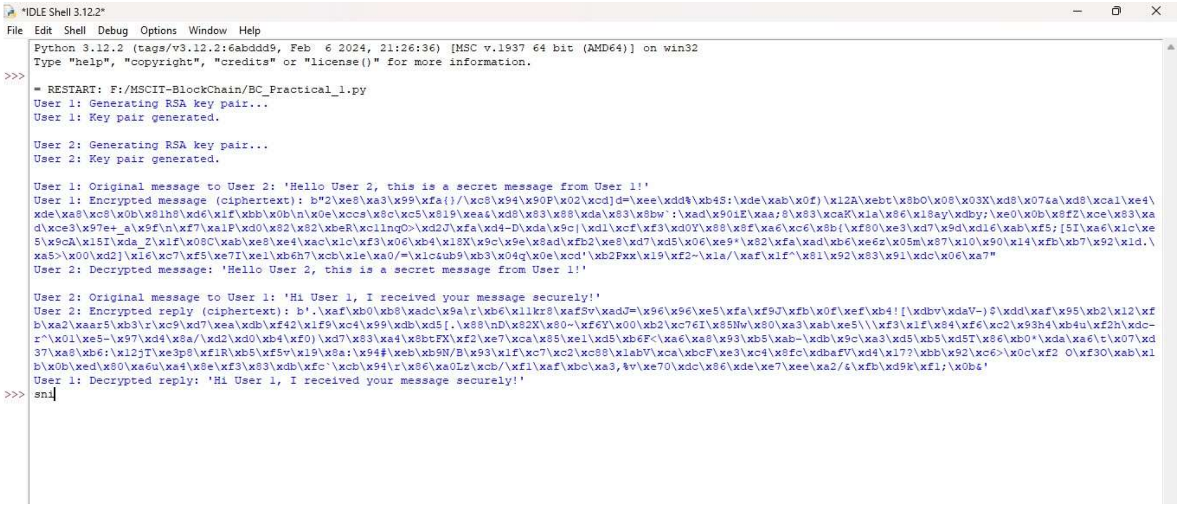
Code:

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
```

```
original_message_user2 = "Hi User 1, I received your message securely!"

print(f"\nUser 2: Original message to User 1: '{original_message_user2}'")


# User 2 encrypts the reply using User 1's public key

encrypted_message_user2_to_user1 = encrypt_message(user1_public_key, original_message_user2)

print(f"User 2: Encrypted reply (ciphertext): {encrypted_message_user2_to_user1}")


# User 1 receives and decrypts the reply using their private key

decrypted_message_user1 = decrypt_message(user1_private_key,
encrypted_message_user2_to_user1)

print(f"User 1: Decrypted reply: '{decrypted_message_user1}'")
```

Output:



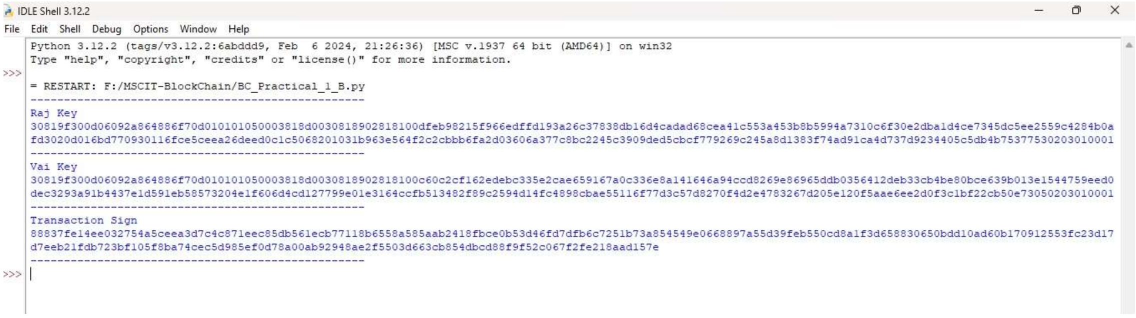
```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:/MSCIT-BlockChain/BC_Practical_1.py
User 1: Generating RSA key pair...
User 1: Key pair generated.

