University of Mumbai

PRACTICAL JOURNAL – PAPER I

PSIT4P1a

Blockchain

SUBMITTED BY

Sandeep Sharma

SEAT NO 40426

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR QUALIFYING
M.Sc. (I.T.) PART-II (SEMESTER – IV) EXAMINATION

2022-2023

Department of Information Technology

3RD FLOOR, DR. SHANKAR DAYAL SHARMA BHAVAN, IDOL
BUILDING, VIDYANAGRI,
SANTACRUZ (E), MUMBAI – 400098.

University of Mumbai

Department of Information Technology

CERTIFICATE

This is to certify that Mr. **Sandeep Jairam Sharma Seat No. 40426** studying in **Master of Science** in **Information Technology Part II Semester IV** has satisfactorily completed the Practical of **PSIT4P1a Blockchain** as prescribed by University of Mumbai, during the academic year **2022-23**.

Signature	Signature	Signature
Guide	External Examiner Examined by	Head of the Department Certified by

College Seal

Date:

INDEX

PRACTICAL 1

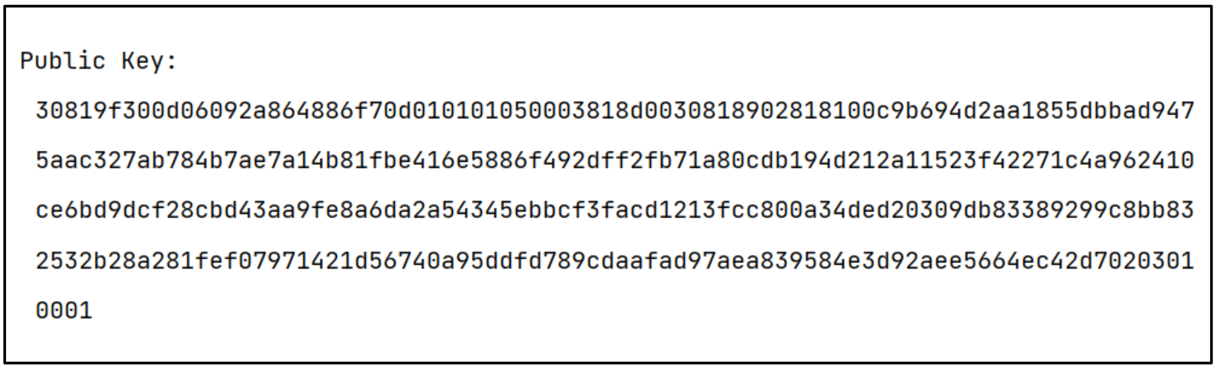
AIM: Write the following programs for Blockchain in Python:

[A] A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it.

CODE:

```
import binascii
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto import Random



class Client:
    def __init__(self):
        random = Random.new().read
        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("a
scii")

# Create an instance of the Client class
UDIT = Client()

# Print the public key (identity) of the client
print("\nPublic Key:", UDIT.identity)
```

OUTPUT:



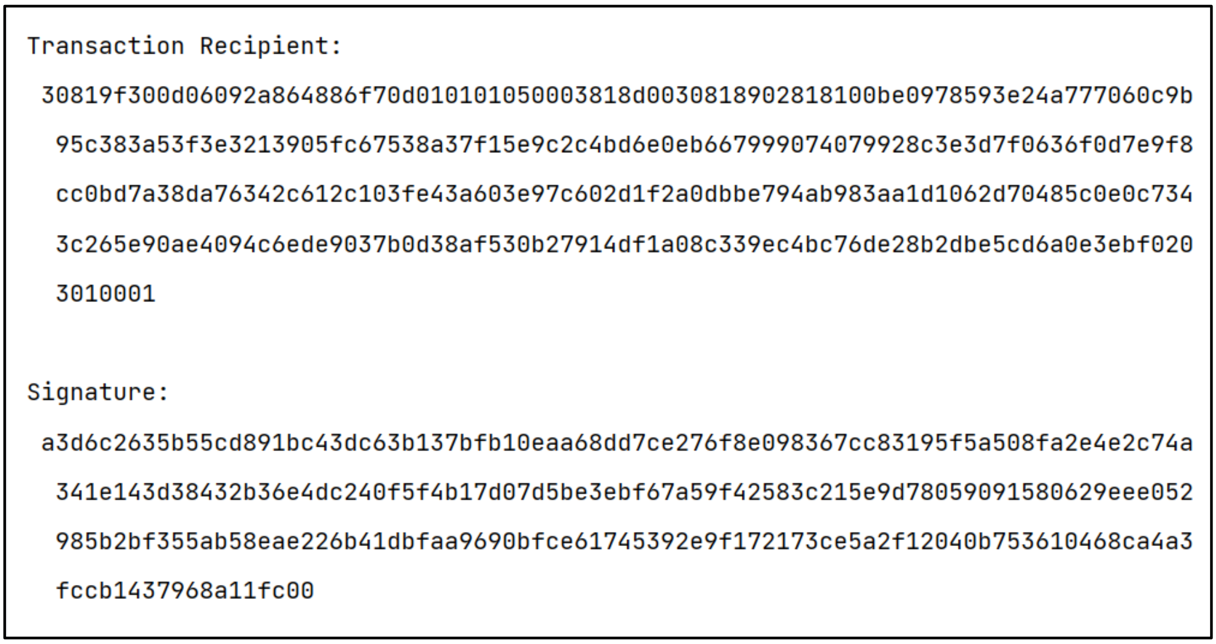
```
Public Key:
 30819f300d06092a864886f70d010101050003818d0030818902818100c9b694d2aa1855dbbad947
 5aac327ab784b7ae7a14b81fbe416e5886f492dff2fb71a80cdb194d212a11523f42271c4a962410
 ce6bd9dcf28cbd43aa9fe8a6da2a54345ebbcf3facd1213fcc800a34ded20309db83389299c8bb83
 2532b28a281fef07971421d56740a95ddfd789cdaafad97aea839584e3d92aee5664ec42d7020301
 0001
```

[B] A transaction class to send and receive money and test it.

CODE:

```
import binascii
import collections
import datetime
from Crypto.Hash import SHA
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto import Random


class Client:
    def __init__(self):
        random = Random.new().read
        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, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()

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



UDIT = Client()
UGC = Client()

t = Transaction(UDIT, UGC.identity, 5.0)

print("\nTransaction Recipient:\n", t.recipient) #
print("\nTransaction Sender:\n", t.sender) print("\nTransaction
Value:\n", t.value)
signature = t.sign_transaction()
print("\nSignature:\n", signature)
```

OUTPUT:



```
Transaction Recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100be0978593e24a777060c9b
  95c383a53f3e3213905fc67538a37f15e9c2c4bd6e0eb667999074079928c3e3d7f0636f0d7e9f8
  cc0bd7a38da76342c612c103fe43a603e97c602d1f2a0dbbe794ab983aa1d1062d70485c0e0c734
  3c265e90ae4094c6ede9037b0d38af530b27914df1a08c339ec4bc76de28b2dbe5cd6a0e3ebf020
  3010001

Signature:
 a3d6c2635b55cd891bc43dc63b137bfb10eaa68dd7ce276f8e098367cc83195f5a508fa2e4e2c74a
  341e143d38432b36e4dc240f5f4b17d07d5be3ebf67a59f42583c215e9d78059091580629eee052
  985b2bf355ab58eae226b41dbfaa9690bfce61745392e9f172173ce5a2f12040b753610468ca4a3
  fccb1437968a11fc00
```

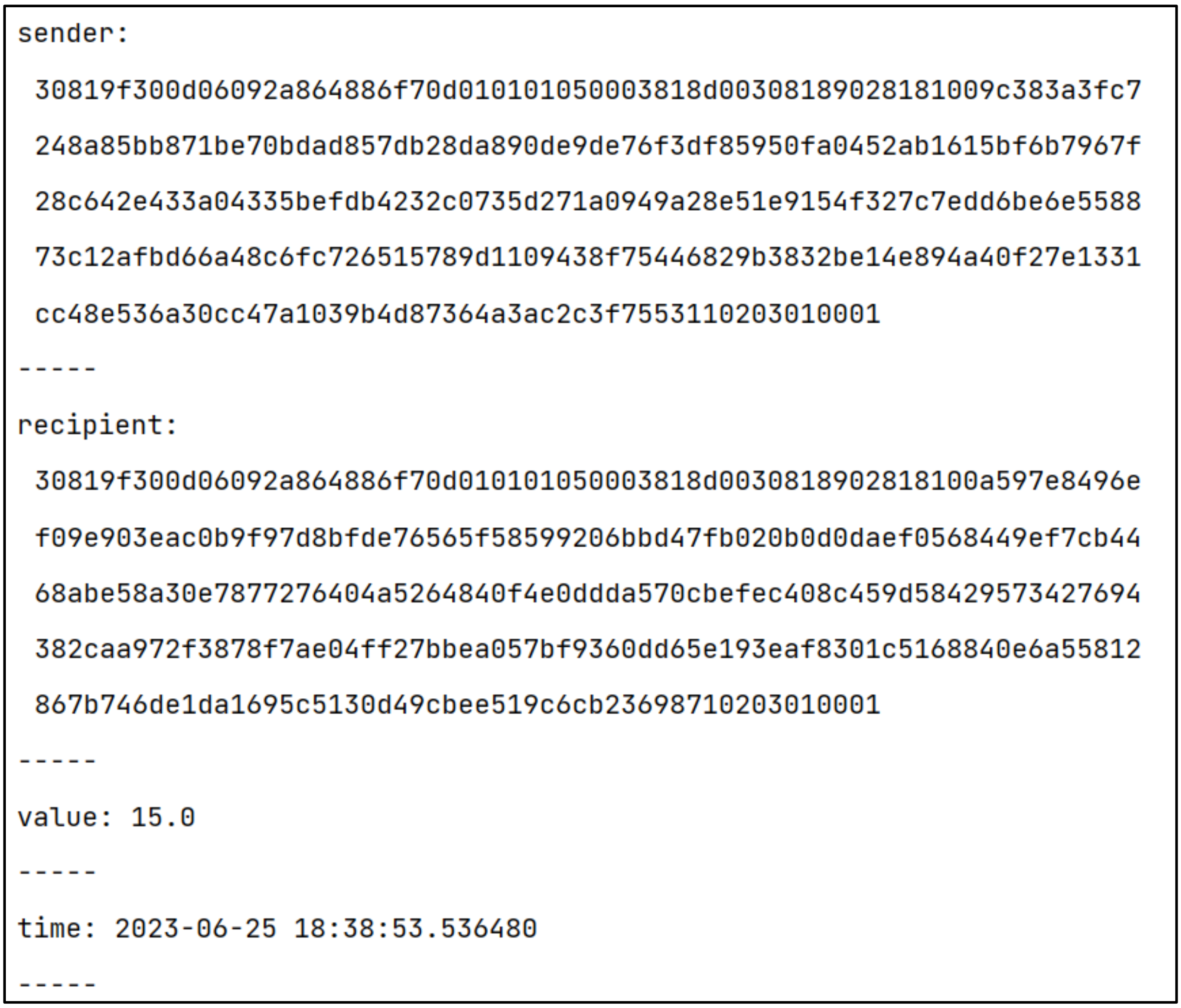
[C] Create multiple transactions and display them.

CODE:

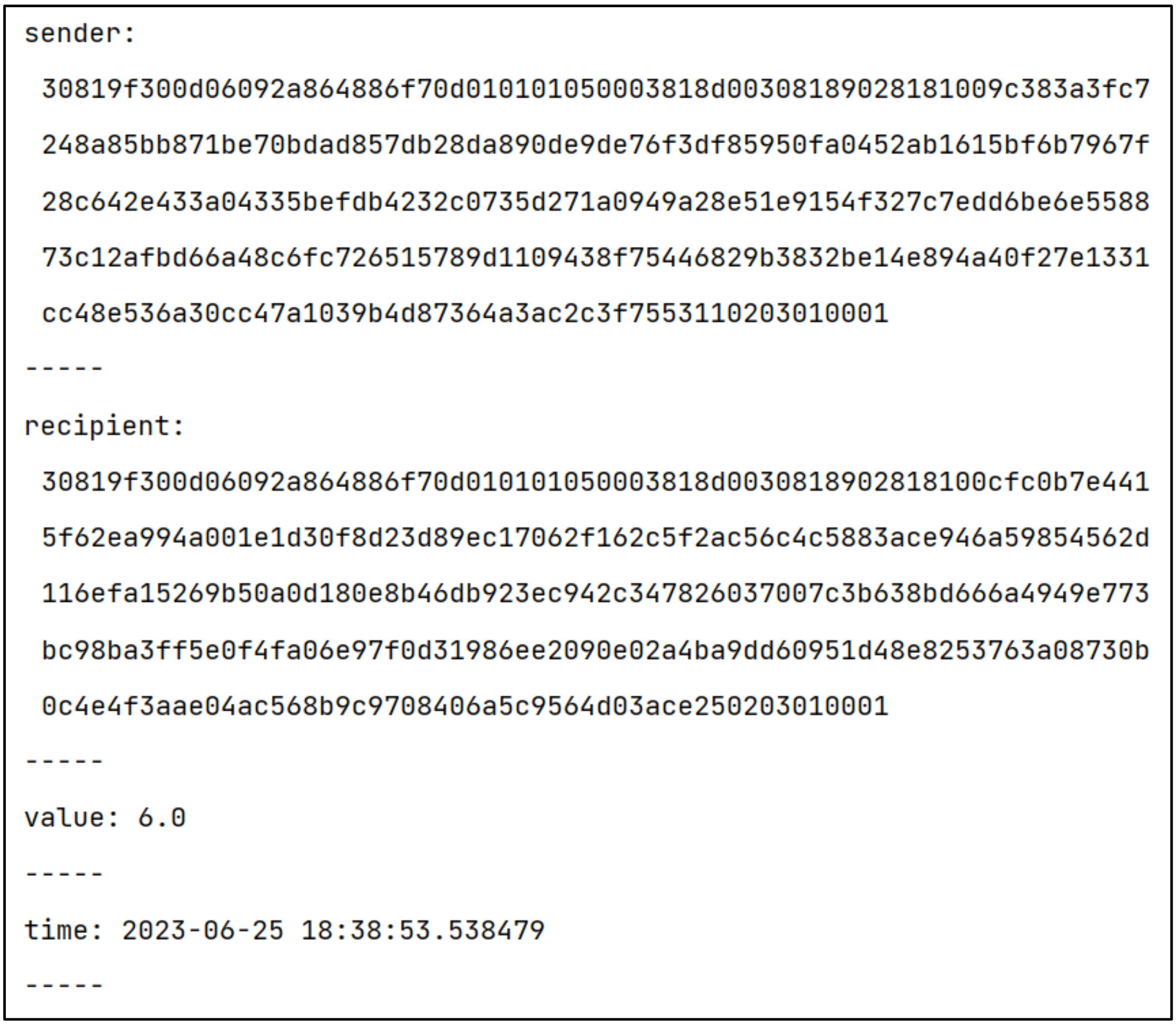
```
import binascii
import collections
import datetime
from Crypto.Hash import SHA
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto import Random



class Client:
    def __init__(self):
        random = Random.new().read
        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("a
scii")



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

    def to_dict(self):
        identity = "Genesis" if self.sender == "Genesis" else self.sender.identity
        return collections.OrderedDict(
            {
                "sender": identity,
                "recipient": self.recipient,
                "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 display_transaction(transaction):
        # for transaction in transactions:
        dict = transaction.to_dict()
        print("sender: " + dict['sender'])
        print('-----')
        print("recipient: " + dict['recipient'])
        print('-----')
        print("value: " + str(dict['value']))
        print('-----')
        print("time: " + str(dict['time']))
        print('-----')



UDIT = Client()
UGC = Client()
AICTE = Client()
MU = Client()
```

```
t1 = Transaction(UDIT, UGC.identity, 15.0)
t1.sign_transaction()
transactions = [t1]
t2 = Transaction(UDIT, AICTE.identity, 6.0)
t2.sign_transaction()
transactions.append(t2)
t3 = Transaction(UGC, MU.identity, 2.0)
t3.sign_transaction()
transactions.append(t3)
t4 = Transaction(AICTE, UGC.identity, 4.0)
t4.sign_transaction()
transactions.append(t4)
for transaction in transactions:
    Transaction.display_transaction(transaction)
    print("      ")
```

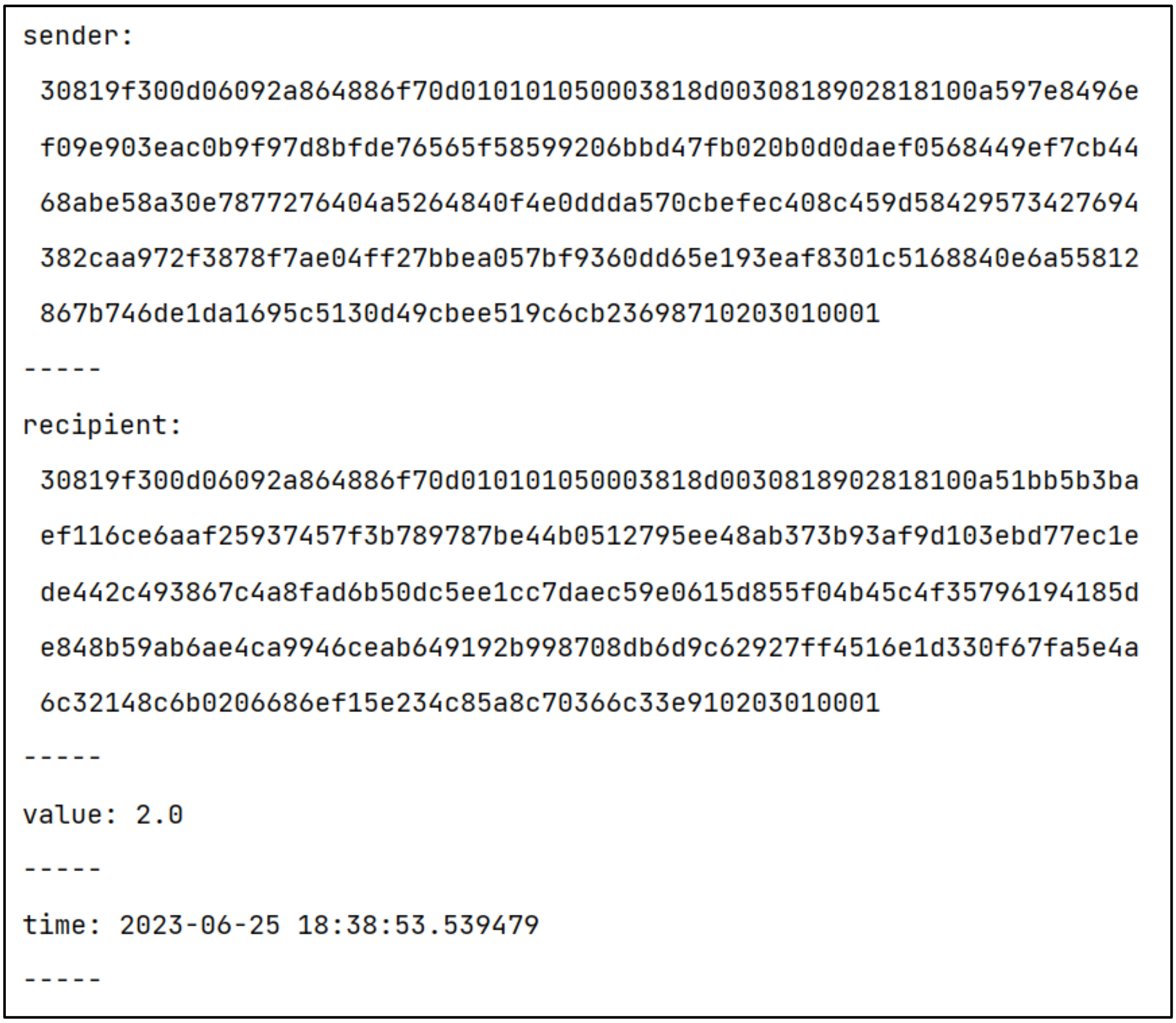
OUTPUT:



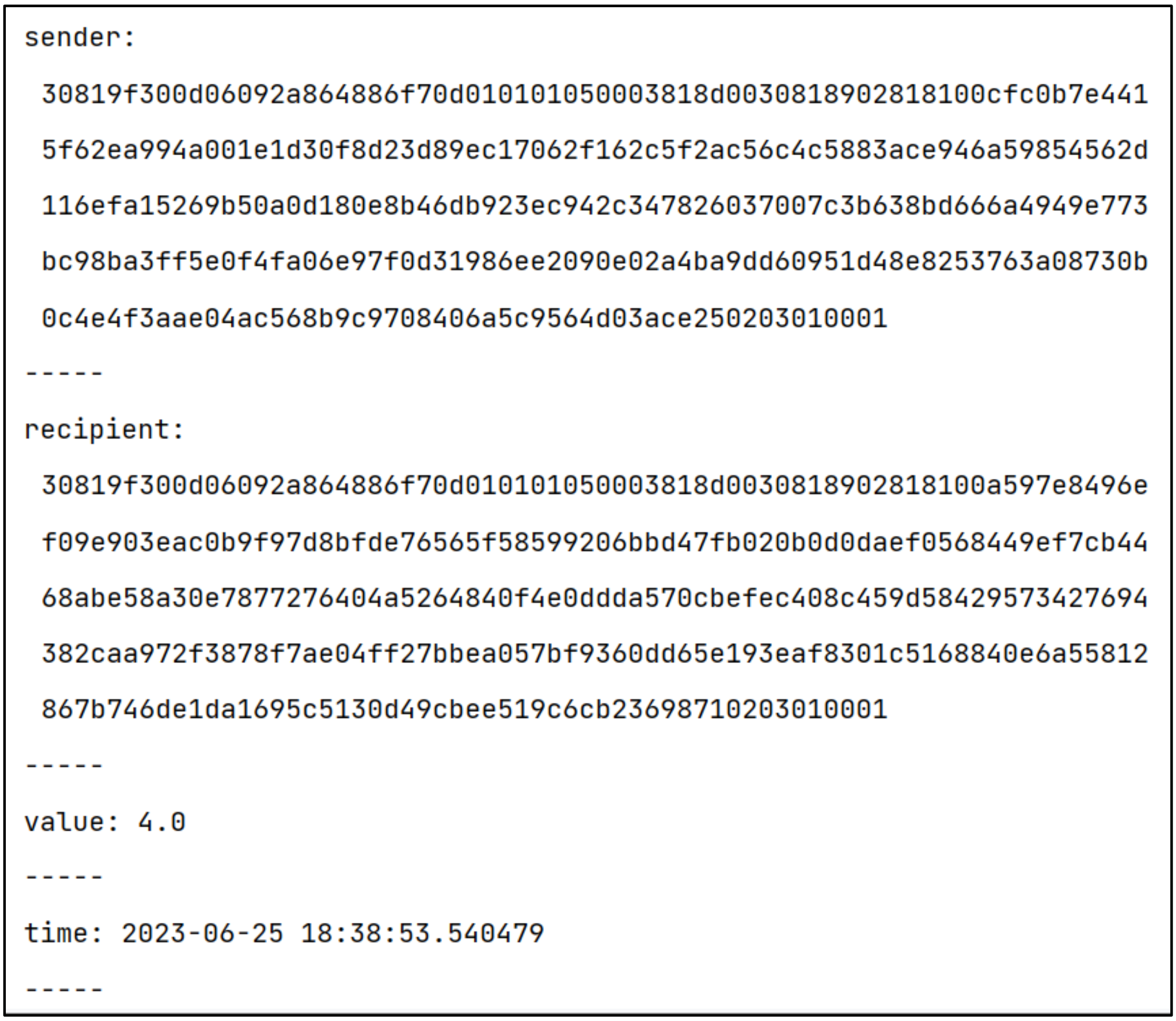
```
sender:
 30819f300d06092a864886f70d010101050003818d0030818902818100 9c383a3fc7
```



```
sender:
 30819f300d06092a864886f70d010101050003818d0030818902818100 9c383a3fc7
 248a85bb871be70bdad857db28da890de9de76f3df85950fa0452ab1615bf6b7967f
 28c642e433a04335befdb4232c0735d271a0949a28e51e9154f327c7edd6be6e5588
 73c12afbd66a48c6fc726515789d1109438f75446829b3832be14e894a40f27e1331
 cc48e536a30cc47a1039b4d87364a3ac2c3f7553110203010001
-----
recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100cfc0b7e441
 5f62ea994a001e1d30f8d23d89ec17062f162c5f2ac56c4c5883ace946a59854562d
 116efa15269b50a0d180e8b46db923ec942c347826037007c3b638bd666a4949e773
 bc98ba3ff5e0f4fa06e97f0d31986ee2090e02a4ba9dd60951d48e8253763a08730b
 0c4e4f3aae04ac568b9c9708406a5c9564d03ace250203010001
-----
value: 6.0
-----
time: 2023-06-25 18:38:53.538479
-----
```



```
sender:
 30819f300d06092a864886f70d010101050003818d0030818902818100a597e8496e
 f09e903eac0b9f97d8bfde76565f58599206bbd47fb020b0d0daef0568449ef7cb44
 68abe58a30e7877276404a5264840f4e0ddda570cbefec408c459d58429573427694
 382caa972f3878f7ae04ff27bbea057bf9360dd65e193eaf8301c5168840e6a55812
 867b746de1da1695c5130d49cbee519c6cb23698710203010001
-----
recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100a51bb5b3ba
 ef116ce6aaf25937457f3b789787be44b0512795ee48ab373b93af9d103ebd77ec1e
 de442c493867c4a8fad6b50dc5ee1cc7daec59e0615d855f04b45c4f35796194185d
 e848b59ab6ae4ca9946ceab649192b998708db6d9c62927ff4516e1d330f67fa5e4a
 6c32148c6b0206686ef15e234c85a8c70366c33e910203010001
-----
value: 2.0
-----
time: 2023-06-25 18:38:53.539479
-----
```



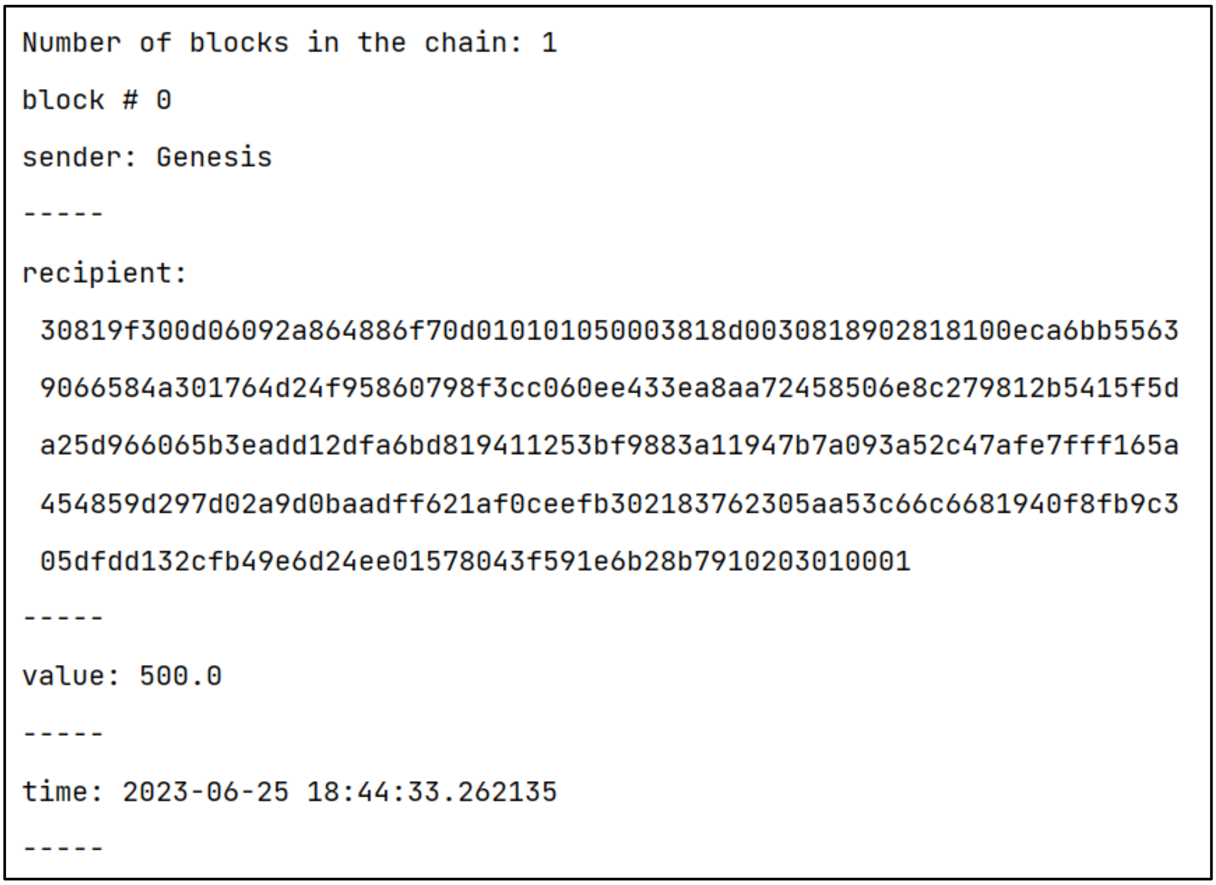
```
sender:
 30819f300d06092a864886f70d010101050003818d0030818902818100cfc0b7e441
 5f62ea994a001e1d30f8d23d89ec17062f162c5f2ac56c4c5883ace946a59854562d
 116efa15269b50a0d180e8b46db923ec942c347826037007c3b638bd666a4949e773
 bc98ba3ff5e0f4fa06e97f0d31986ee2090e02a4ba9dd60951d48e8253763a08730b
 0c4e4f3aae04ac568b9c9708406a5c9564d03ace250203010001
-----
recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100a597e8496e
 f09e903eac0b9f97d8bfde76565f58599206bbd47fb020b0d0daef0568449ef7cb44
 68abe58a30e7877276404a5264840f4e0ddda570cbefec408c459d58429573427694
 382caa972f3878f7ae04ff27bbea057bf9360dd65e193eaf8301c5168840e6a55812
 867b746de1da1695c5130d49cbee519c6cb23698710203010001
-----
value: 4.0
-----
time: 2023-06-25 18:38:53.540479
-----
```

[D] Create a blockchain, a genesis block and execute it.

CODE:

```
import binascii
import collections
import datetime
from Crypto.Hash import SHA
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto import Random


class Client:
    def __init__(self):
        random = Random.new().read
        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("a
scii")


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

    def to_dict(self):
        identity = "Genesis" if self.sender == "Genesis" else
self.sender.identity
        return collections.OrderedDict(
            {
                "sender": identity,
                "recipient": self.recipient,
                "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 display_transaction(transaction):
        # for transaction in transactions:
        dict = transaction.to_dict()
        print("sender: " + dict['sender'])
        print('-----')
        print("recipient: " + dict['recipient'])
        print('-----')
        print("value: " + str(dict['value']))
        print('-----')
        print("time: " + str(dict['time']))
        print('-----')


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


def dump_blockchain(blocks):
    print(f"\nNumber of blocks in the chain: {len(blocks)}")
    for i, block in enumerate(blocks):
        print(f"block # {i}")
        for transaction in block.verified_transactions:
            Transaction.display_transaction(transaction)
            print("   ")
    print("      ")


UDIT = Client()
t0 = Transaction("Genesis", UDIT.identity, 500.0)
block0 = Block(UDIT)
block0.previous_block_hash = ""
NONCE = None
block0.verified_transactions.append(t0)
digest = hash(block0)
last_block_hash = digest
TPCoins = [block0]
dump_blockchain(TPCoins)
```

OUTPUT:



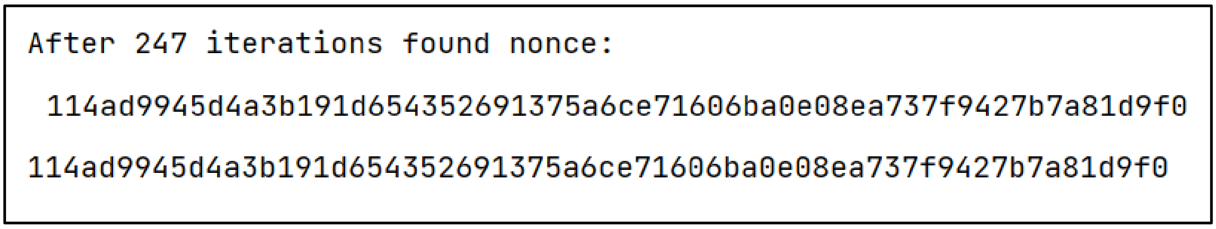
```
Number of blocks in the chain: 1

block # 0

sender: Genesis

-----

recipient:

 30819f300d06092a864886f70d010101050003818d0030818902818100eca6bb5563

 9066584a301764d24f95860798f3cc060ee433ea8aa72458506e8c279812b5415f5d

 a25d966065b3eadd12dfa6bd819411253bf9883a11947b7a093a52c47afe7fff165a

 454859d297d02a9d0baadff621af0ceefb302183762305aa53c66c6681940f8fb9c3

 05dfdd132cfb49e6d24ee01578043f591e6b28b7910203010001

-----

value: 500.0

-----

time: 2023-06-25 18:44:33.262135

-----
```

[E] Create a mining function and test it.

CODE:

```
import hashlib



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(f"After {str(i)} iterations found nonce: {digest}")
            return digest

print(mine("test message", 2))
```

OUTPUT:



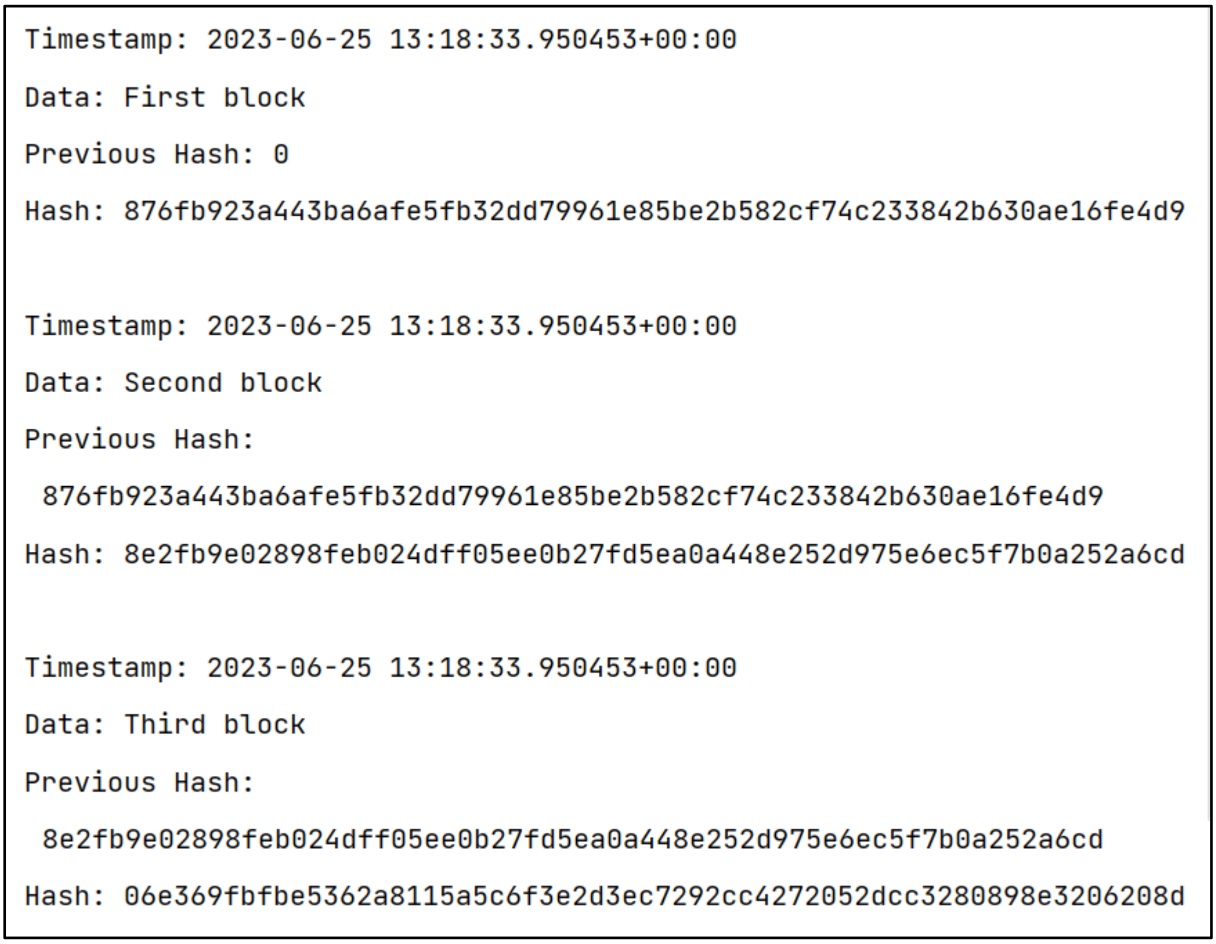
```
After 247 iterations found nonce:
 114ad9945d4a3b191d654352691375a6ce71606ba0e08ea737f9427b7a81d9f0
114ad9945d4a3b191d654352691375a6ce71606ba0e08ea737f9427b7a81d9f0
```

[F] Add 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()
# 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:



```
Timestamp: 2023-06-25 13:18:33.950453+00:00

Data: First block

Previous Hash: 0

Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9


Timestamp: 2023-06-25 13:18:33.950453+00:00

Data: Second block

Previous Hash:

 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9

Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd


Timestamp: 2023-06-25 13:18:33.950453+00:00

Data: Third block

Previous Hash:

 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd

Hash: 06e369fbfbe5362a8115a5c6f3e2d3ec7292cc4272052dcc3280898e3206208d
```

PRACTICAL 2

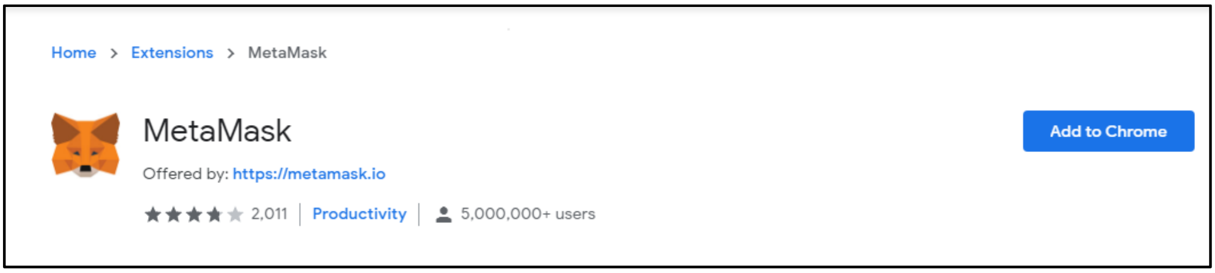
AIM: Install and configure Go Ethereum and the Mist browser. Develop and test a sample application.

Step 1: Go to Chrome Web Store Extensions Section.

Step 2: Search MetaMask.

Step 3: Check the number of downloads to make sure that the legitimate MetaMask is being installed, as hackers might try to make clones of it.

Step 4: Click the Add to Chrome button.



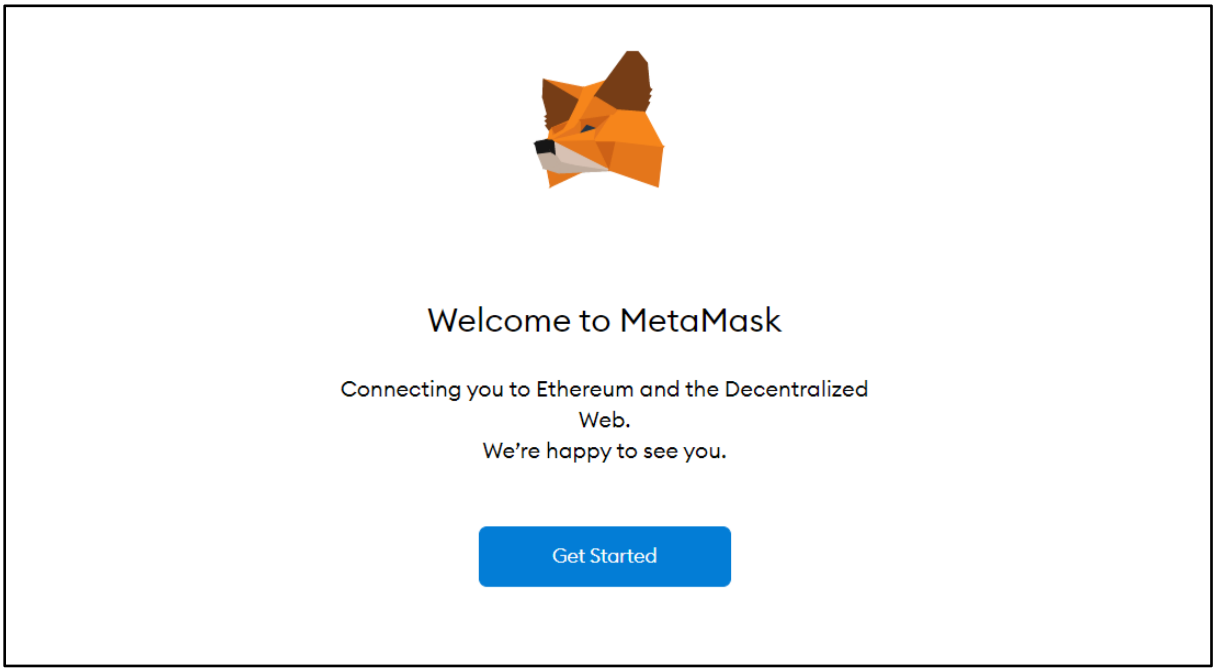
Home > Extensions > MetaMask

MetaMask

Offered by: https://metamask.io

★★★★☆ 2,011 | Productivity | 5,000,000+ users

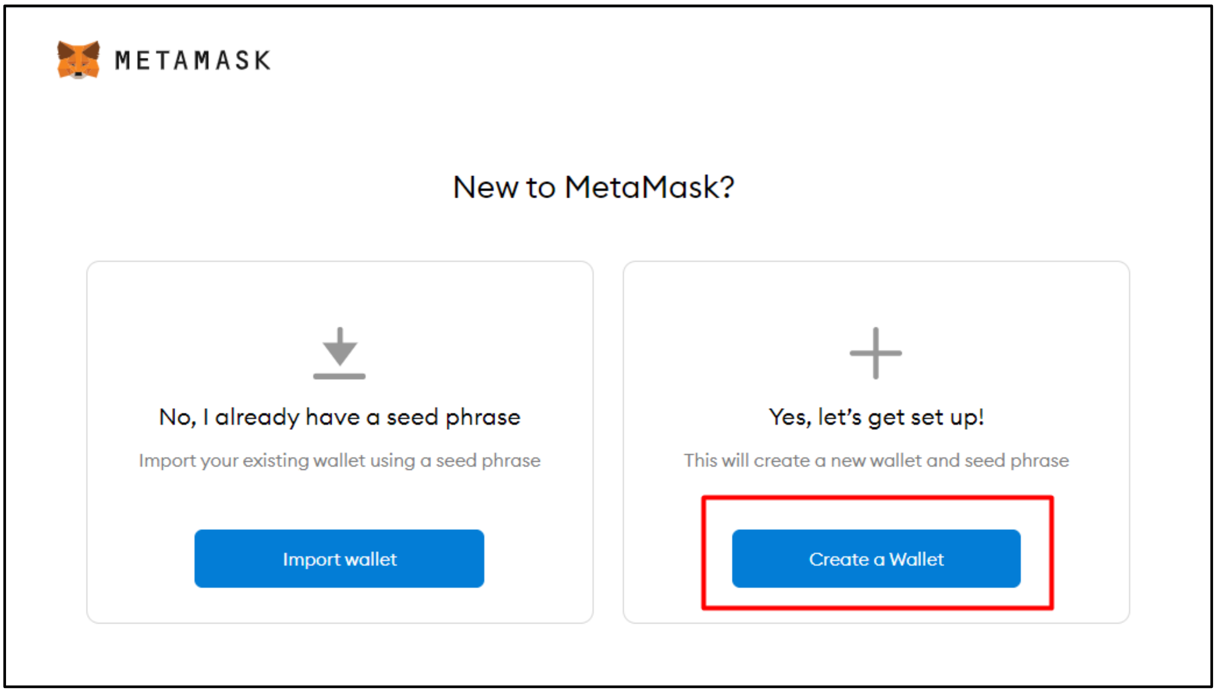
Step 5: Once installation is complete this page will be displayed. Click on the Get Started button.