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\xe8\xa3\x99\xfa{}/\xc8\x94\x90P\x02\xcd]d=\xee\xdd%\xb4S:\xde\xab\x0f)\x12A\xebt\x8bO\x08\x03X\xd8\x07&a\xd8\xca1\xe4\
xde\xa8\xc8\x0b\x81h8\xd6\x1f\xbb\x0b\n\x0e\xccs\x8c\xc5\x819\xea&\xd8\x83\x88\xda\x83\x8bw`:\xad\x90iE\xaa;8\x83\xcaK\x1a\x86\x18ay\xdby;\xe0\x0b\x8fZ\xce\x83\xa
d\xce3\x97e+_a\x9f\n\xf7\xa1P\xd0\x82\x82\xbeR\xc11nqO>\xd2J\xfa\xd4-D\xda\x9c|\xd1\xcf\xf3\xd0Y\x88\x8f\xa6\xc6\x8b{\xf80\xe3\xd7\x9d\xd16\xab\xf5;[5I\xa6\x1c\xe
5\x9cA\x15I\xda_Z\x1f\x08C\xab\xe8\xe4\xac\x1c\xf3\x06\xb4\x18X\x9c\x9e\x8ad\xfb2\xe8\xd7\xd5\x06\xe9*\x82\xfa\xad\xb6\xe6z\x05m\x87\x10\x90\x14\xfb\xb7\x92\x1d.\
xa5>\x00\xd2]\x16\xc7\xf5\xe7I\xe1\xb6h7\xcb\x1e\xa0/=\x1c&ub9\xb3\x04q\x0e\xcd'\xb2Pxx\x19\xf2~\x1a/\xaf\x1f^\x81\x92\x83\x91\xdc\x06\xa7"
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'.\xaf\xb0\xb8\xadc\x9a\r\xb6\x11kr8\xafSv\xadJ=\x96\x96\xe5\xfa\xf9J\xfb\x0f\xef\xb4![\xdbv\xdaV-)$\xdd\xaf\x95\xb2\x12\xf
b\xa2\xaar5\xb3\r\xc9\xd7\xea\xdb\xf42\x1f9\xc4\x99\xdb\xd5[.\x88\nD\x82X\x80~\xf6Y\x00\xb2\xc76I\x85Nw\x80\xa3\xab\xe5\\\xf3\x1f\x84\xf6\xc2\x93h4\xb4u\xf2h\xdc-
r^\x01\xe5-\x97\xd4\x8a/\xd2\xd0\xb4\xf0)\xd7\x83\xa4\x8btFX\xf2\xe7\xca\x85\xe1\xd5\xb6F<\xa6\xa8\x93\xb5\xab-\xdb\x9c\xa3\xd5\xb5\xd5T\x86\xb0*\xda\xa6\t\x07\xd
37\xa8\xb6:\x12jT\xe3p8\xf1R\xb5\xf5v\x19\x8a:\x94#\xeb\xb9N/B\x93\x1f\xc7\xc2\xc88\x1abV\xca\xbcF\xe3\xc4\x8fc\xdbafV\xd4\x17?\xbb\x92\xc6>\x0c\xf2 O\xf3O\xab\x1
b\x0b\xed\x80\xa6u\xa4\x8e\xf3\x83\xdb\xfc`\xcb\x94\r\x86\xa0Lz\xcb/\xf1\xaf\xbc\xa3,%v\xe70\xdc\x86\xde\xe7\xee\xa2/&\xfb\xd9k\xf1;\x0b&'
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)
```



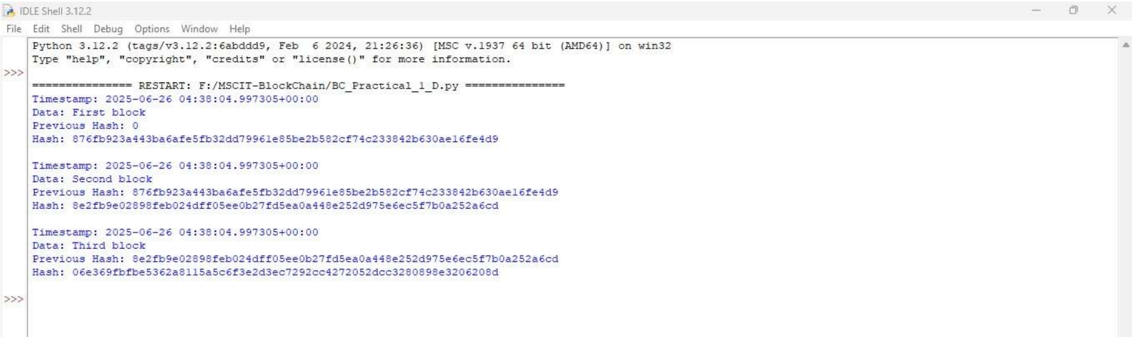
```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:/MSCIT-BlockChain/BC_Practical_1_B.py
--------------------------------------------------
Raj Key
30819f300d06092a864886f70d010101050003818d0030818902818100dfeb98215f966edffd193a26c37838db16d4cadad68cea41c553a453b8b5994a7310c6f30e2dba1d4ce7345dc5ee2559c4284b0a
fd3020d016bd770930116fce5ceea26deed0c1c5068201031b963e564f2c2cbbb6fa2d03606a377c8bc2245c3909ded5cbcf779269c245a8d1383f74ad91ca4d737d9234405c5db4b75377530203010001
--------------------------------------------------
Vai Key
30819f300d06092a864886f70d010101050003818d0030818902818100c60c2cf162edebc335e2cae659167a0c336e8a141646a94ccd8269e86965ddb0356412deb33cb4be80bce639b013e1544759eed0
dec3293a91b4437e1d591eb58573204e1f606d4cd127799e01e3164ccfb513482f89c2594d14fc4898cbae55116f77d3c57d8270f4d2e4783267d205e120f5aae6ee2d0f3c1bf22cb50e73050203010001
--------------------------------------------------
Transaction Sign
88837fe14ee032754a5ceea3d7c4c871eec85db561ecb77118b6558a585aab2418fbce0b53d46fd7dfb6c7251b73a854549e0668897a55d39feb550cd8a1f3d658830650bdd10ad60b170912553fc23d17
d7eeb21fdb723bf105f8ba74cec5d985ef0d78a00ab92948ae2f5503d663cb854dbcd88f9f52c067f2fe218aad157e
--------------------------------------------------
>>> |
```

C. 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:



```
    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: 06e369fbfbe5362a8115a5c6f3e2d3ec7292cc4272052dcc3280898e3206208d

>>>
```

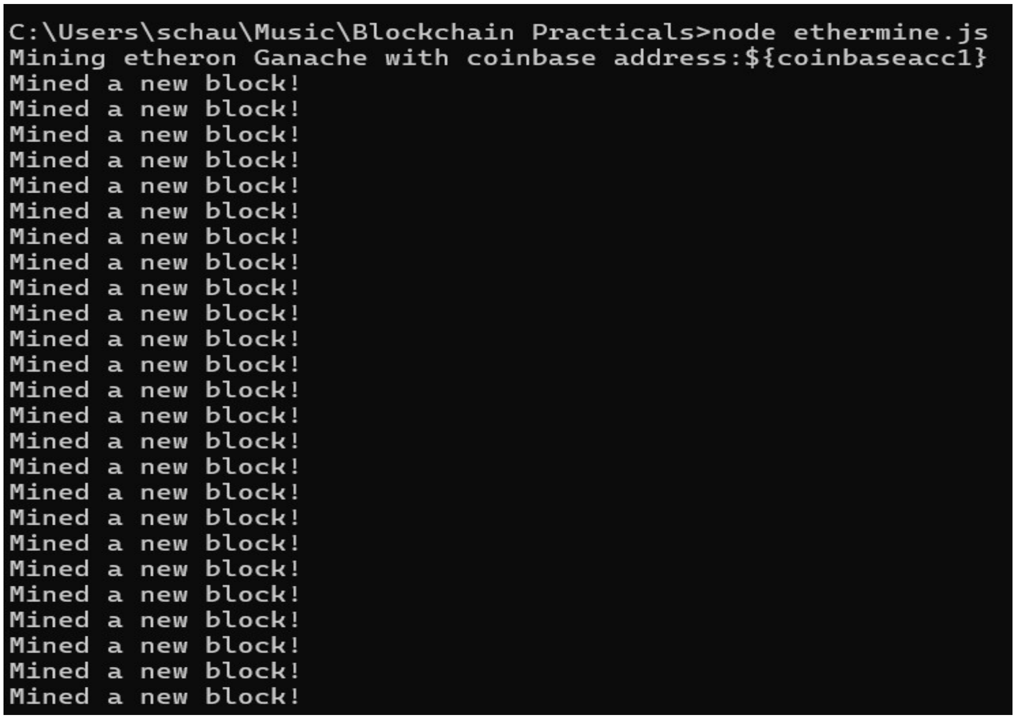
Practical No. 2

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:



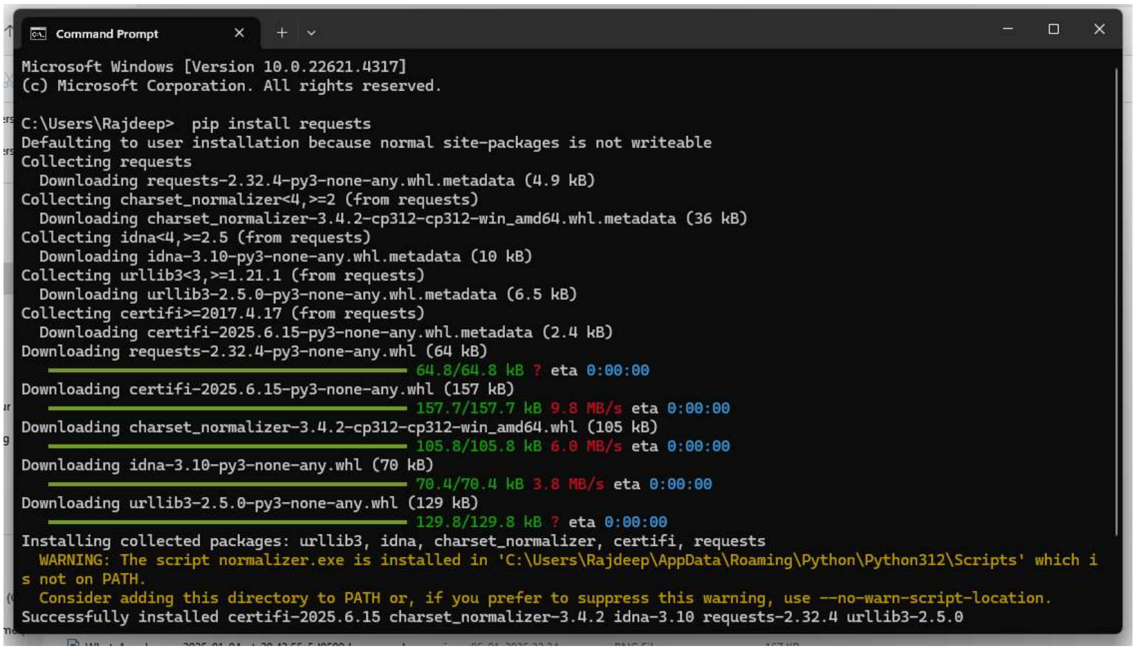
```
C:\Users\schau\Music\Blockchain Practicals>node ethermine.js
Mining etheron Ganache with coinbase address:${coinbaseacc1}
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
```

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

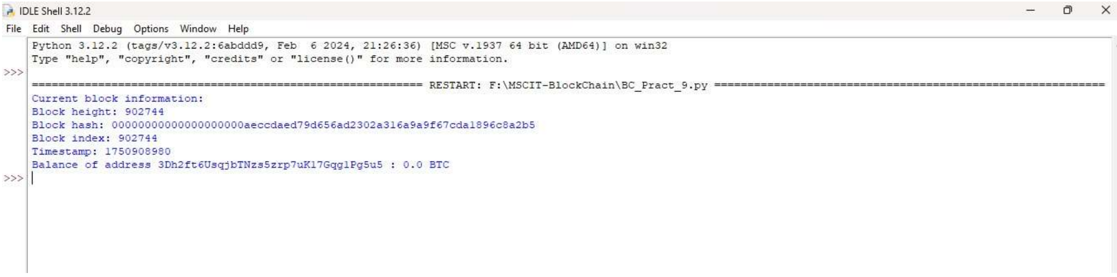
Bitcoin Core Api

```
# 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 i
s 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
```