Welcome to MetaMask

Connecting you to Ethereum and the Decentralized Web.
We're happy to see you.

Step 6: This is the first time creating a wallet, so click the Create a Wallet button. If there is already a wallet then import the already created using the Import Wallet button.



New to MetaMask?

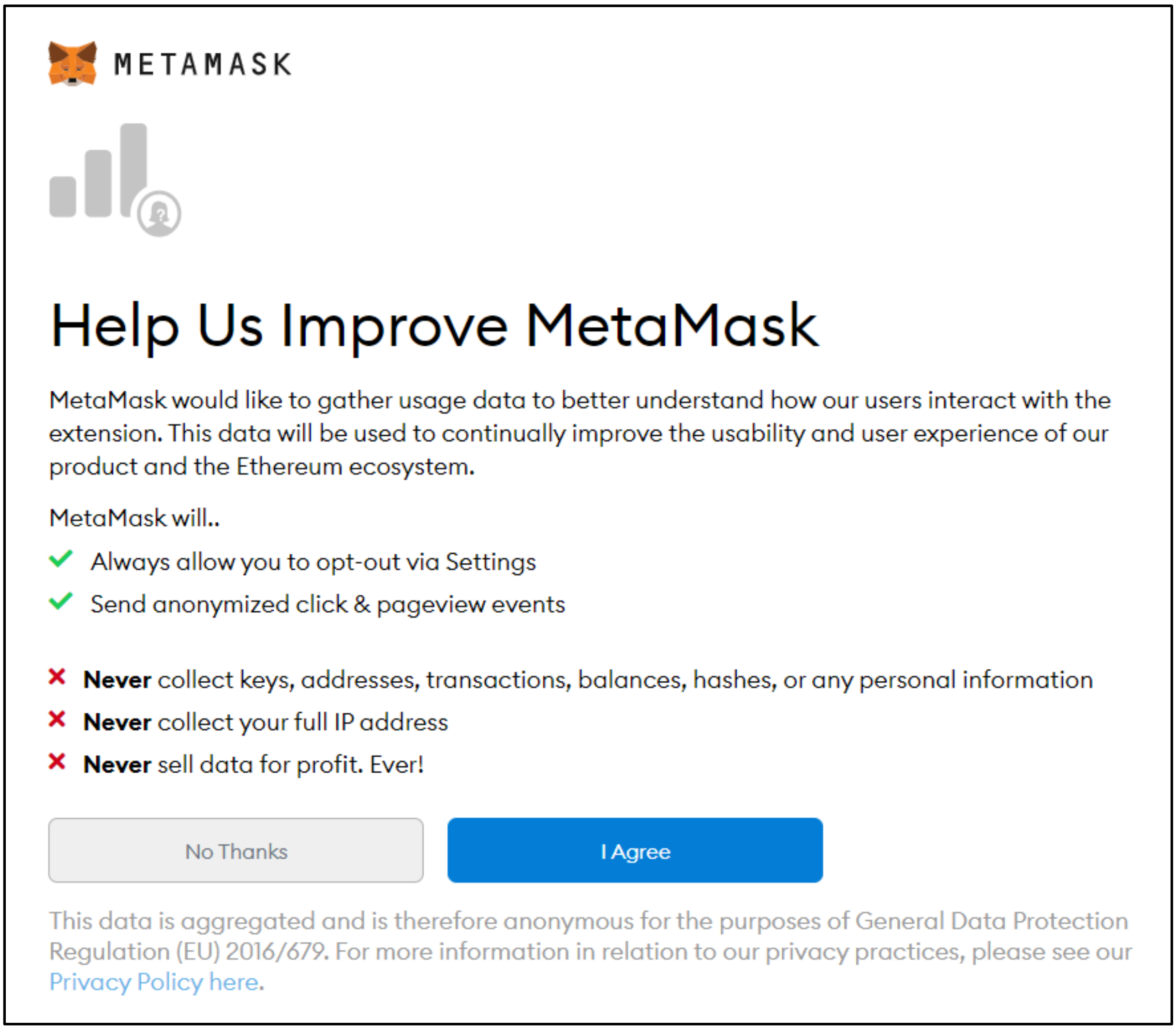
No, I already have a seed phrase

Import your existing wallet using a seed phrase

Yes, let's get set up!

This will create a new wallet and seed phrase

Step 7: Click I Agree button to allow data to be collected to help improve MetaMask or else click the No Thanks button. The wallet can still be created even if the user will click on the No Thanks button.



Help Us Improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.

MetaMask will..

✓ Always allow you to opt-out via Settings

✓ Send anonymized click & pageview events

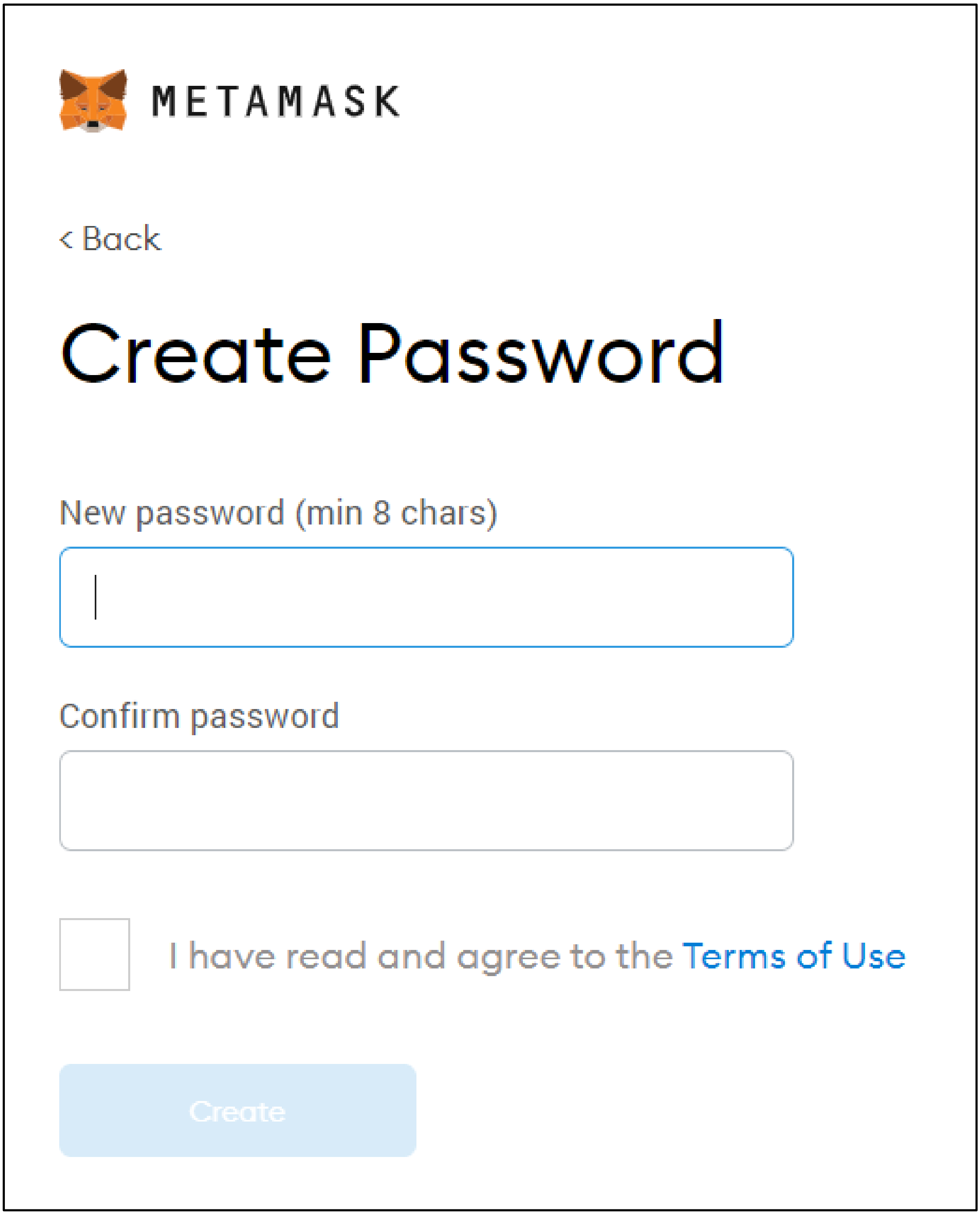
✗ **Never** collect keys, addresses, transactions, balances, hashes, or any personal information

✗ **Never** collect your full IP address

✗ **Never** sell data for profit. Ever!

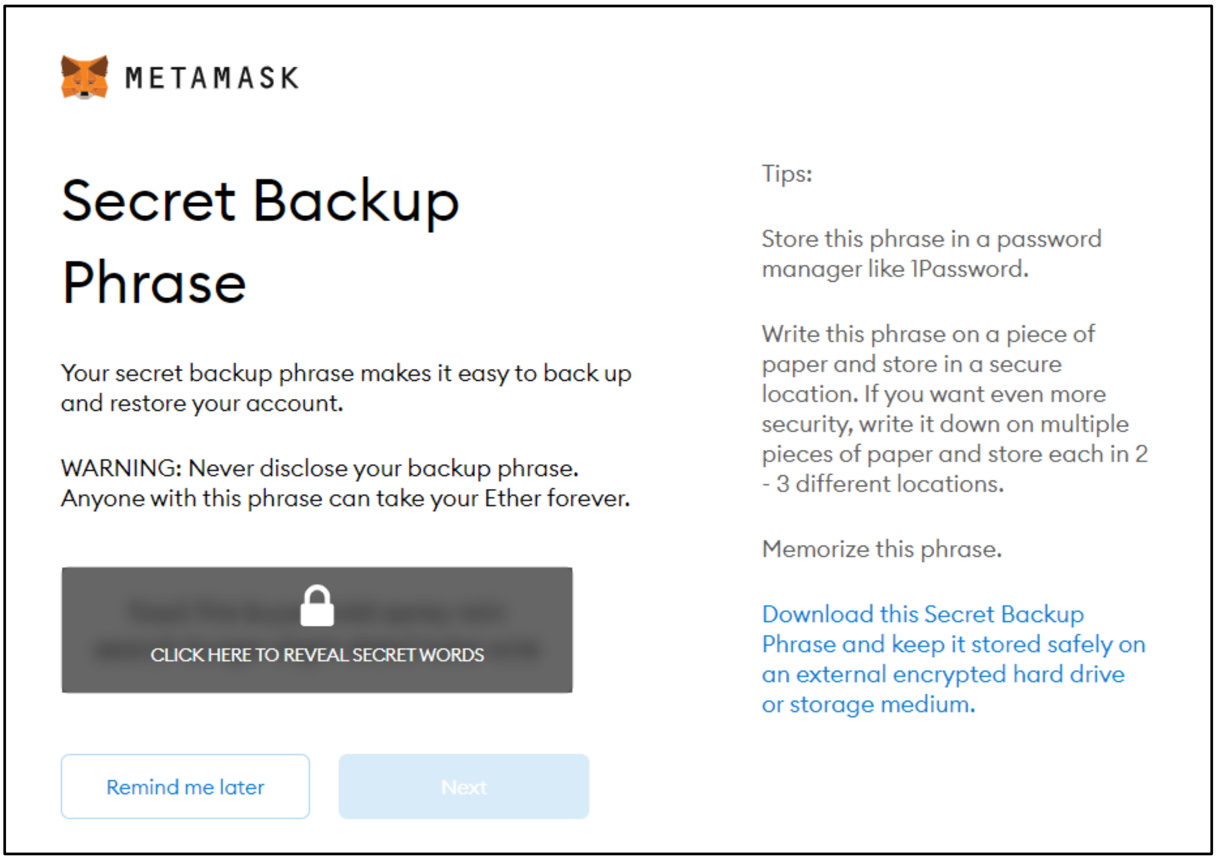
This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679. For more information in relation to our privacy practices, please see our Privacy Policy here.

Step 8: Create a password for your wallet. This password is to be entered every time the browser is launched and wants to use MetaMask. A new password needs to be created if chrome is uninstalled or if there is a switching of browsers. In that case, go through the Import Wallet button. This is because MetaMask stores the keys in the browser. Agree to Terms of Use.


METAMASK
< Back
Create Password
New password (min 8 chars)
Confirm password
□ I have read and agree to the Terms of Use
Create

Step 9: Click on the dark area which says Click here to reveal secret words to get your secret phrase.

Step 10: This is the most important step. Back up your secret phrase properly. Do not store your secret phrase on your computer. Please read everything on this screen until you understand it completely before proceeding. The secret phrase is the only way to access your wallet if you forget your password. Once done click the Next button.



METAMASK

Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.

Tips:

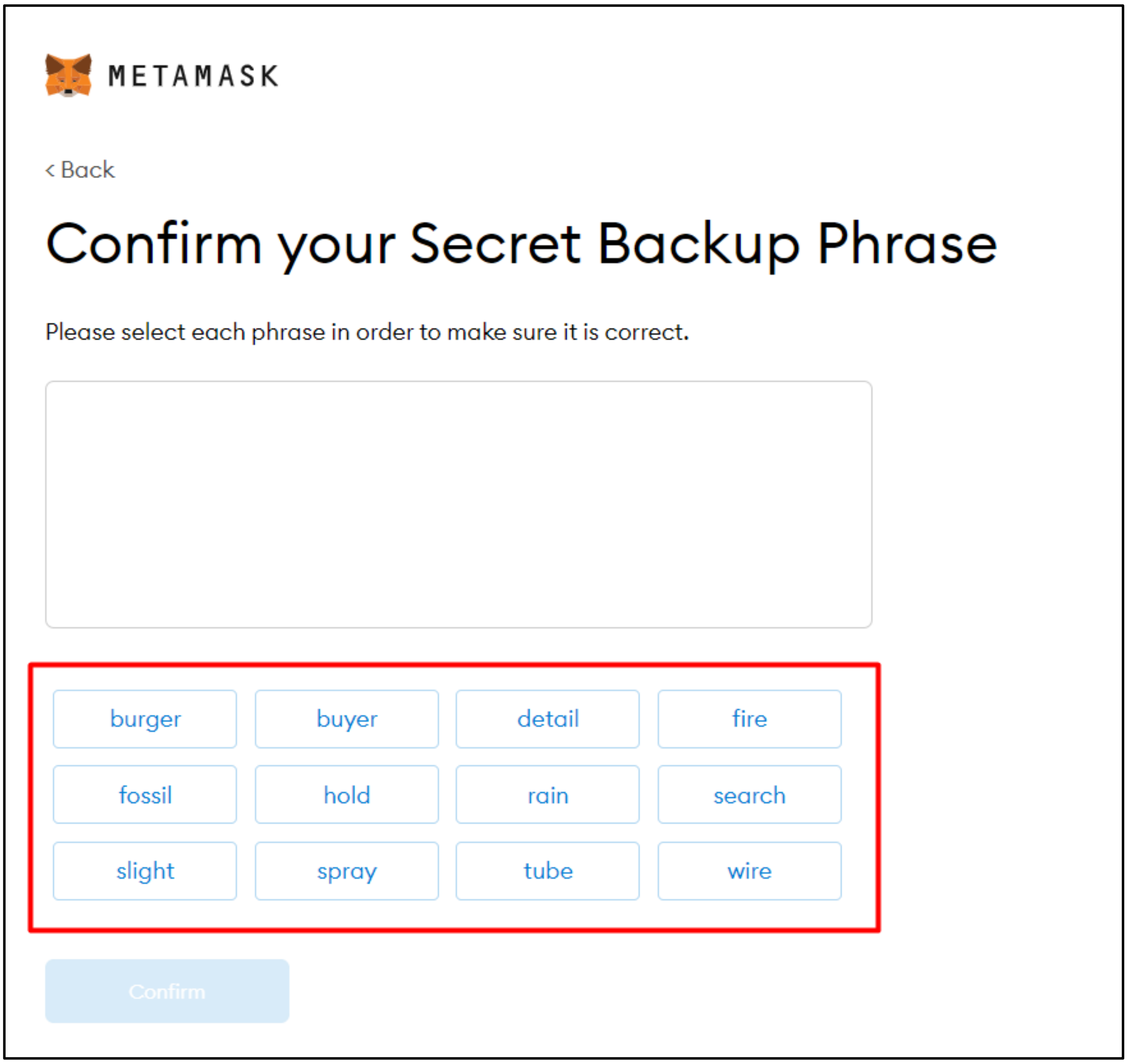
Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

Download this Secret Backup Phrase and keep it stored safely on an external encrypted hard drive or storage medium.

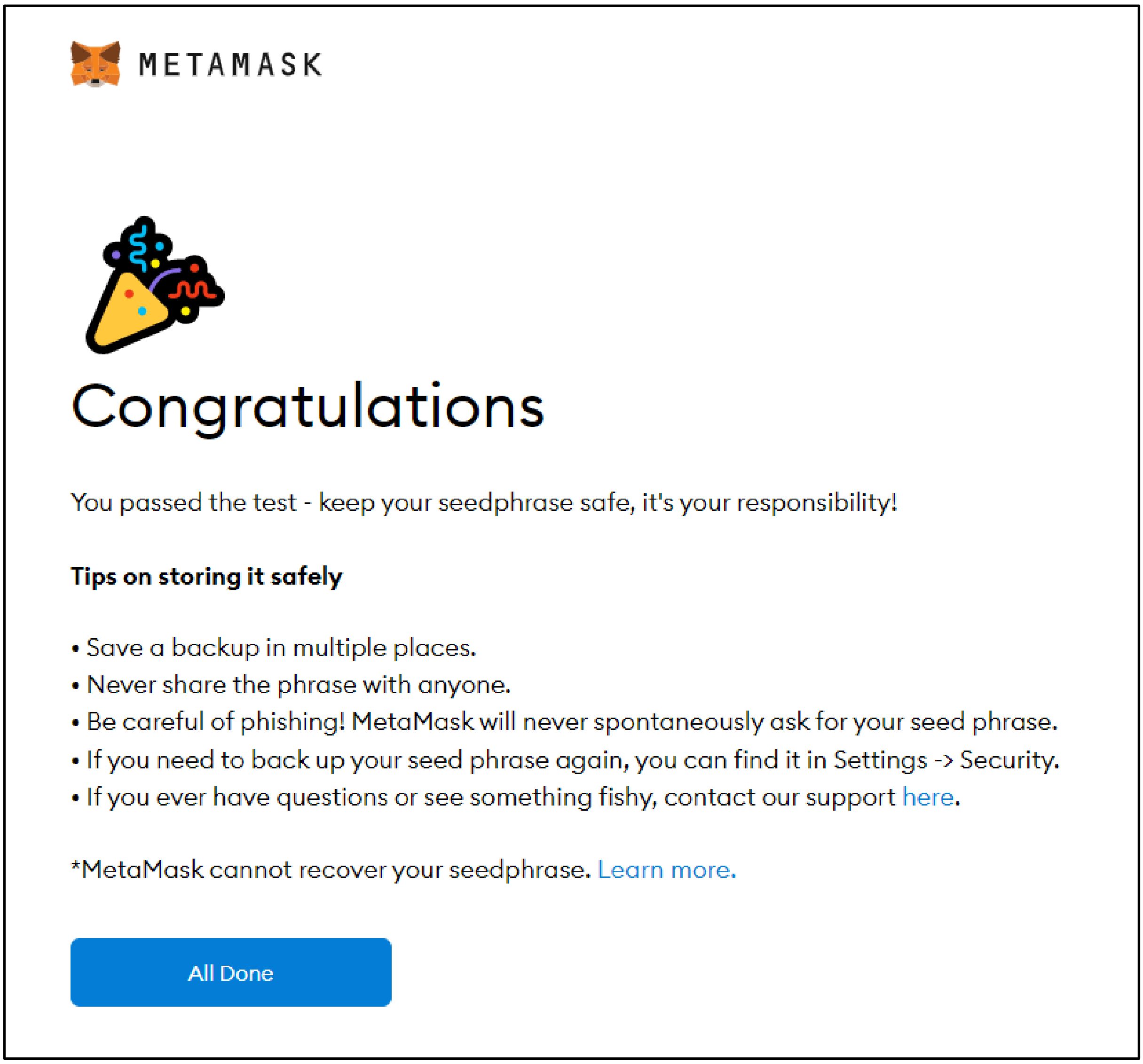
Step 11: Click the buttons respective to the order of the words in your seed phrase. In other words, type the seed phrase using the button on the screen. If done correctly the Confirm button should turn blue.