```
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_Pract_9.py ============================================================
Current block information:
Block height: 902744
Block hash: 00000000000000000000aeccdaed79d656ad2302a316a9a9f67cda1896c8a2b5
Block index: 902744
Timestamp: 1750908980
Balance of address 3Dh2ft6UsqjbTNzs5zrp7uK17Gqg1Pg5u5 : 0.0 BTC
>>>
```

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

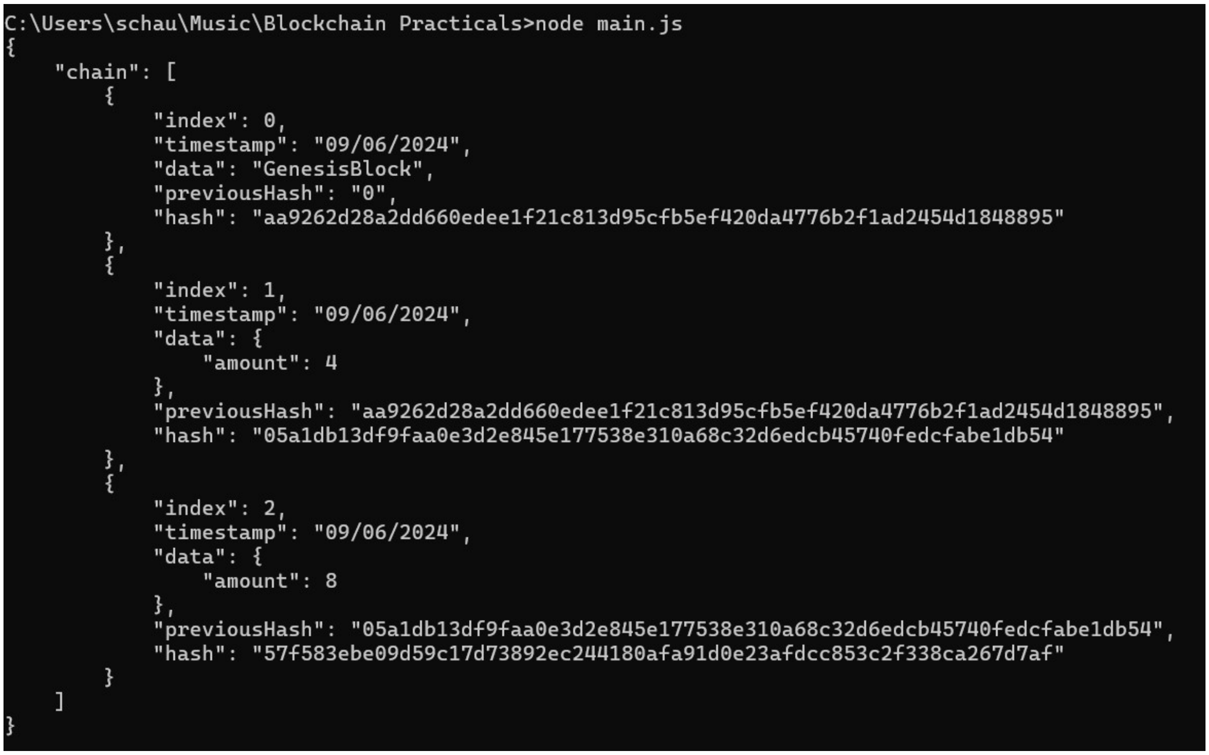
Make Sure you have Installed node.js in their System.

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(leti=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"
        }
    ]
}
```

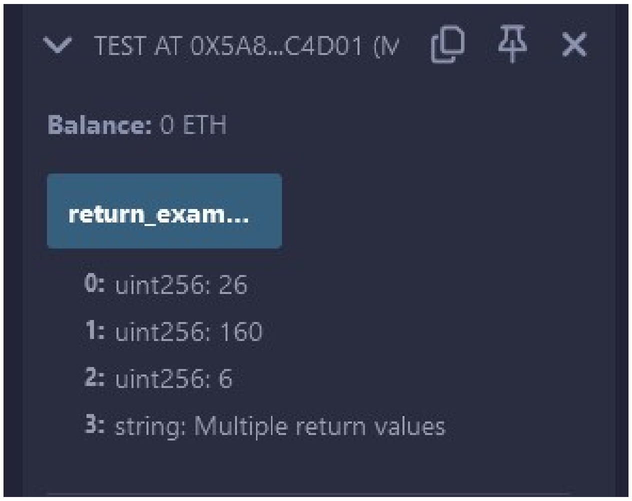
Practical No. 3

B. 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:

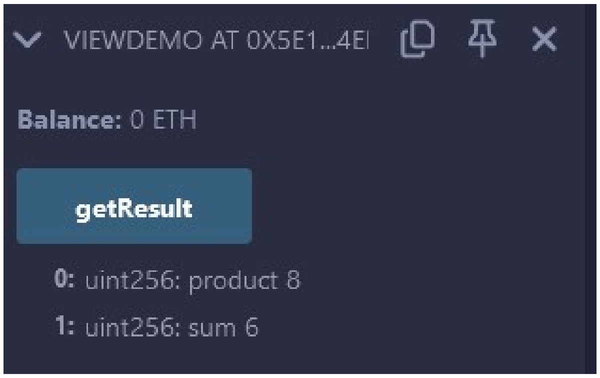


```
Balance: 0 ETH

0: uint256: 26
1: uint256: 160
2: uint256: 6
3: string: Multiple return values
```

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:



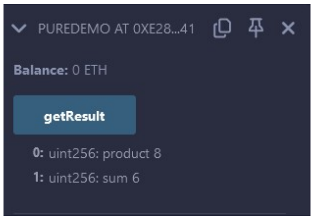
Balance: 0 ETH

```
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:



Balance: 0 ETH

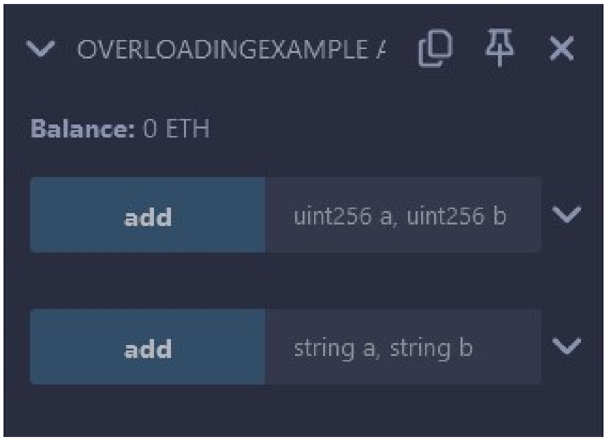
```
0: uint256: product 8
1: uint256: sum 6
```

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

1. Function Overloading

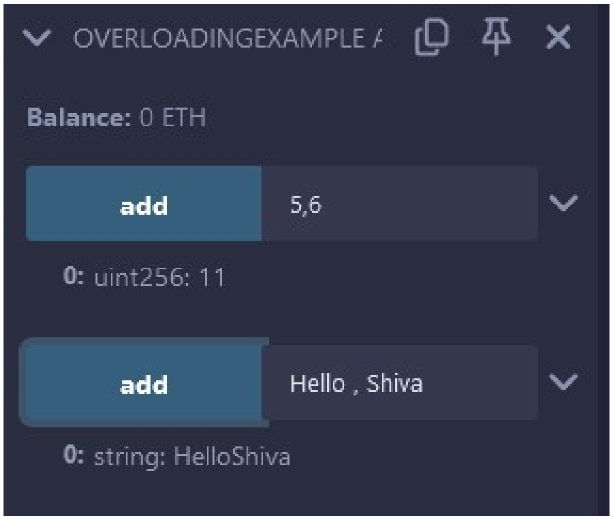
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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:



```
OVERLOADINGEXAMPLE A
Balance: 0 ETH
add    uint256 a, uint256 b
add    string a, string b
```

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

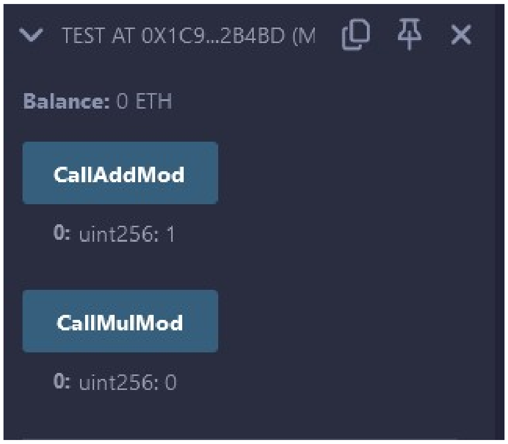