Confirm your Secret Backup Phrase

Please select each phrase in order to make sure it is correct.

Step 12: Click the Confirm button. Please follow the tips mentioned.



Congratulations

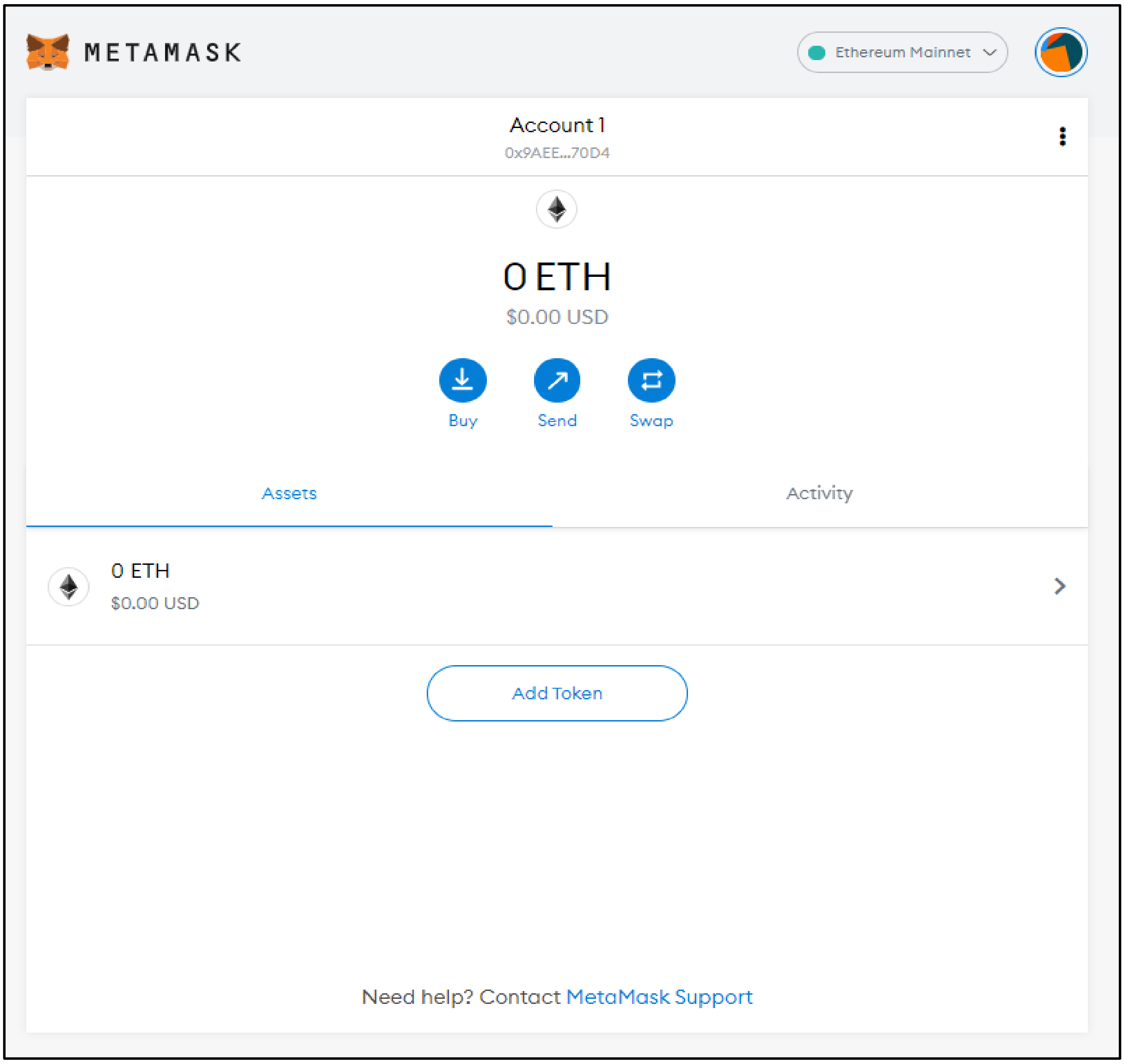
You passed the test - keep your seedphrase safe, it's your responsibility!

Tips on storing it safely

- Save a backup in multiple places.
- Never share the phrase with anyone.
- Be careful of phishing! MetaMask will never spontaneously ask for your seed phrase.
- If you need to back up your seed phrase again, you can find it in Settings -> Security.
- If you ever have questions or see something fishy, contact our support here.

*MetaMask cannot recover your seedphrase. Learn more.

Step 13: One can see the balance and copy the address of the account by clicking on the Account 1 area.



Account 1

0x9AEE...70D4

0 ETH

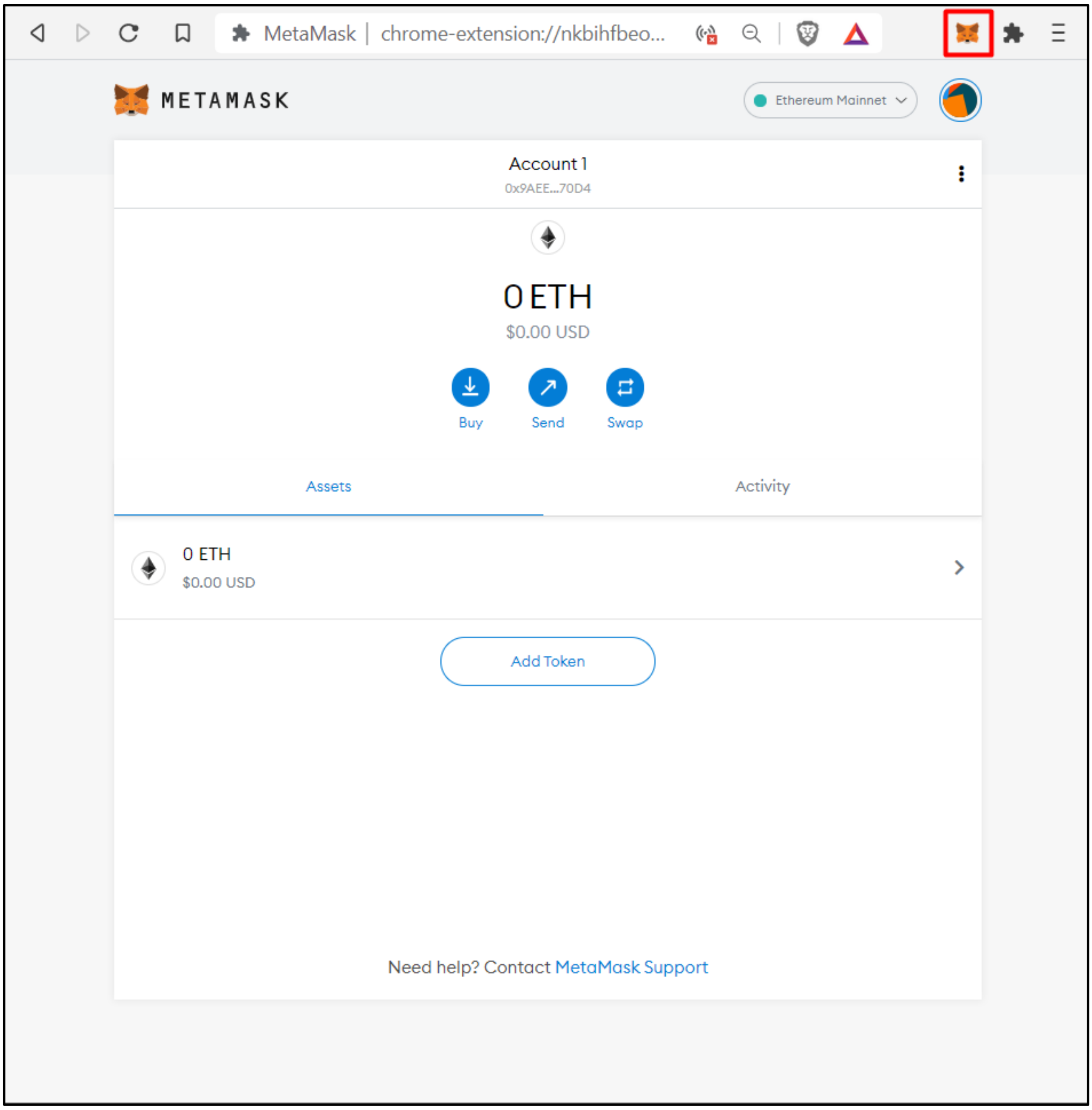
$0.00 USD

0 ETH

$0.00 USD

Need help? Contact MetaMask Support

Step 14: One can access MetaMask in the browser by clicking the Foxface icon on the top right. If the Foxface icon is not visible, then click on the puzzle piece icon right next to it.



PRACTICAL 3

AIM: Implement and demonstrate the use of the following in Solidity:

[A] Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.

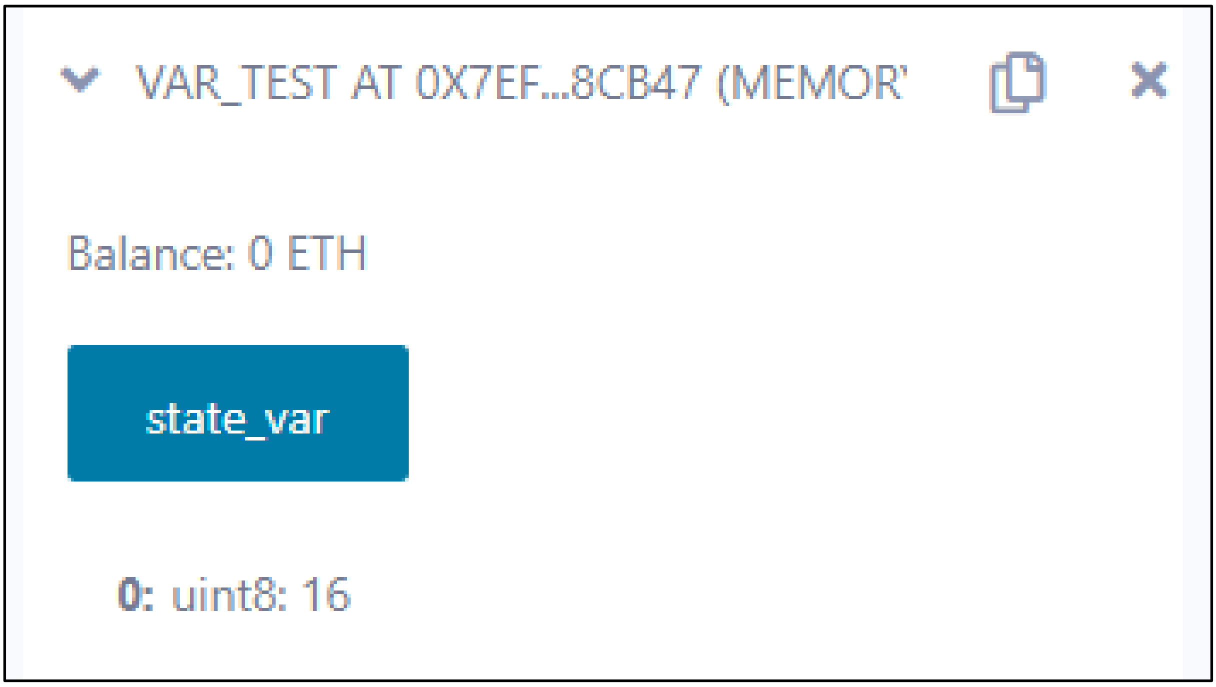
[I] Variable

[i] State Variable

CODE:

```
//Solidity program to demonstrate state variables
pragma solidity ^0.5.0;

// Creating a contract
contract var_Test
{
    // Declaring a state variable
    uint8 public state_var;

    // Defining a constructor
    constructor() public {
        state_var = 16;
    }
}
```

OUTPUT:



VAR_TEST AT 0X7EF...8CB47 (MEMOR'

Balance: 0 ETH

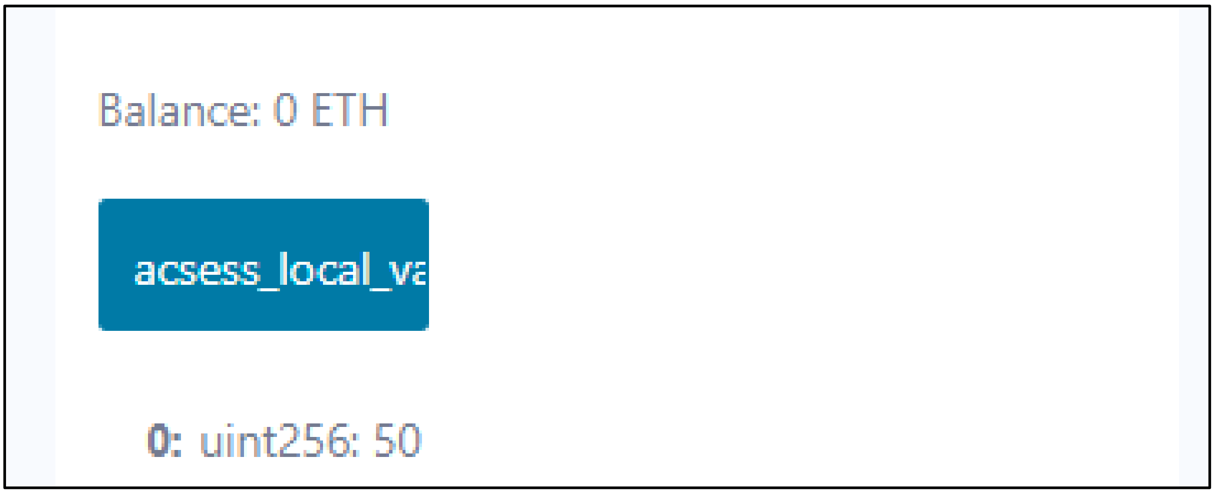
```
0: uint8: 16
```

[ii] Local Variable

CODE:

```
//Solidity program to demonstrate Local variables
pragma solidity ^0.5.0;
// Creating a contract
contract local_var_Test
{
    // Defining function to show the declaration and
    // scope of Local variables
    function acsess_local_variable() public pure returns(uint) {
        // Initializing Local variables
        uint a = 10;
        uint b = 40;
        uint sum = a + b;
        // Access the Local variable
        return sum;
    }
}
```

OUTPUT:



Balance: 0 ETH

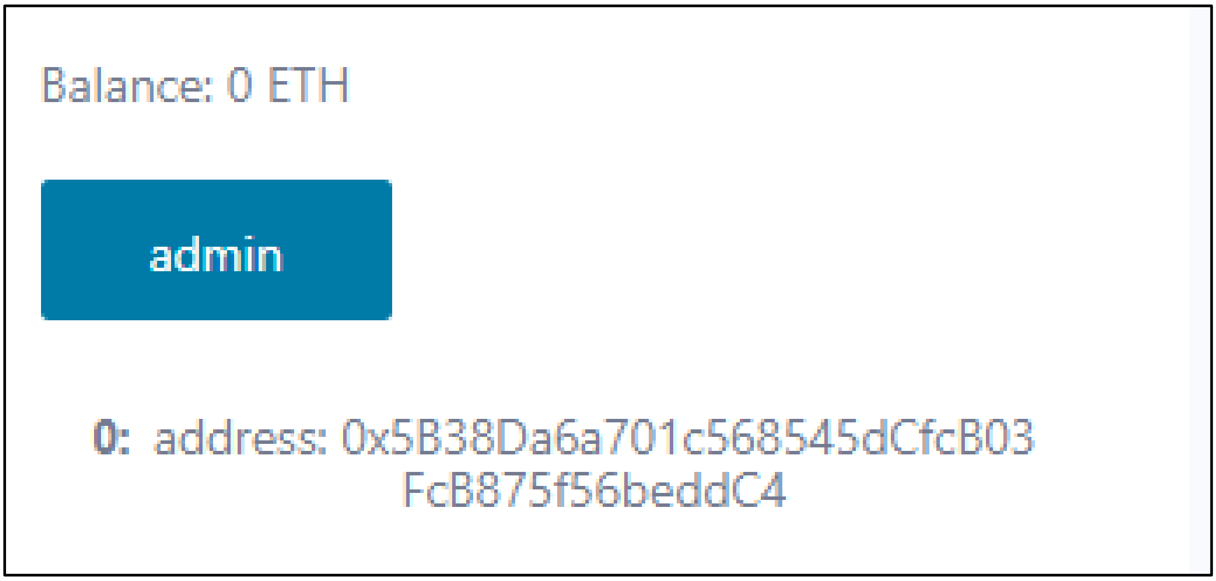
0: uint256: 50

[iii] Global Variable

CODE:

```
//Solidity program to show Global variables
pragma solidity ^0.5.0;
// Creating a contract
contract globalTest
{
    // Defining a variable
    address public admin;
    // Creating a constructor to
    // use Global variable
    constructor() public
    {
        admin = msg.sender;
    }
}
```

OUTPUT:



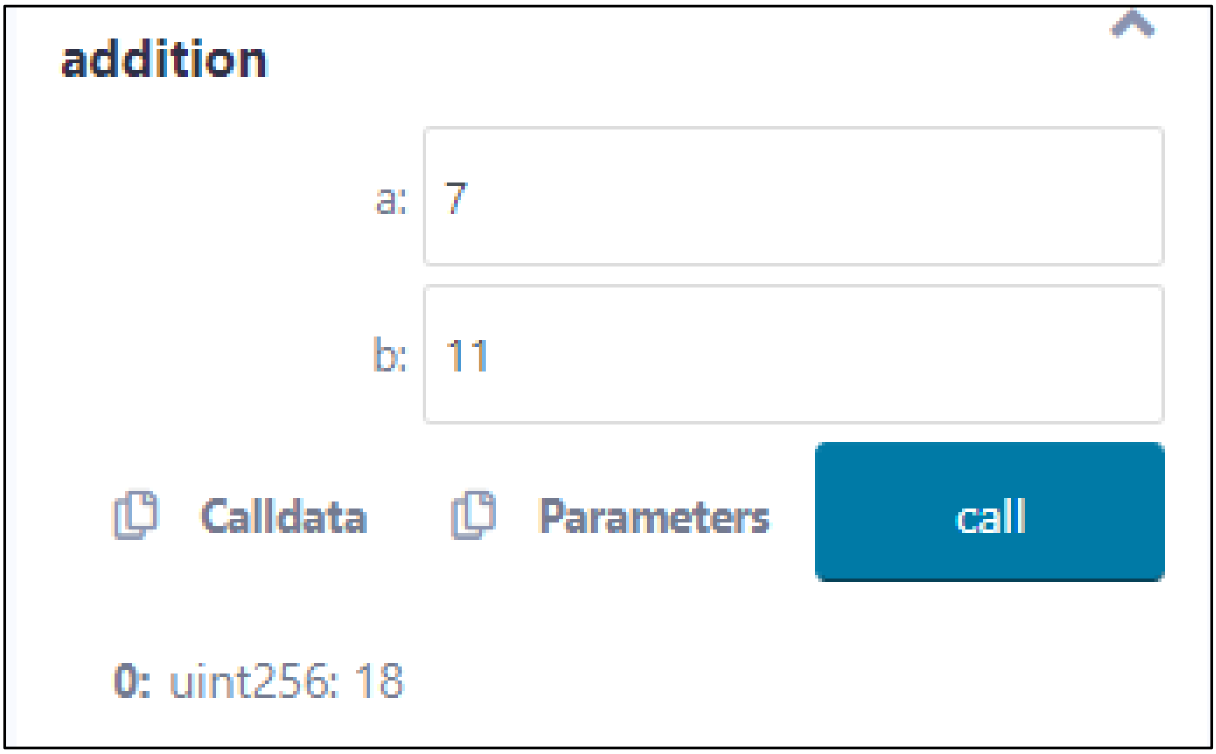
```
Balance: 0 ETH

admin

0: address: 0x5B38Da6a701c568545dCfcB03
           FcB875f56beddC4
```