```
OVERLOADINGEXAMPLE A
Balance: 0 ETH
add    5,6
0: uint256: 11
add    Hello , Shiva
0: string: HelloShiva
```

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:

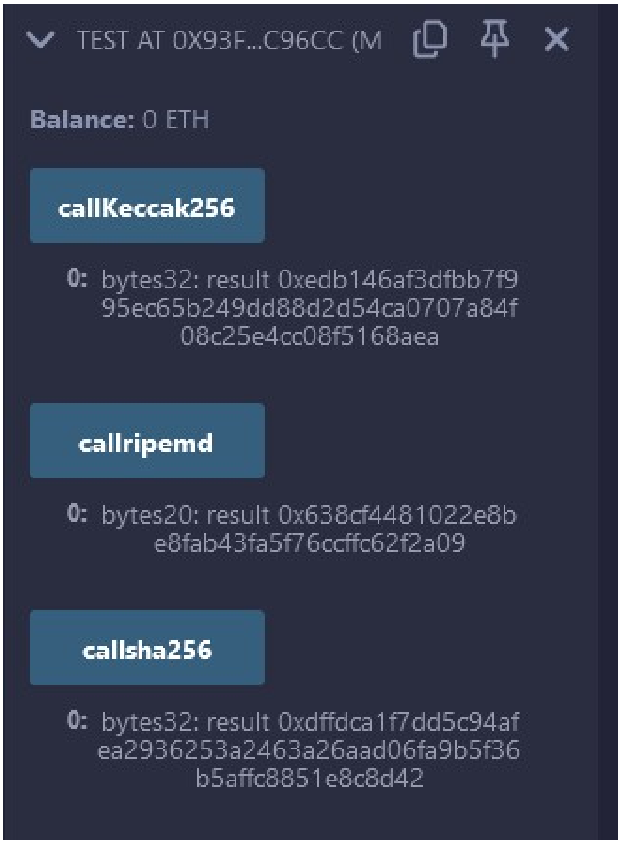


```
Balance: 0 ETH
0: uint256: 1
0: uint256: 0
```

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:



```
0: bytes32: result 0xedb146af3dfbb7f9
   95ec65b249dd88d2d54ca0707a84f
   08c25e4cc08f5168aea

0: bytes20: result 0x638cf4481022e8b
   e8fab43fa5f76ccffc62f2a09

0: bytes32: result 0xdffdca1f7dd5c94af
   ea2936253a2463a26aad06fa9b5f36
   b5affc8851e8c8d42
```

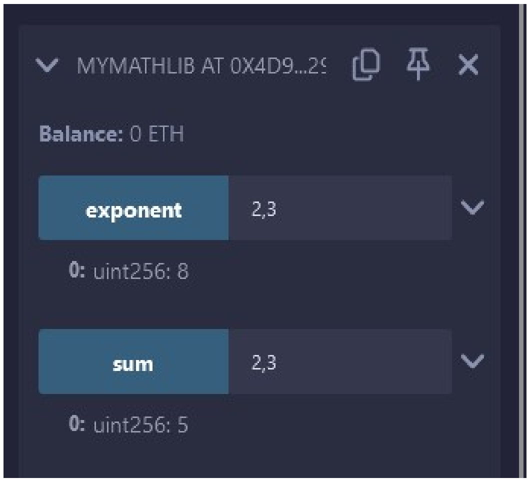
D. Write a Solidity program that demonstrates use of libraries, assembly, events, and error handling.

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

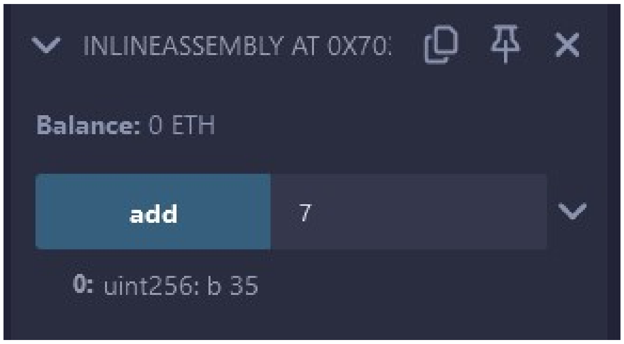


```
MYMATHLIB AT 0X4D9...29

Balance: 0 ETH

exponent    2,3
0: uint256: 8

sum    2,3
0: uint256: 5
```

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



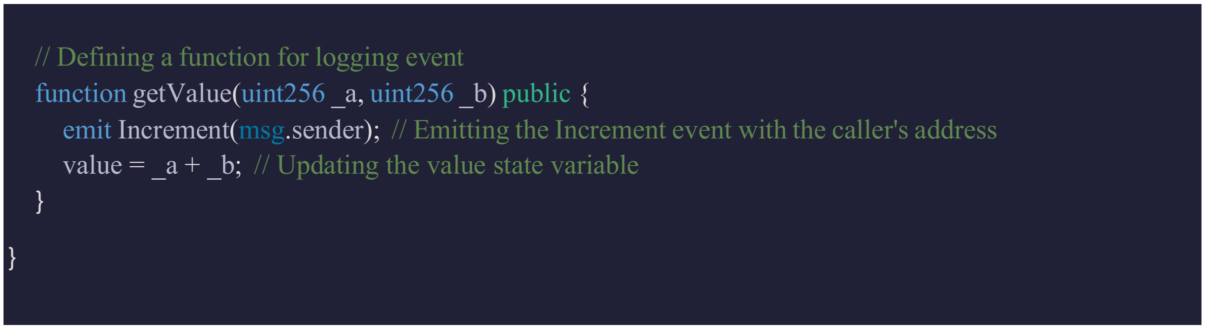
Balance: 0 ETH

7

```
0: uint256: b 35
```

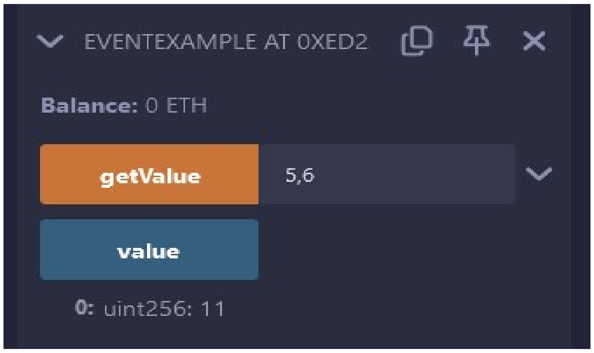
3. Events

```
// 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:

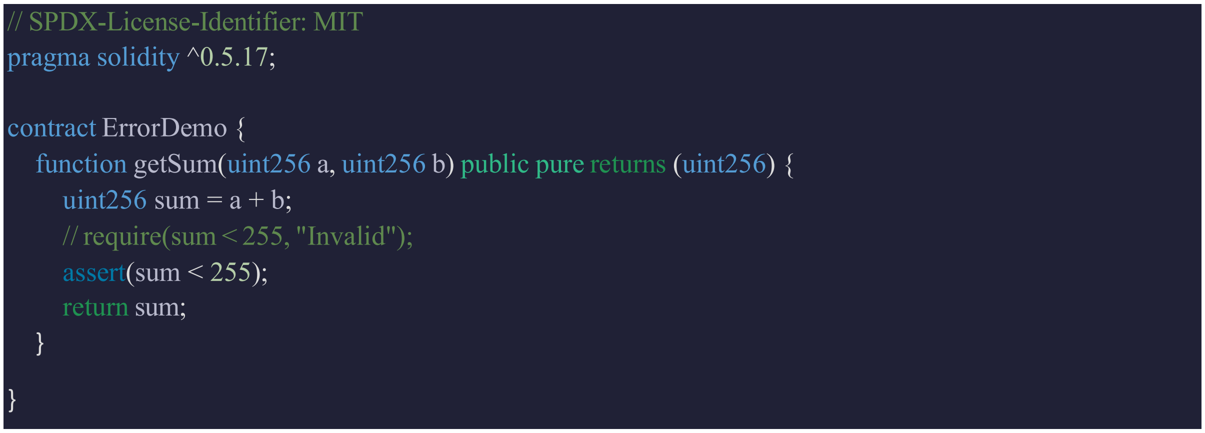


Balance: 0 ETH

5,6

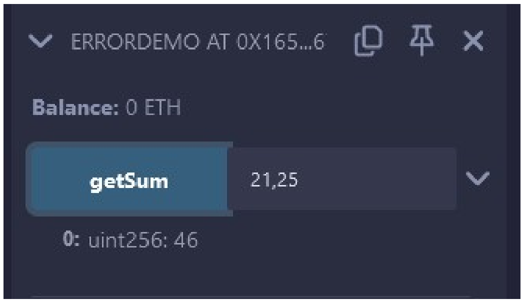
```
0: uint256: 11
```

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:



Balance: 0 ETH

21,25

```
0: uint256: 46
```