[II] Operators

CODE:

```
pragma solidity ^0.4.0;

contract Operators {

    uint256 result = 0;

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

    function subtraction(uint256 a, uint256 b) public pure returns (uint256) {
        return a - b;
    }

    function division(uint256 a, uint256 b) public pure returns (uint256) {
        return a / b;
    }

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

OUTPUT:

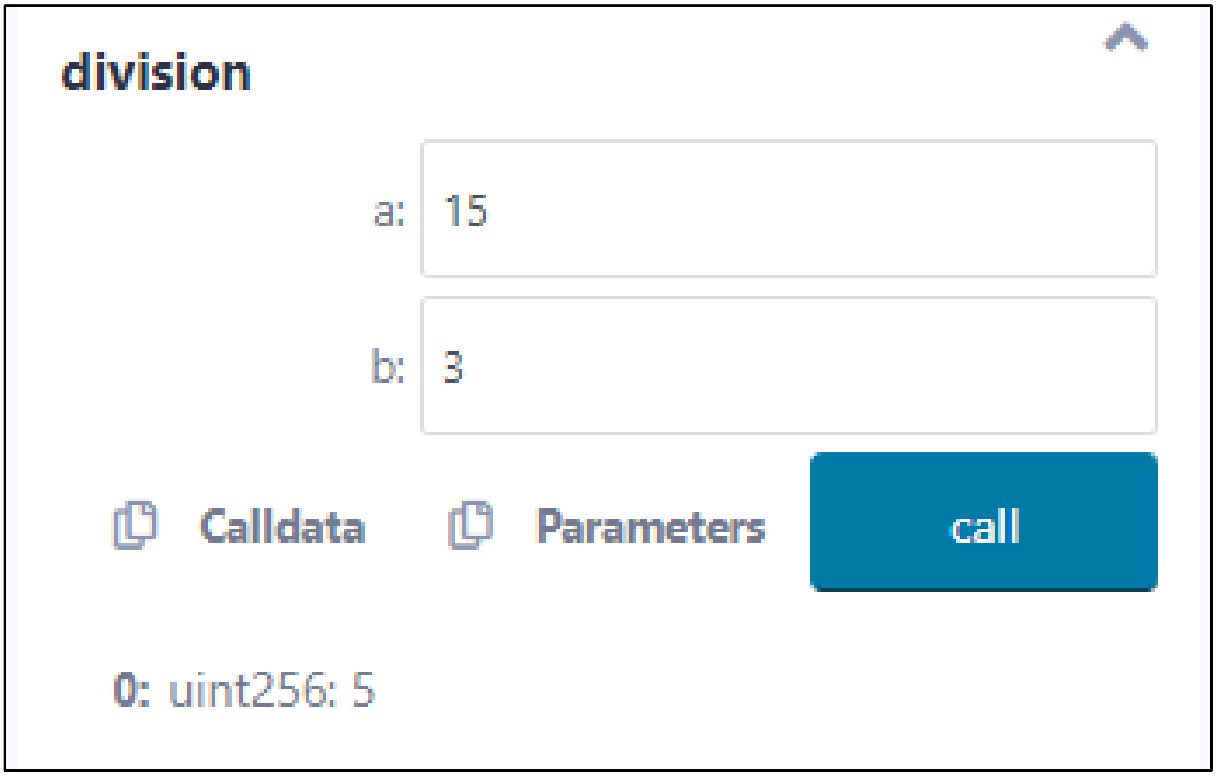


addition

a: 7

b: 11

0: uint256: 18

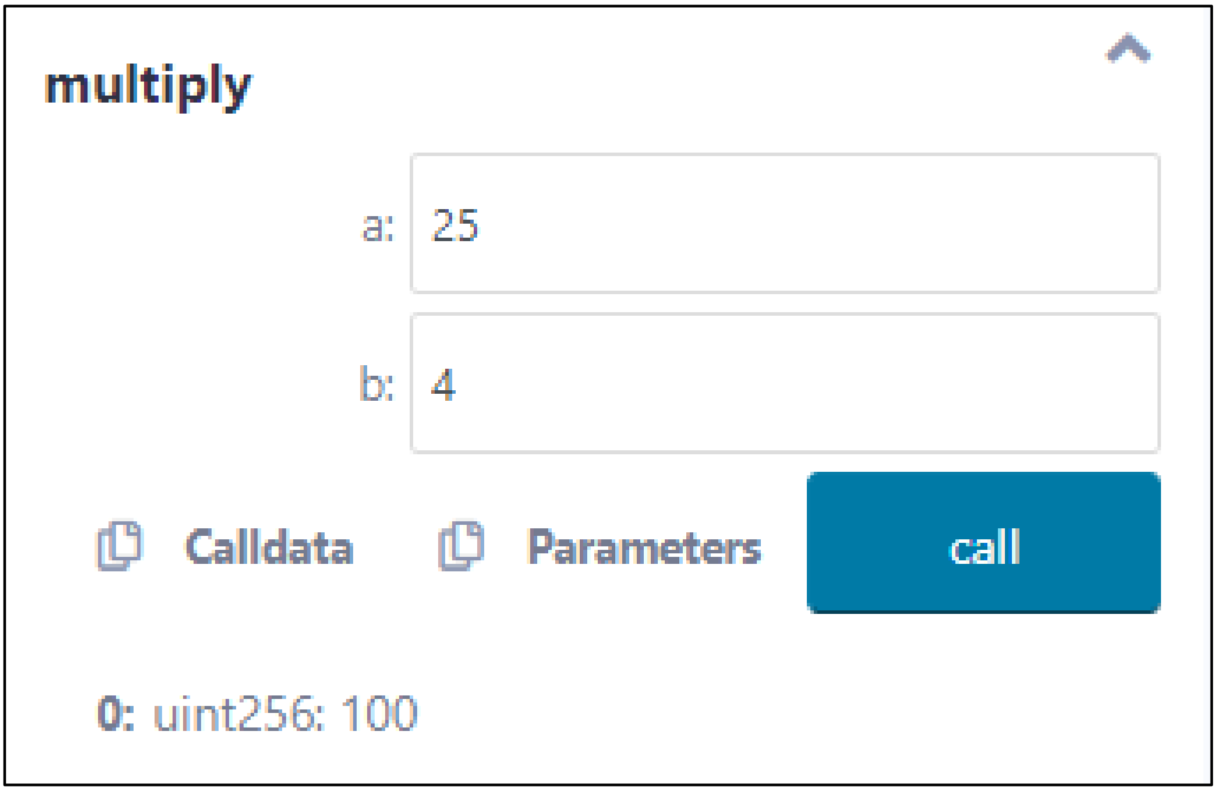


division

a: 15

b: 3

0: uint256: 5

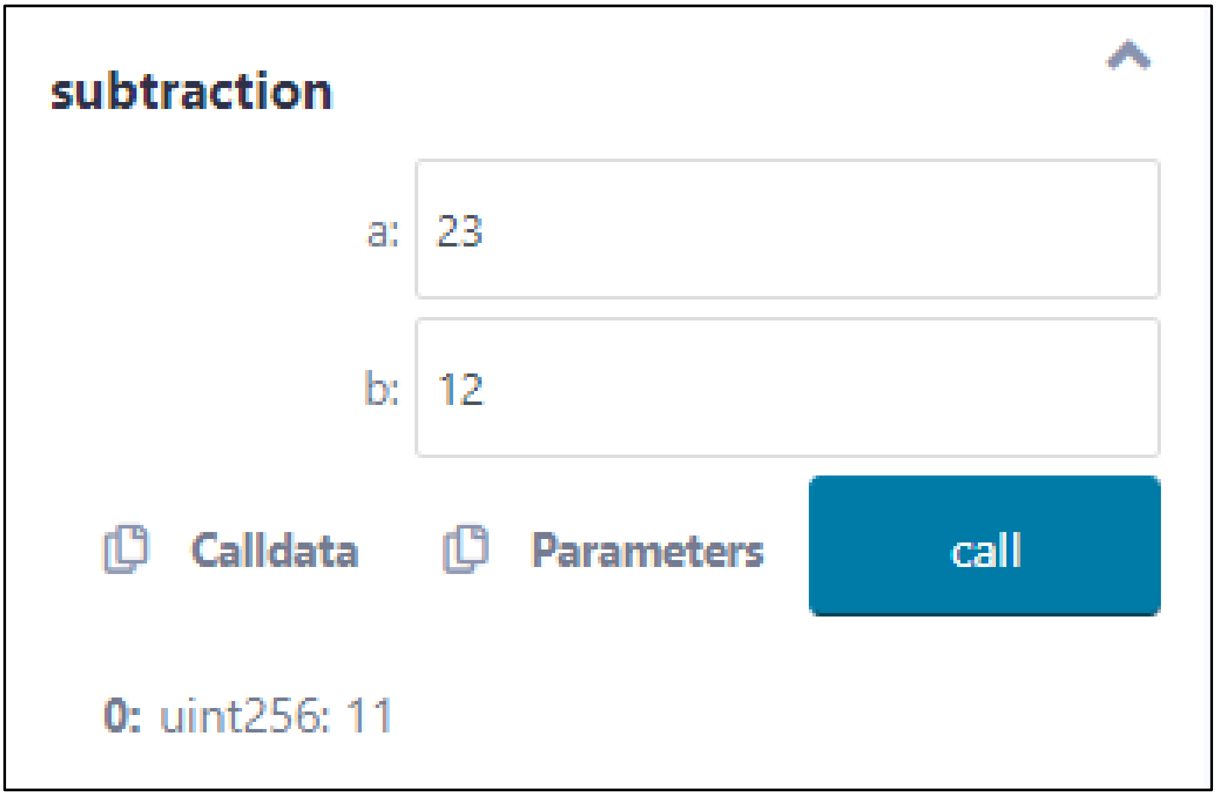


multiply

a: 25

b: 4

0: uint256: 100



subtraction

a: 23

b: 12

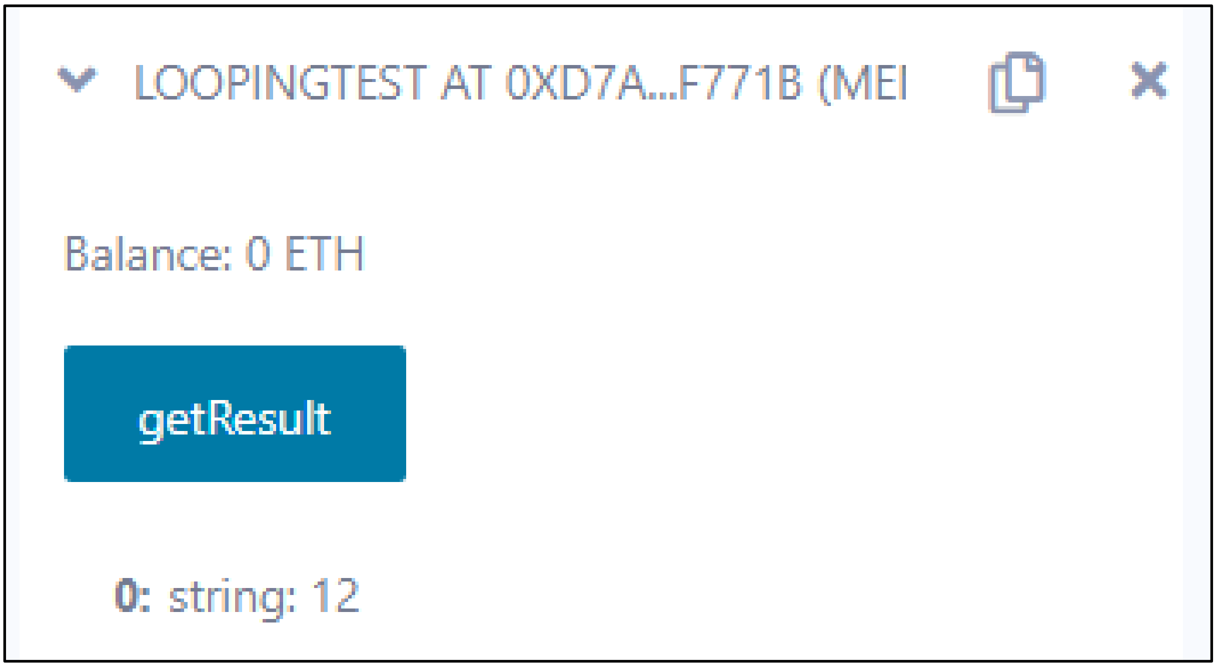
0: uint256: 11

[III] Loops

CODE:

```
pragma solidity ^0.5.0;

contract LoopingTest {
    uint256 storedData;

    constructor() public {
        storedData = 10;
    }

    function getResult() public pure returns (string memory) {
        uint256 a = 10;
        uint256 b = 2;
        uint256 result = a + b;
        return integerToString(result);
    }

    function integerToString(uint256 _i) internal pure returns (string memory) {
        if (_i == 0) {
            return "0";
        }
        uint256 j = 0;
        uint256 len;
        for (j = _i; j != 0; j /= 10) {
            //for loop example
            len++;
        }
        bytes memory bstr = new bytes(len);
        uint256 k = len - 1;
        while (_i != 0) {
            bstr[k--] = bytes1(uint8(48 + (_i % 10)));
            _i /= 10;
        }
        return string(bstr); //access local variable
    }
}
```

OUTPUT:



Balance: 0 ETH

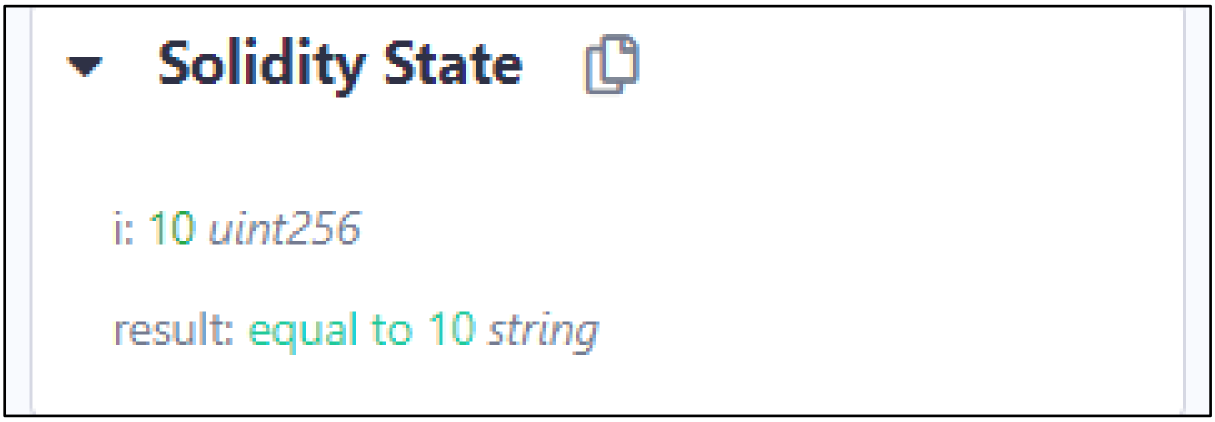
```
0: string: 12
```

[IV] Decision Making

CODE:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;

// Creating a contract
contract Types {

    // Declaring state variables
    uint256 i = 10;
    string result;

    function decision_making() public payable returns (string memory) {
        if (i < 10) {
            result = "less than 10";
        }
        else if (i == 10) {
            result = "equal to 10";
        }
        else {
            result = "greater than 10";
        }
        return result;
    }
}
```

OUTPUT:



```
i: 10 uint256

result: equal to 10 string
```

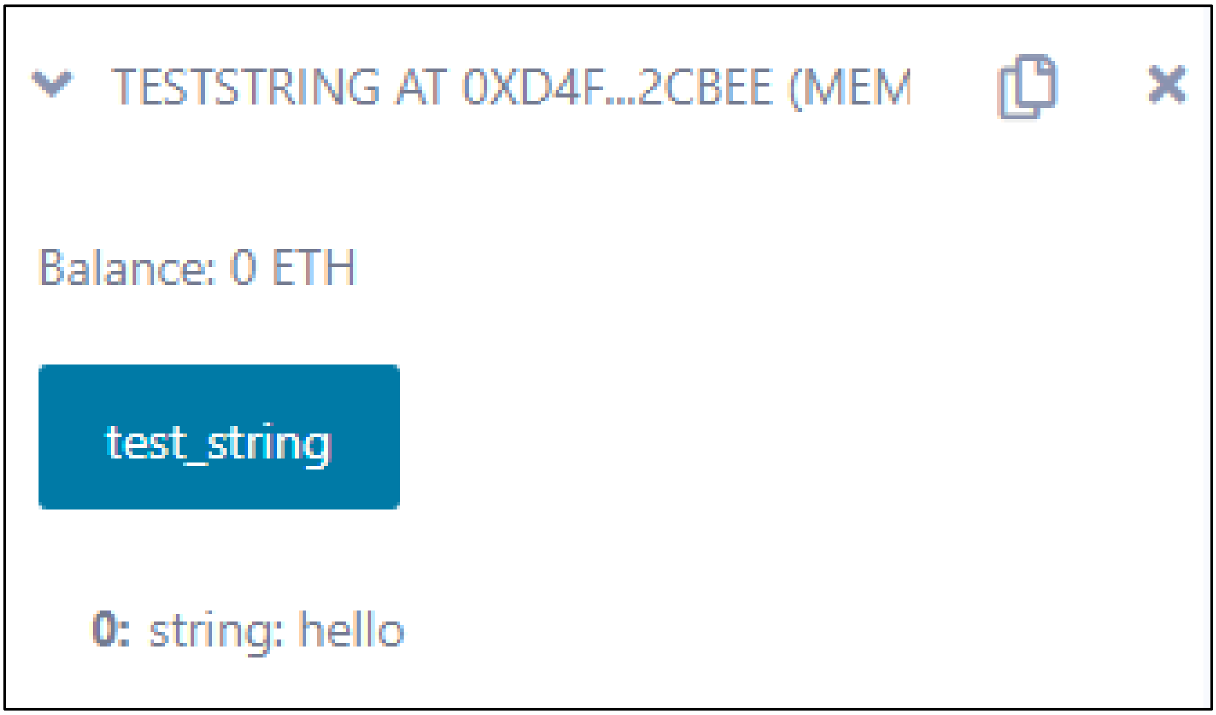
[V] Strings

[i] Double Quotes

CODE:

```
pragma solidity ^0.5.0;

contract testString
{
    function test_string() public pure returns (string memory)
    {
        string memory a = "hello";
        return a;
    }
}
```

OUTPUT:



TESTSTRING AT 0XD4F...2CBEE (MEM

Balance: 0 ETH

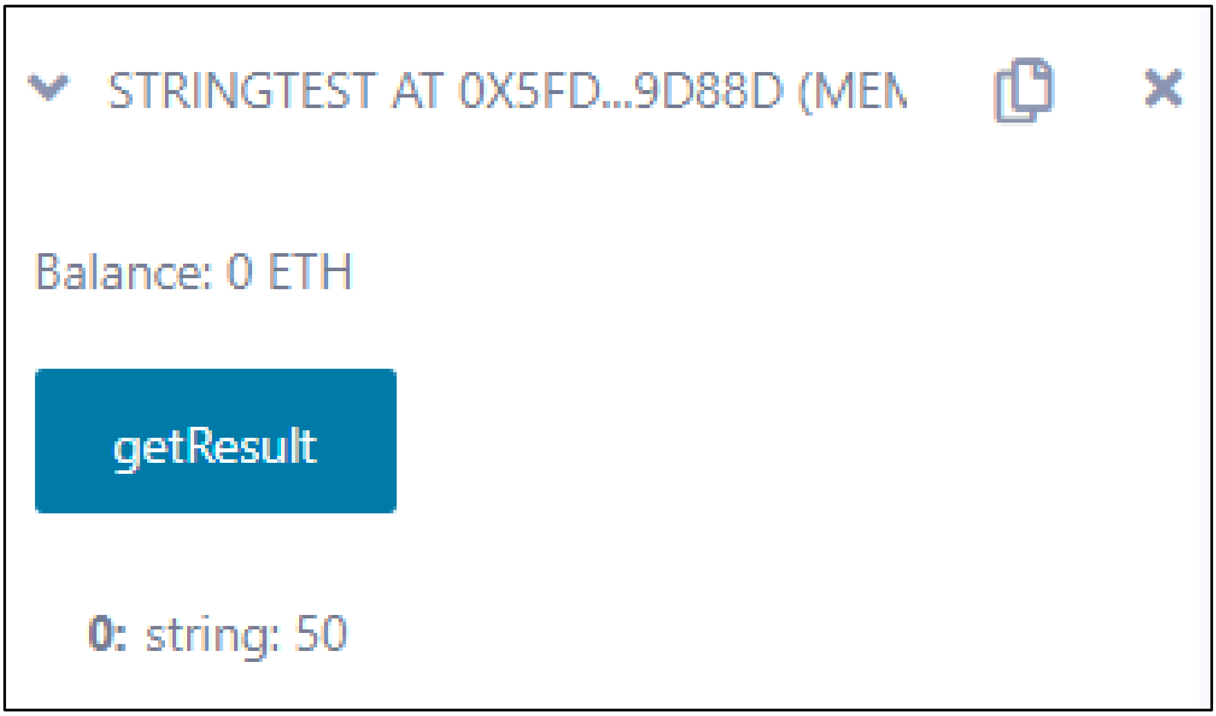
0: string: hello

[ii] Single Quotes

CODE:

```
pragma solidity ^0.5.2;
contract stringTest
{
    function getResult() public pure returns(string memory) {
        uint a = 25;
        uint b = 25;
        uint result = a + b;
        return integerToString(result);
    }
    function integerToString(uint _i) internal pure returns (string memory){
        if (_i == 0)
        {
            return "0";
        }
        uint j = _i;
        uint len;
        while (j != 0)
        {
            len++;
            j/= 10;
        }
        bytes memory bstr = new bytes(len);
        uint k = len - 1;
        while (_i != 0)
        {
            bstr[k-- ] = byte(uint8(48 + _i % 10));
            _i /= 10;
        }
    return string(bstr);
    }
}
```

OUTPUT:



STRINGTEST AT 0X5FD...9D88D (MEN

Balance: 0 ETH

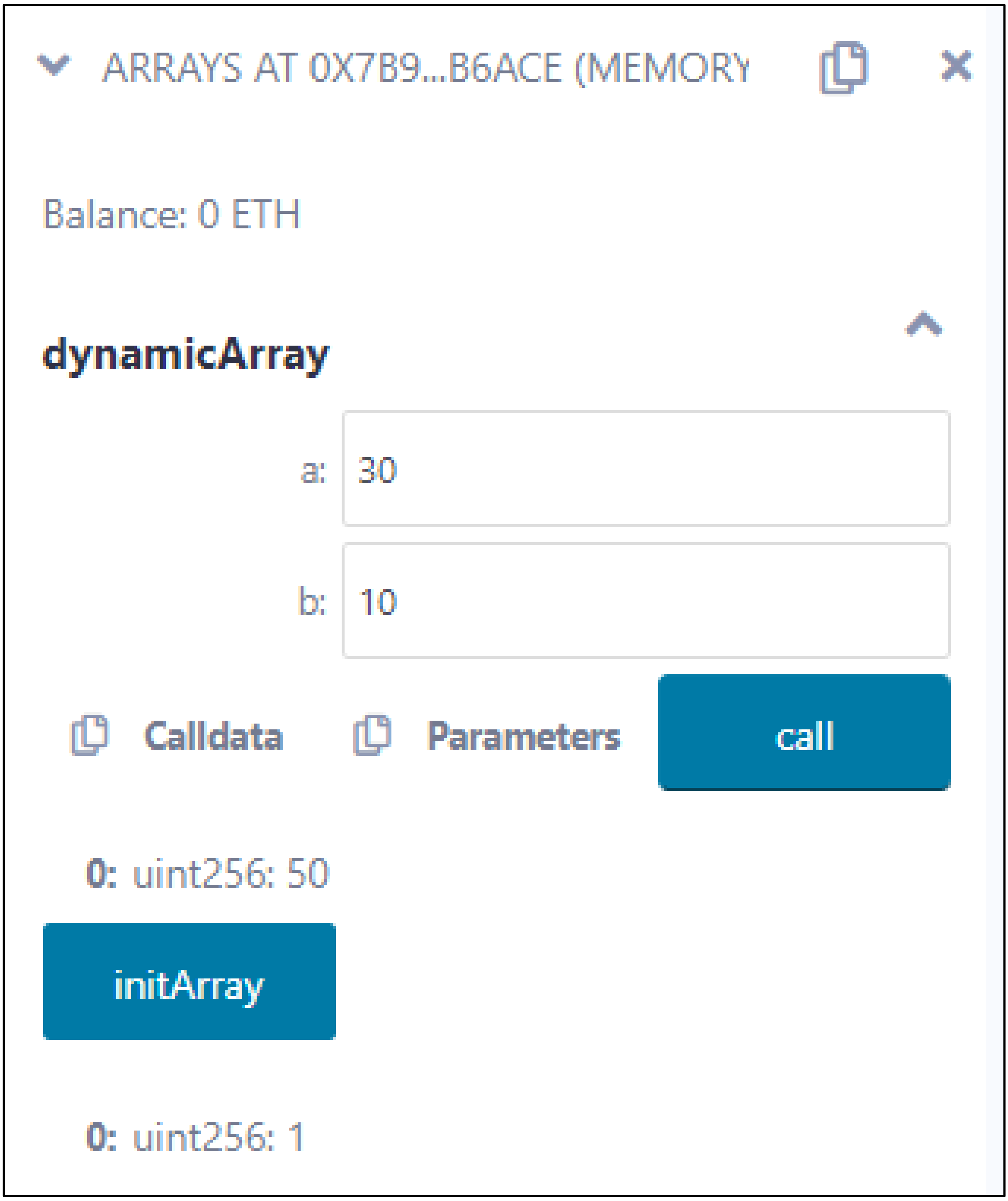
0: string: 50

[VI] Arrays

CODE:

```
// SPDX-License-Identifier: MIT
pragma solidity 0.7.0;

contract Arrays {

    function initArray() public pure returns (uint256) {
        uint128[3] memory array = [1, 2, uint128(3)];
        return array[0];
    }

    function dynamicArray(uint256 a, uint256 b) public pure returns (uint256)
{

        uint128[] memory array = new uint128[](a);
        uint128 val = 5;

        for (uint128 j = 0; j < a; j++) {
            array[j] = j * val;
        }

        return array[b];
    }
}
```

OUTPUT:



ARRAYS AT 0X7B9...B6ACE (MEMORY

Balance: 0 ETH

dynamicArray

a: 30

b: 10

```
0: uint256: 50
```

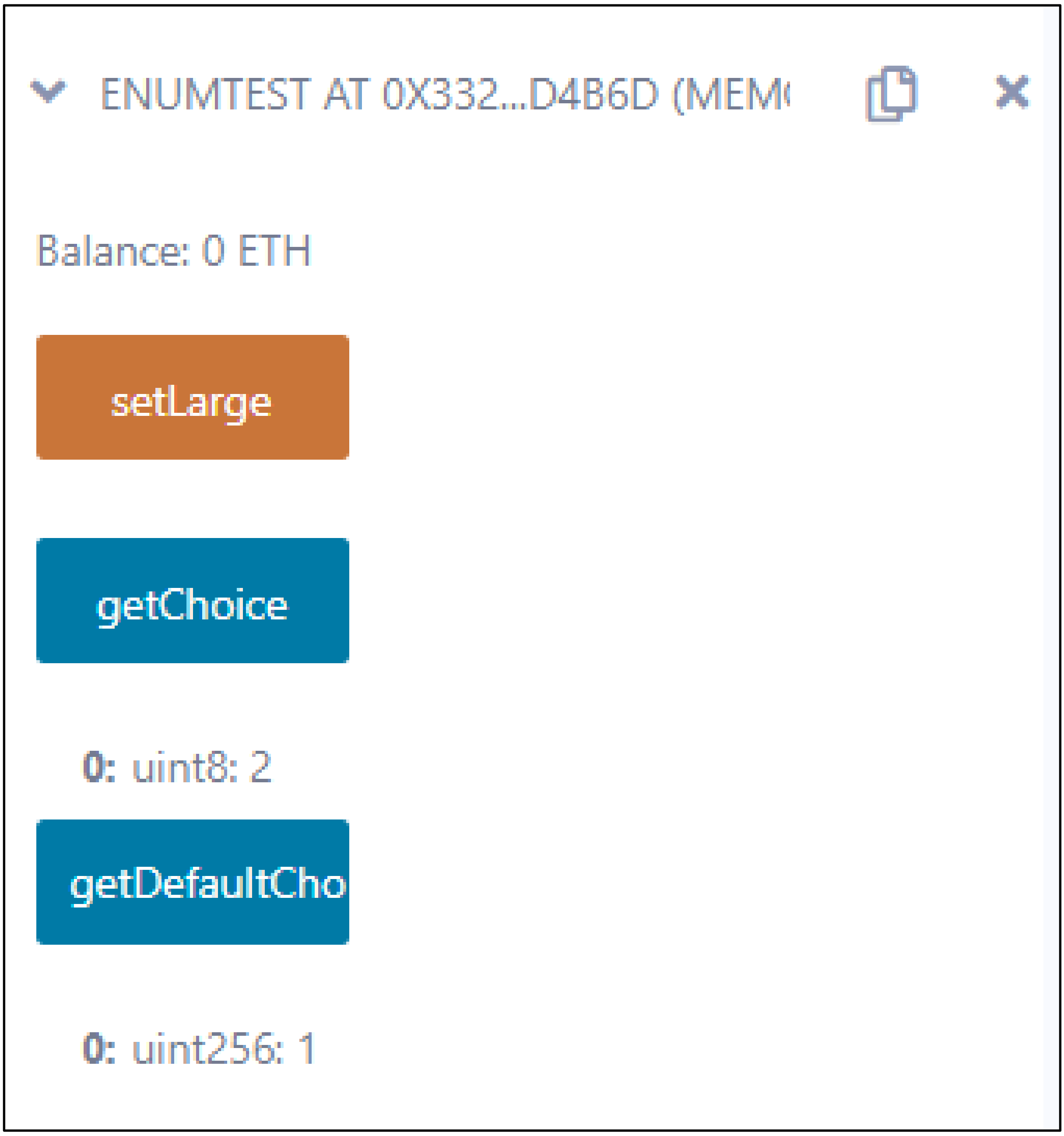
```
0: uint256: 1
```

[VII] Enums

CODE:

```
pragma solidity ^0.5.0;

contract enumTest {
    enum FreshJuiceSize {
        SMALL,
        MEDIUM,
        LARGE
    }
    FreshJuiceSize choice;
    FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;

    function setLarge() public {
        choice = FreshJuiceSize.LARGE;
    }

    function getChoice() public view returns (FreshJuiceSize) {
        return choice;
    }

    function getDefaultChoice() public pure returns (uint256) {
        return uint256(defaultChoice);
    }
}
```

OUTPUT:



Balance: 0 ETH

```
0: uint8: 2
```

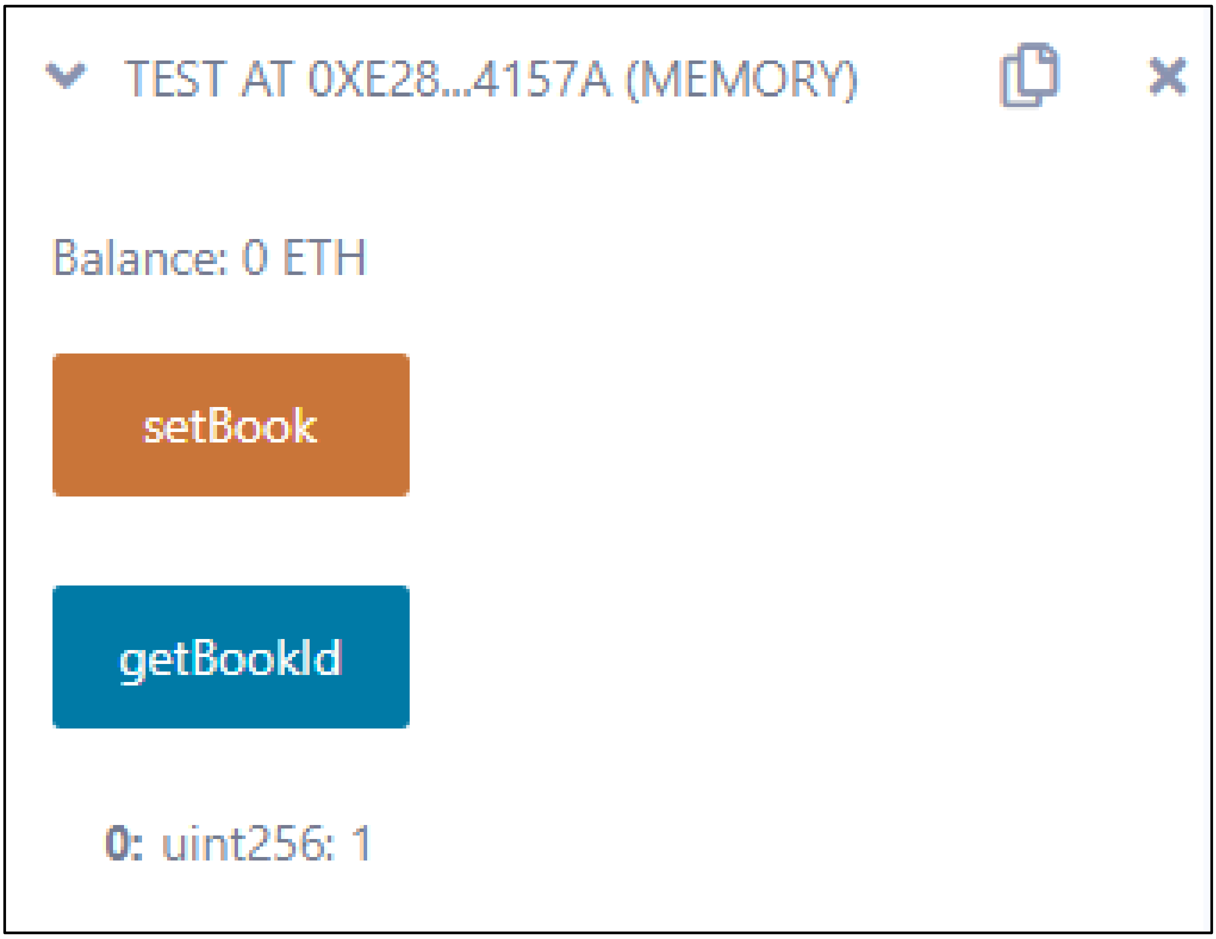
```
0: uint256: 1
```

[VIII] Structs

STEPS:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;

contract test {
    struct Book {
        string title;
        string author;
        uint book_id;
    }

    Book book;

    function setBook() public {
        book = Book('Learn Java', 'TP', 1);
    }

    function getBookId() public view returns (uint) {
        return book.book_id;
    }
}
```

OUTPUT:



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

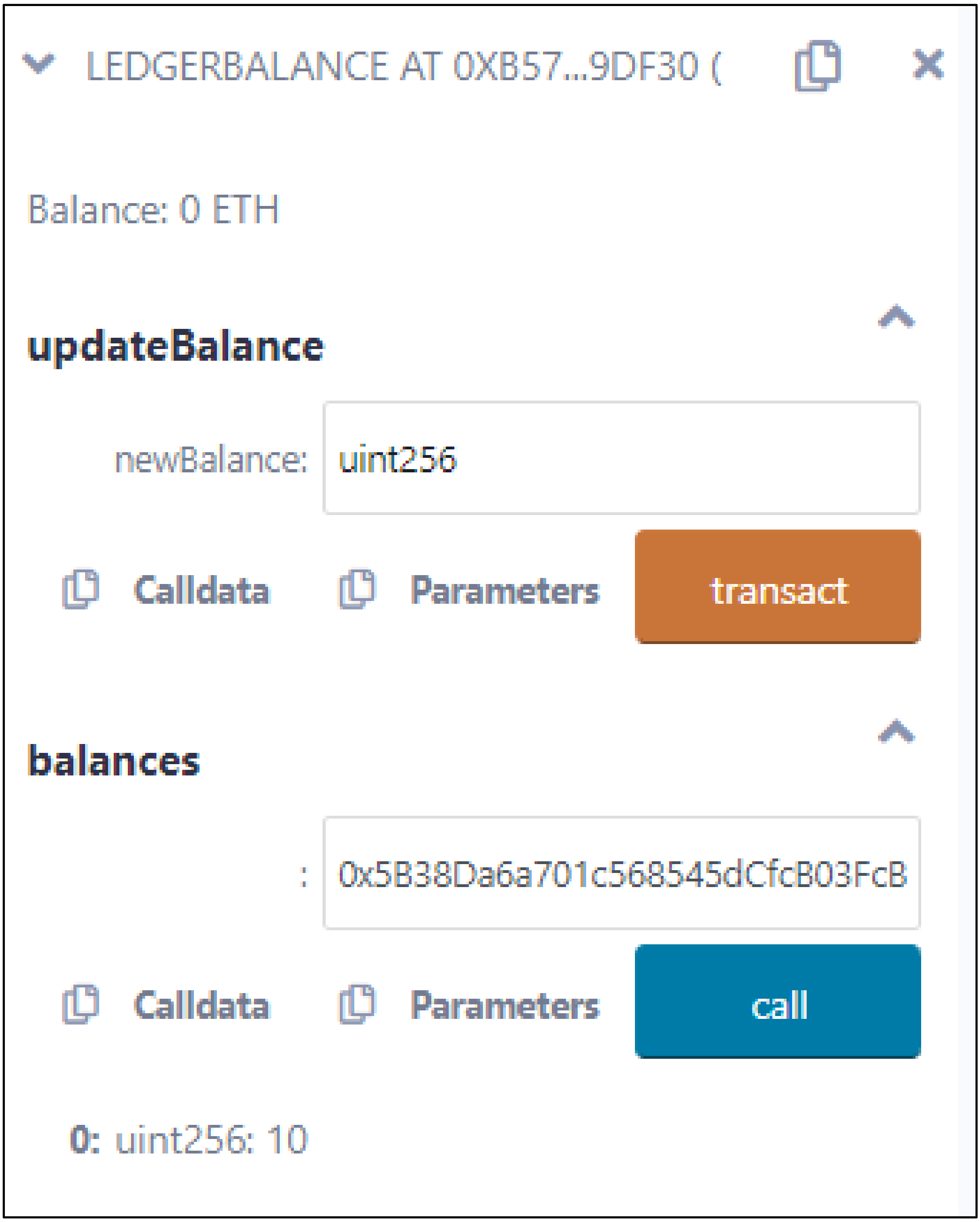
[IX] Mappings

CODE:

```
pragma solidity ^0.5.0;

contract LedgerBalance
{
    mapping(address => uint) public balances;

    function updateBalance(uint newBalance) public
    {
    balances[msg.sender] = newBalance;
    }
}
contract Updater
{
    function updateBalance() public returns (uint)
    {
        LedgerBalance ledgerBalance = new LedgerBalance();
        ledgerBalance.updateBalance(10);
        return ledgerBalance.balances(address(this));
    }
}
```

OUTPUT:

With Original Balance: 10



LEDGERBALANCE AT 0XB57...9DF30 (

Balance: 0 ETH

updateBalance

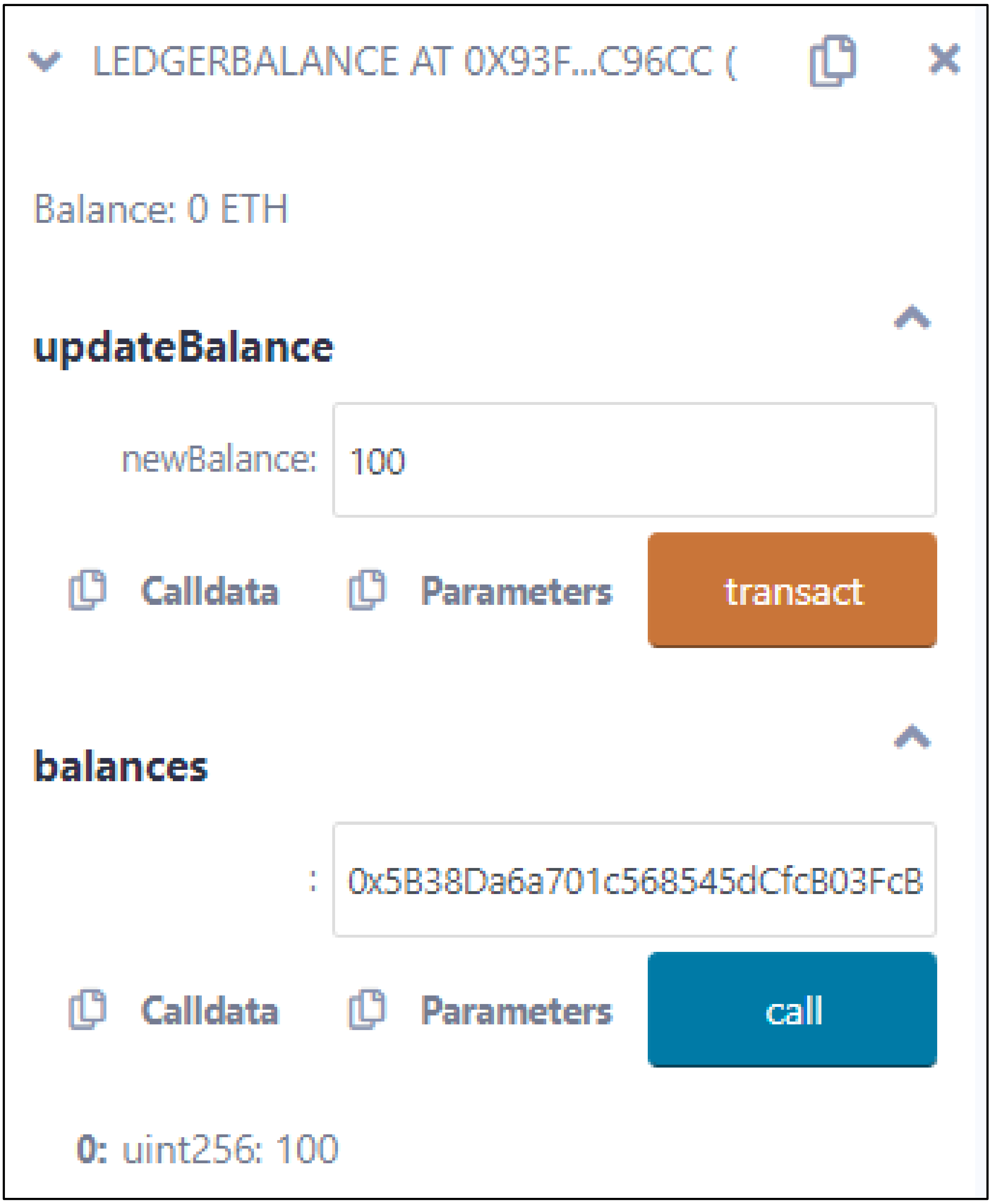
newBalance: uint256

balances

: 0x5B38Da6a701c568545dCfcB03FcB

0: uint256: 10

With Updated Balance: 100



Balance: 0 ETH

updateBalance

newBalance: 100

balances

: 0x5B38Da6a701c568545dCfcB03FcB

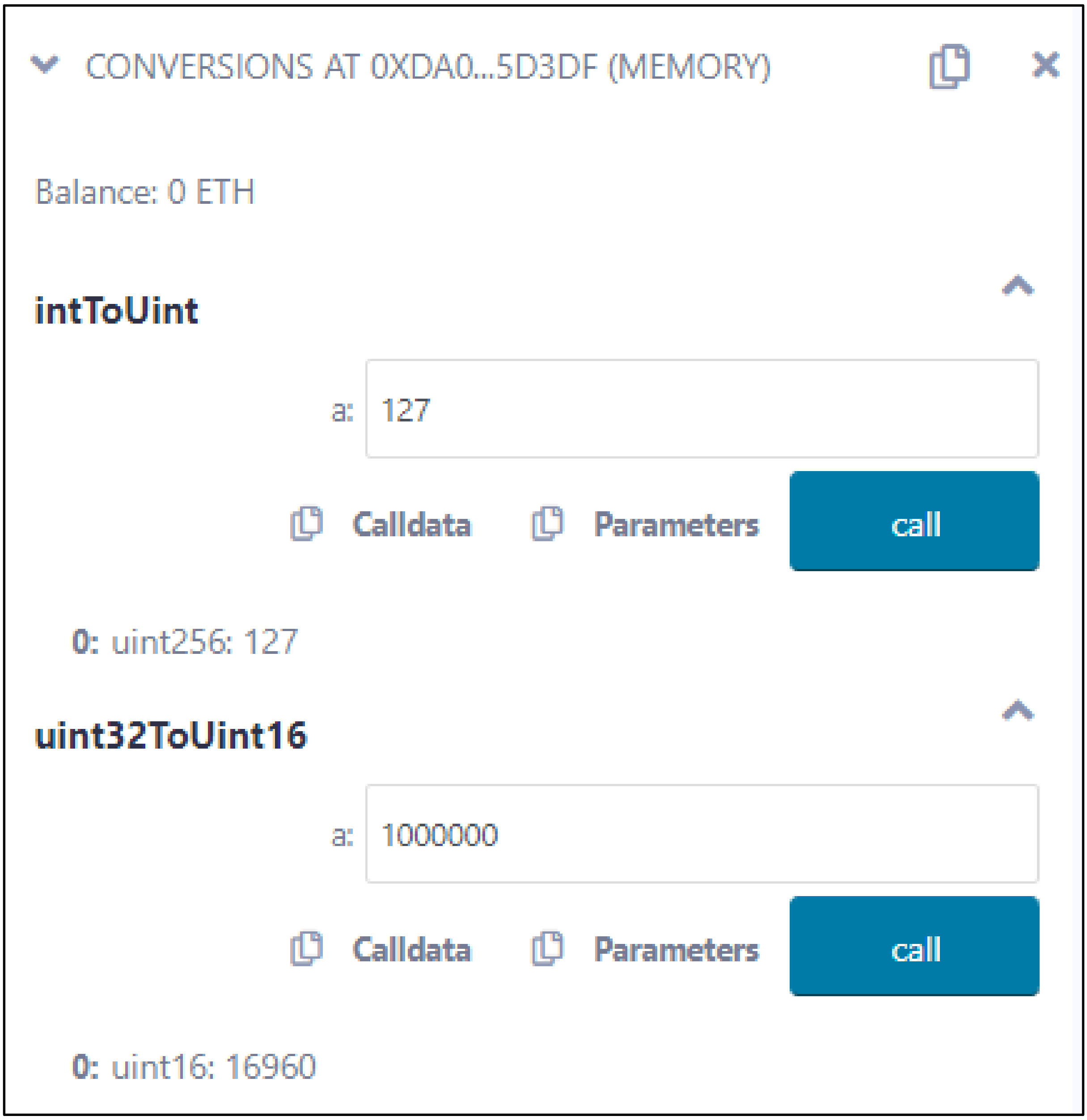
0: uint256: 100

[X] Conversions

CODE:

```
pragma solidity ^0.4.0;

contract Conversions {
    function intToUint(int8 a) public pure returns (uint256) {
        uint256 b = uint256(a);
        return b;
    }

    function uint32ToUint16(uint32 a) public pure returns (uint16) {
        uint16 b = uint16(a);
        return b;
    }
}
```

OUTPUT:



Balance: 0 ETH

intToUint

a: 127

```
0: uint256: 127
```

uint32ToUint16

a: 1000000

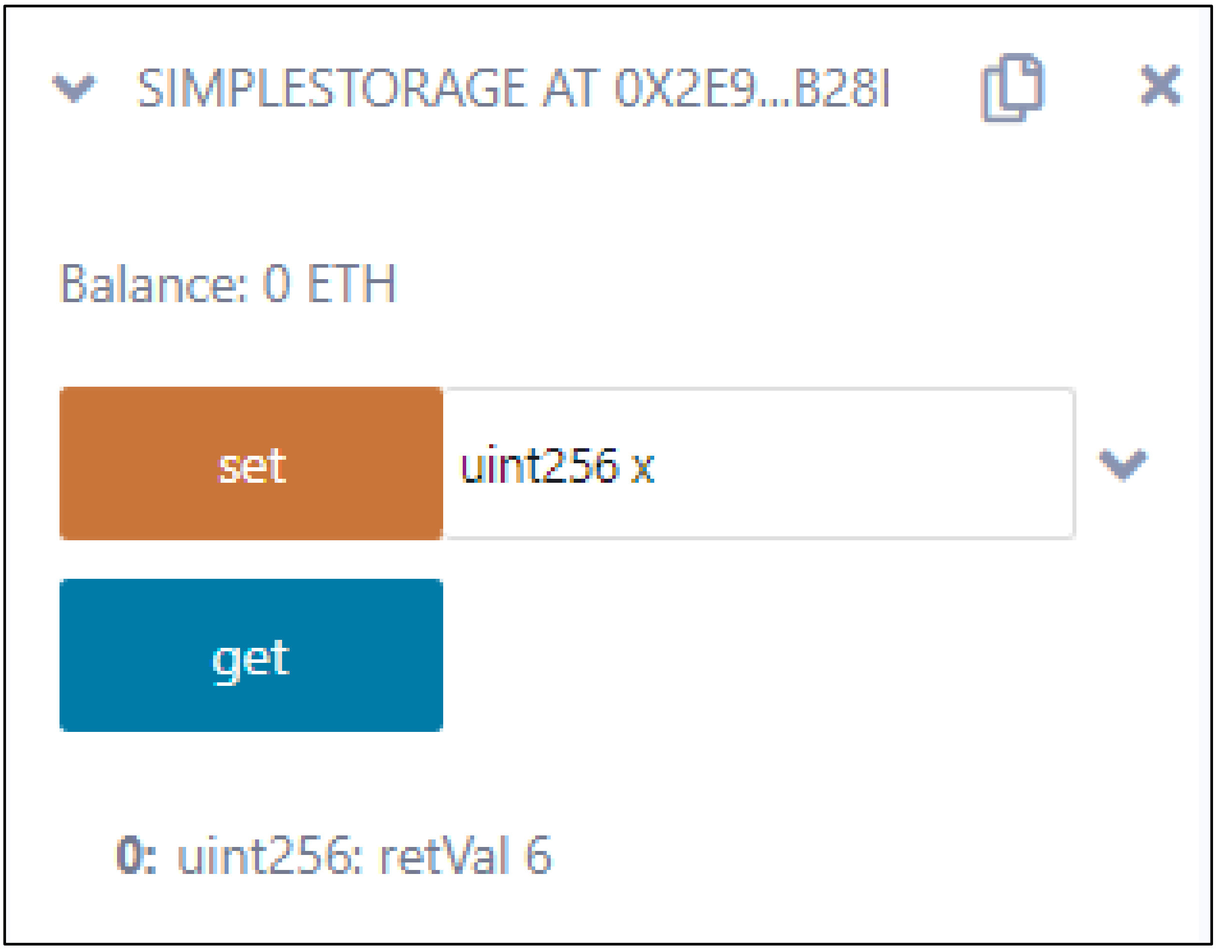
```
0: uint16: 16960
```

[XI] Ether Units

CODE:

```
pragma solidity ^0.4.6;

contract SimpleStorage {
    uint256 storedData = 2;

    function set(uint256 x) public {
        storedData = x;
        /*
        Ether Units
        Wei
        Finney
        Szabo
        Ether
        */
        if (2000000000000000000 == 2 ether) {
            storedData = 2;
        }
        else {
            storedData = 3;
        }
        /*
        Time Units
        seconds
        minutes
        hours
        days
        weeks
        month
        years
        */
        if (120 seconds == 2 minutes) {
            storedData = 6;
        }
        else {
            storedData = 9;
        }
    }

    function get() constant public returns (uint256 retVal)
    {
        return storedData;
    }
}
```

OUTPUT:



Balance: 0 ETH

uint256 x

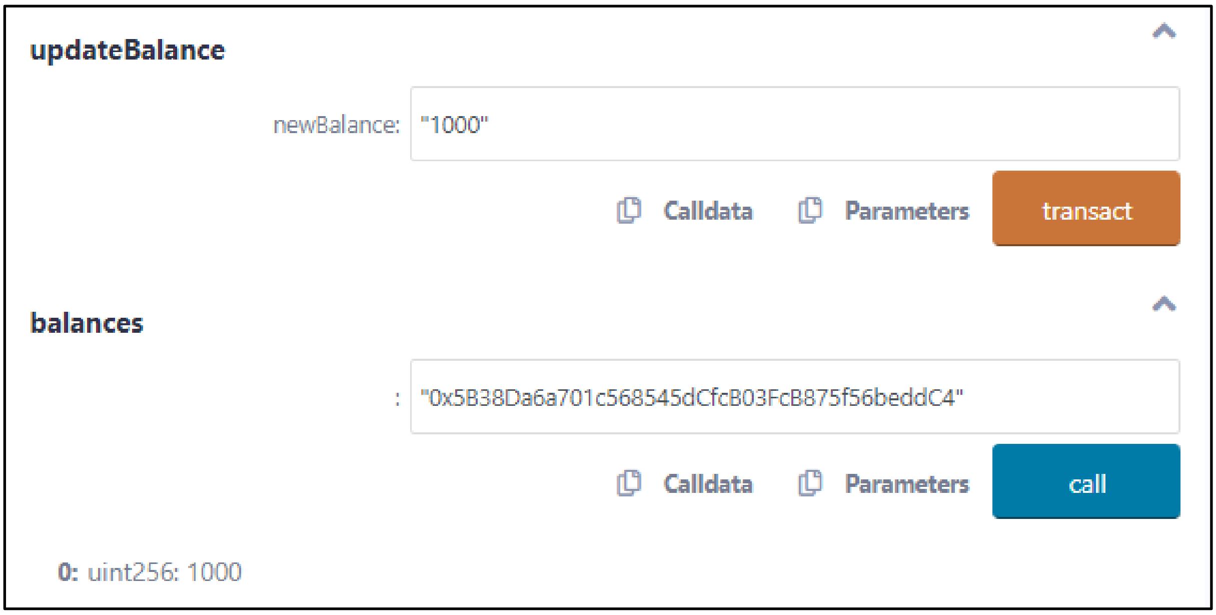
0: uint256: retVal 6

[XII] Special Variables

CODE:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;

contract LedgerBalance {
    mapping(address => uint256) public balances;

    function updateBalance(uint256 newBalance) public {
        balances[msg.sender] = newBalance;
    }
}

contract Updater {
    function updateBalance() public returns (uint256) {
        LedgerBalance ledgerBalance = new LedgerBalance();
        ledgerBalance.updateBalance(10);
        return ledgerBalance.balances(address(this));
    }
}
```

OUTPUT:



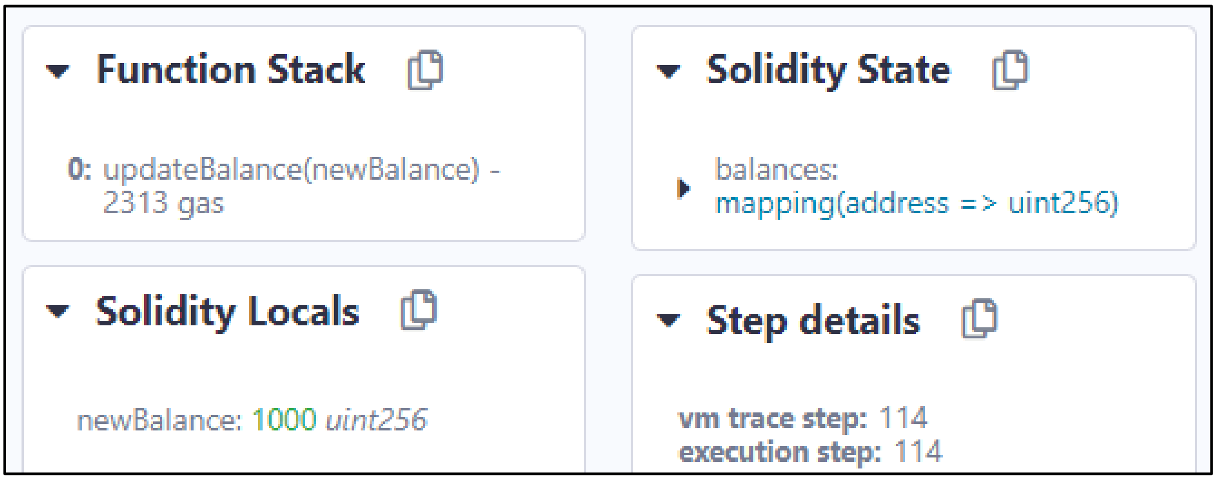
updateBalance

newBalance: "1000"

balances

: "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"

0: uint256: 1000



0: updateBalance(newBalance) - 2313 gas

balances: mapping(address => uint256)

newBalance: 1000 *uint256*

vm trace step: 114
execution step: 114

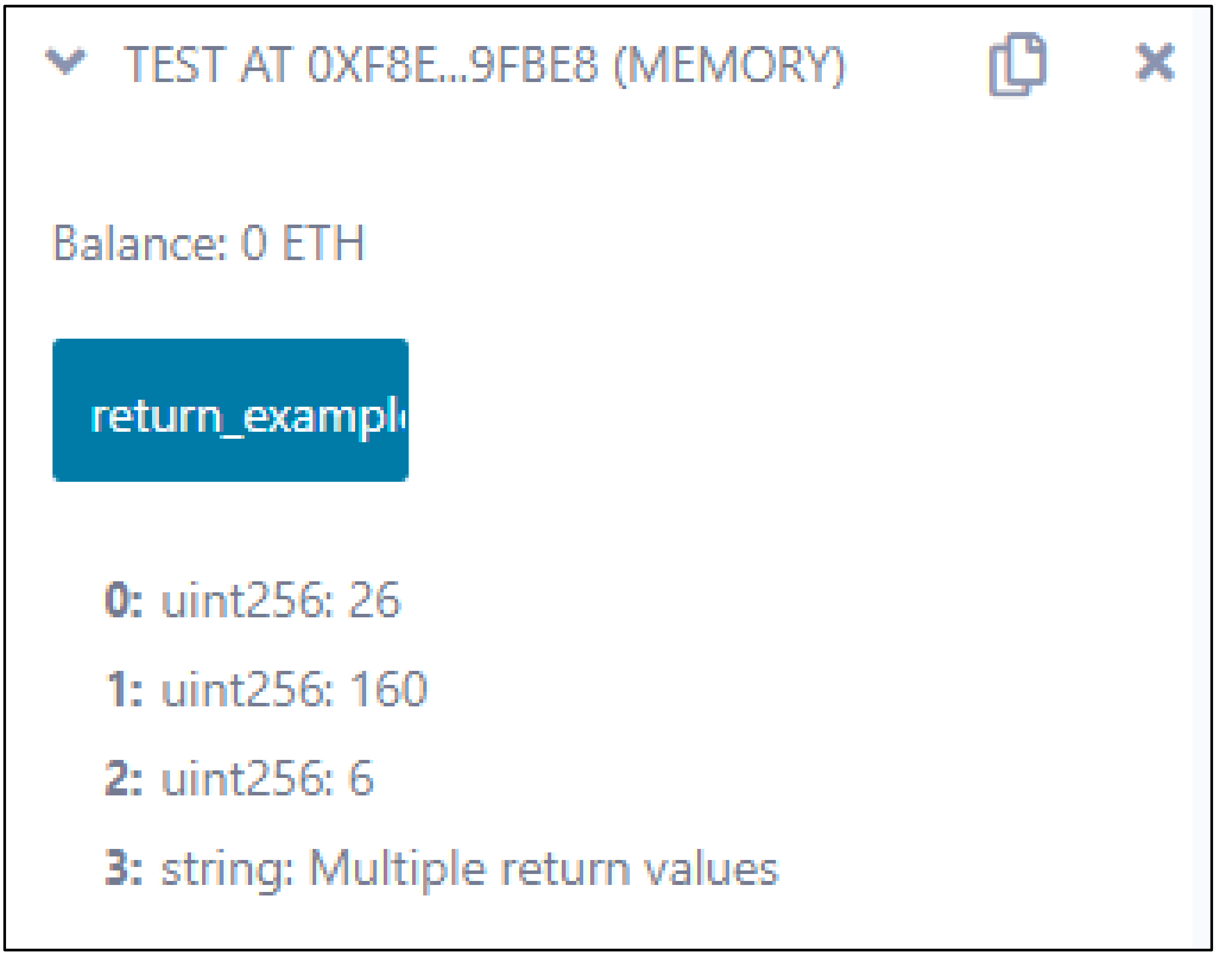
[B] Functions, Function Modifiers, View Functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

[I] Functions

CODE:

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

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

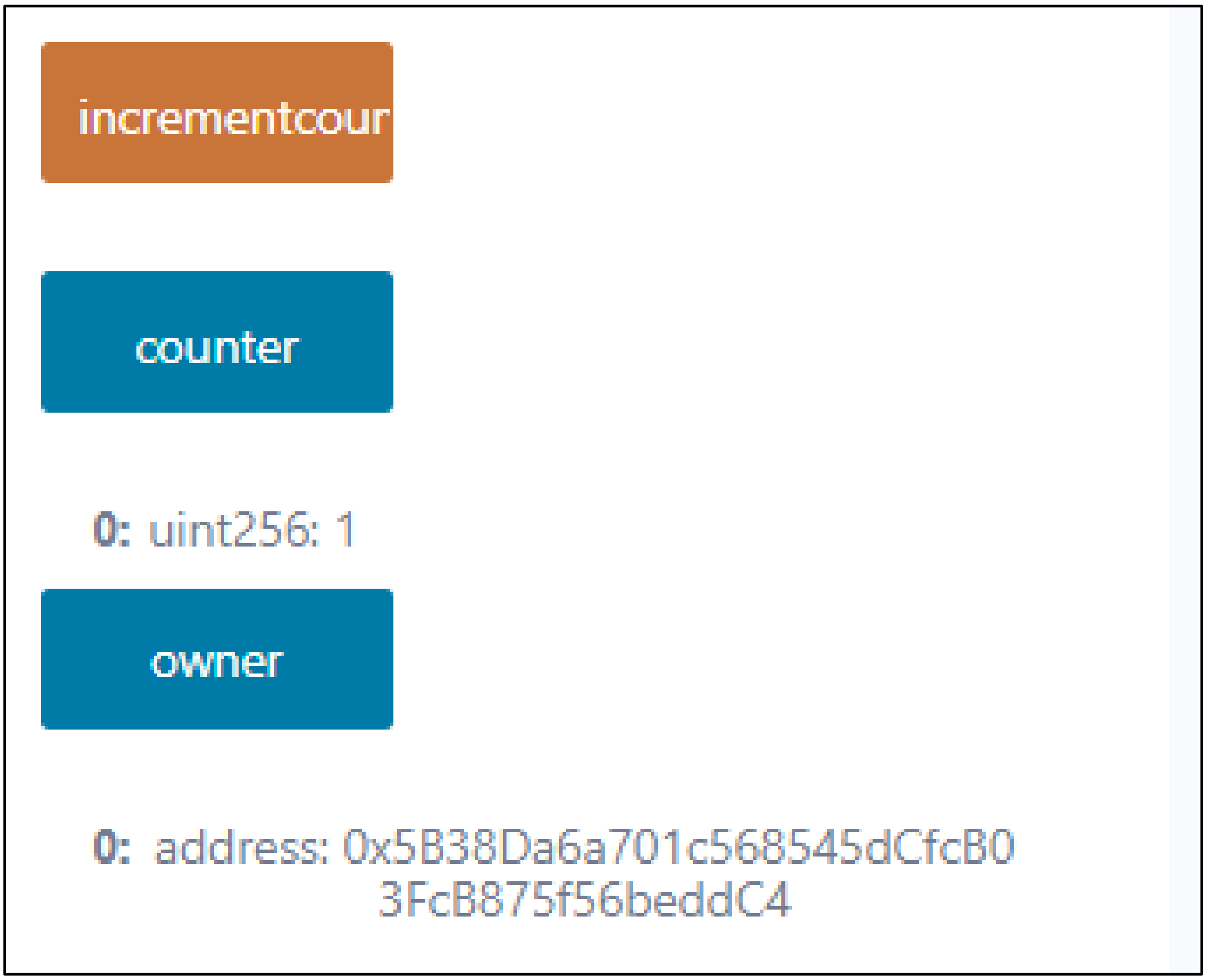
```
0: uint256: 26
1: uint256: 160
2: uint256: 6
3: string: Multiple return values
```

[II] Function Modifiers

CODE:

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

contract ExampleContract {
  address public owner = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;
  uint256 public counter;

  modifier onlyowner() {
    require(msg.sender == owner, "Only the contract owner can call");
    _;
  }

  function incrementcounter() public onlyowner {
    counter++;
  }
}
```

OUTPUT:



```
0: uint256: 1
```

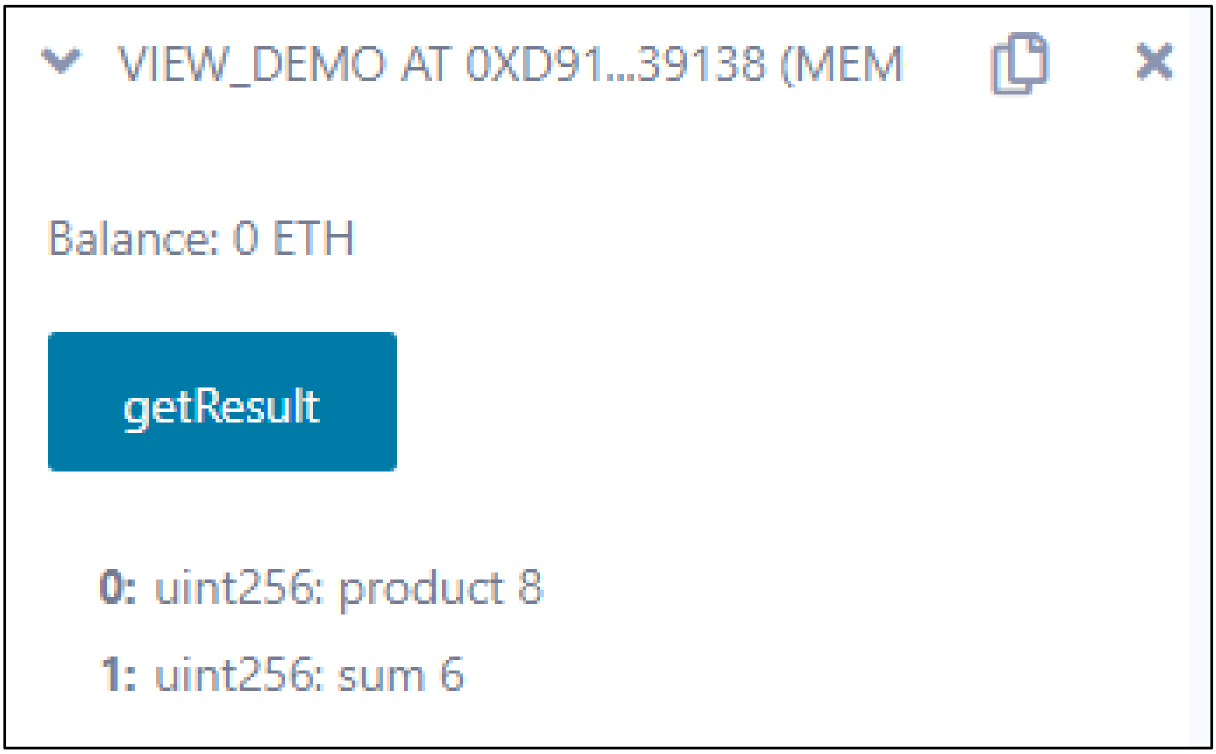
```
0: address: 0x5B38Da6a701c568545dCfcB0
            3FcB875f56beddC4
```

[III] View Functions

CODE:

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

contract view_demo {
  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

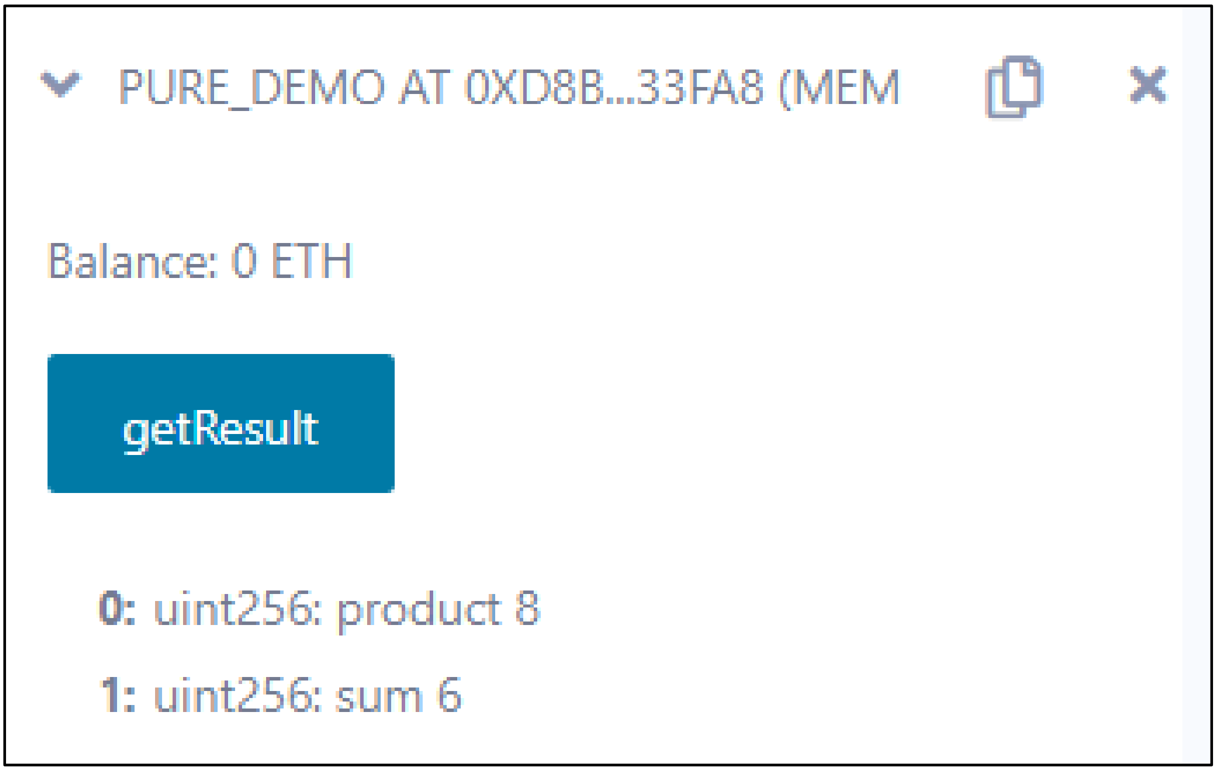
1: uint256: sum 6

[IV] Pure Functions

CODE:

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

contract pure_demo {
  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

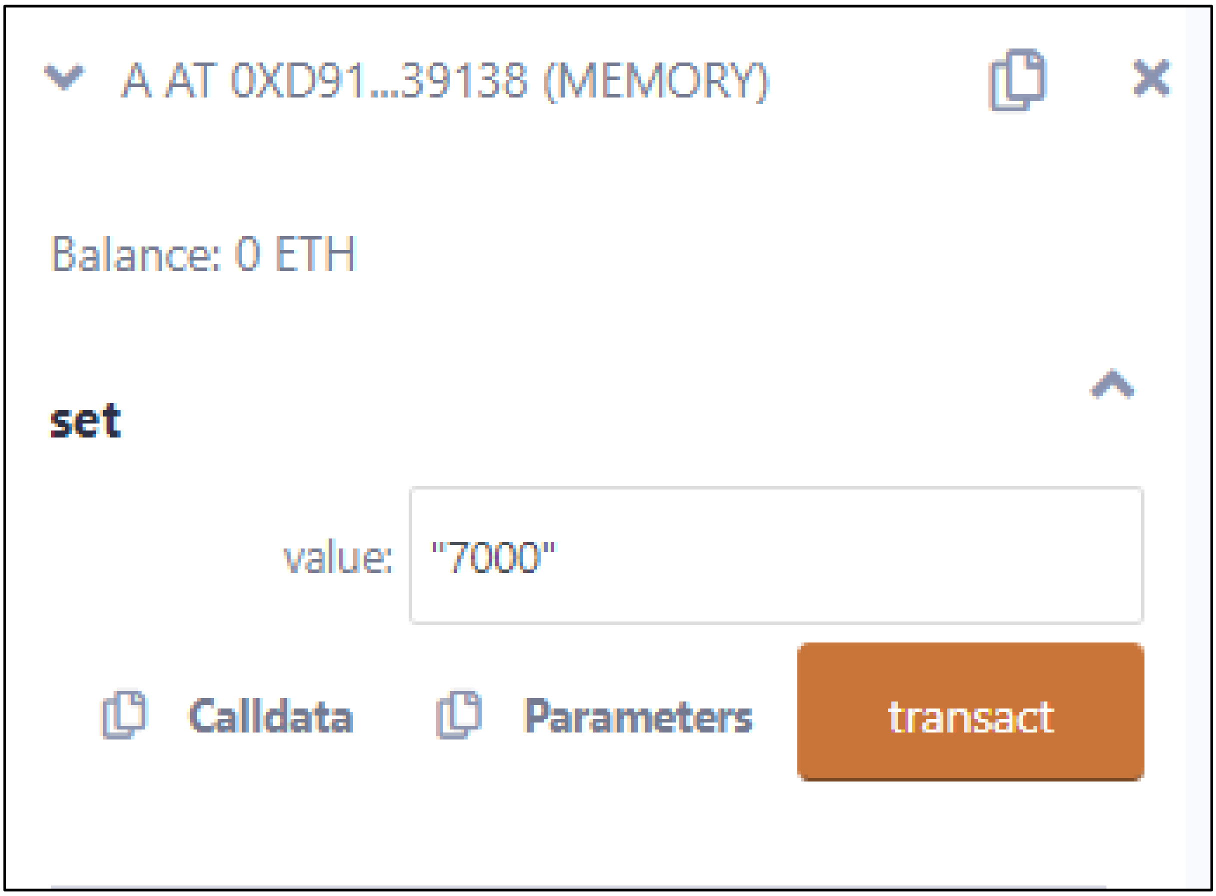
```
0: uint256: product 8
1: uint256: sum 6
```

[V] Fallback Function

CODE:

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

contract A {
  uint256 n;

  function set(uint256 value) external {
    n = value;
  }

  function() external payable {
    n = 0;
  }
}

contract example {
  function callA(A a) public returns (bool) {
    (bool success, ) = address(a).call(abi.encodeWithSignature("setter()"));
    require(success);
    address payable payableA = address(uint160(address(a)));
    return (payableA.send(2 ether));
  }
}
```

OUTPUT:

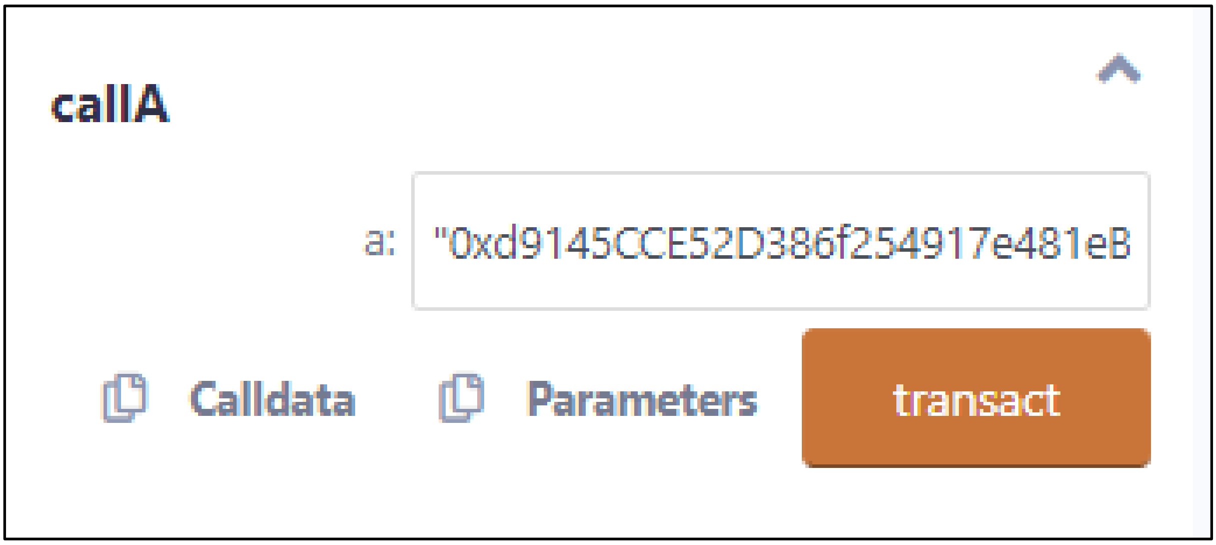


A AT 0XD91...39138 (MEMORY)

Balance: 0 ETH

set

value: "7000"



callA

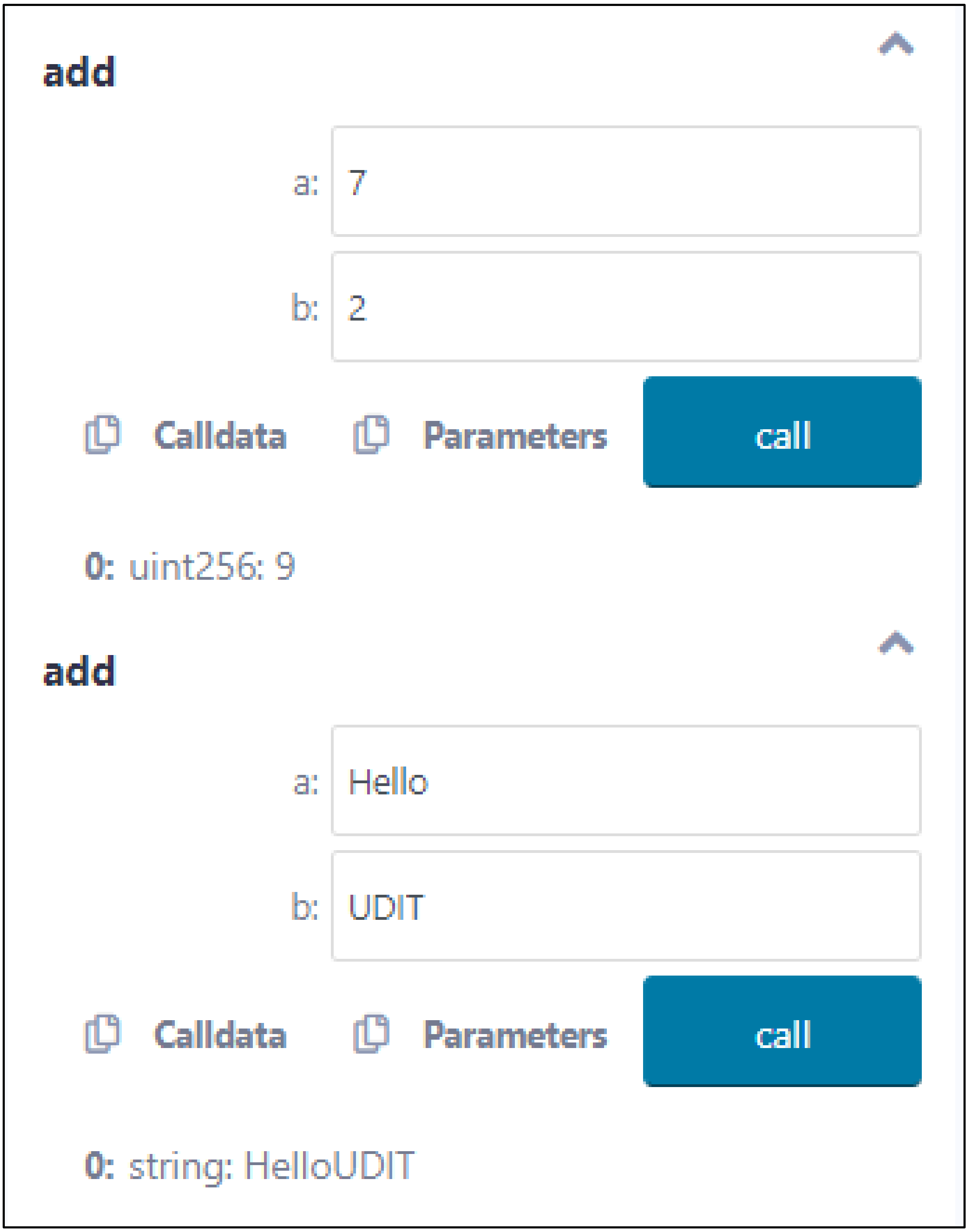
a: "0xd9145CCE52D386f254917e481eB

[VI] Function Overloading

CODE:

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

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:



add

a: 7

b: 2

0: uint256: 9

add

a: Hello

b: UDIT

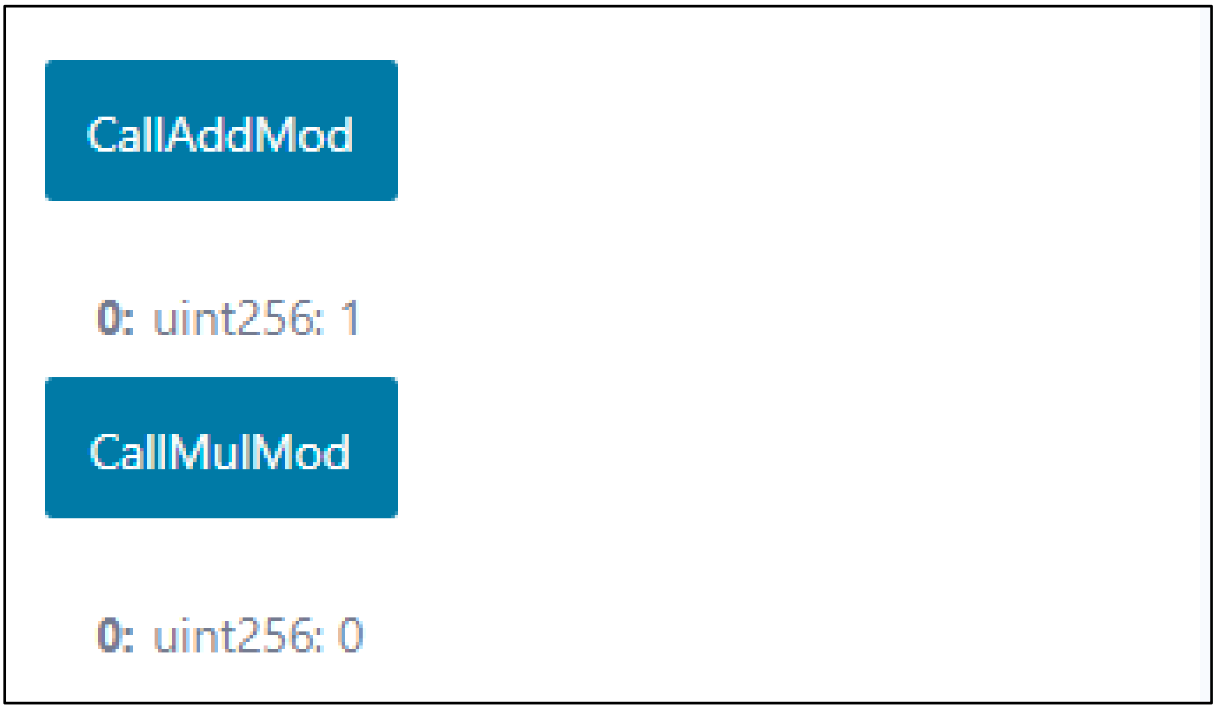
0: string: HelloUDIT

[VII] Mathematical Functions

CODE:

```
// SPDX-License-Identifier: MIT
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:



```
0: uint256: 1
```

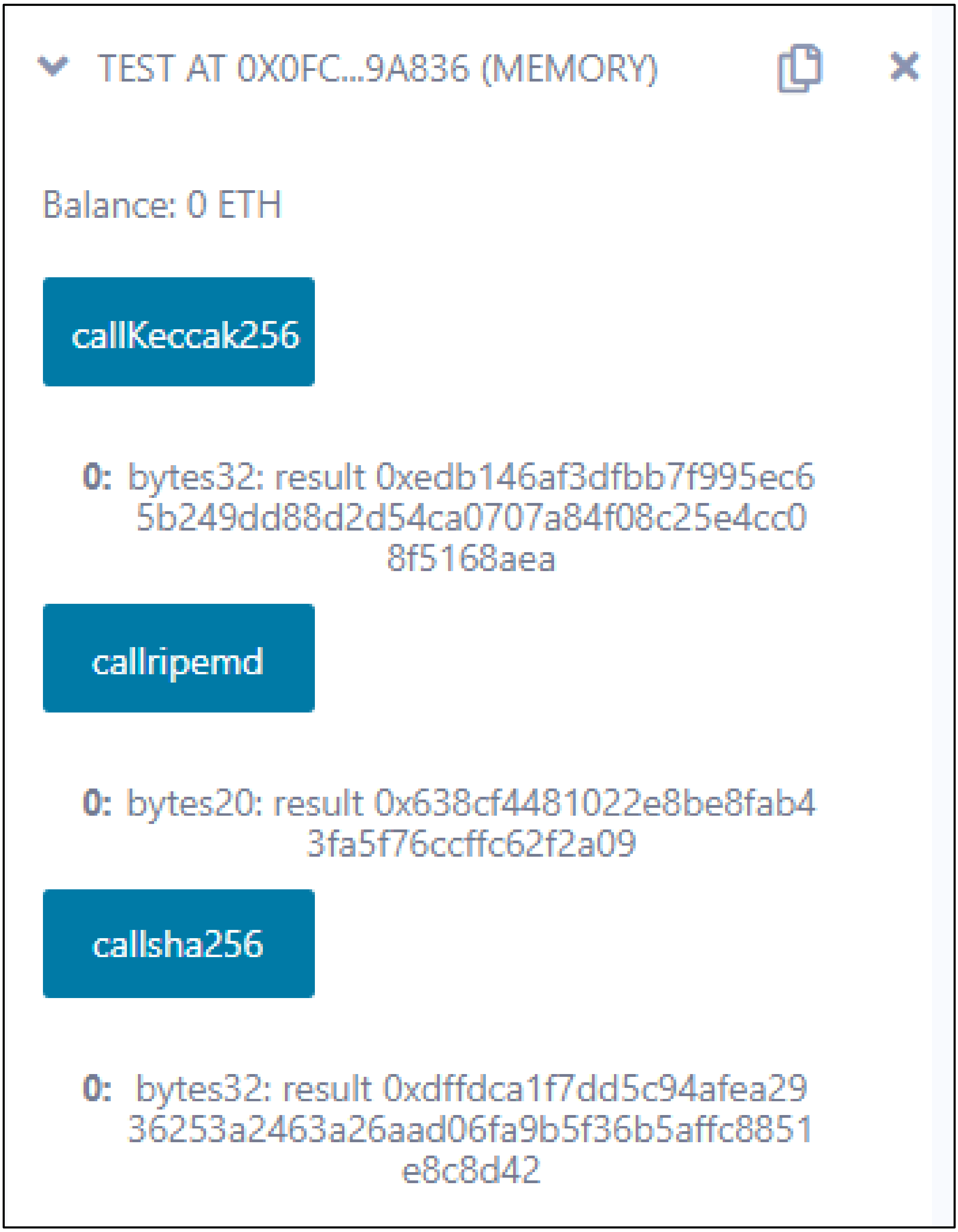
```
0: uint256: 0
```

[VIII] Cryptographic Functions

CODE:

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

contract Test{ function callKeccak256() public pure returns(bytes32 result){

  return keccak256("BLOCKCHAIN");



}

function callsha256() public pure returns(bytes32 result){

  return sha256("BLOCKCHAIN");



}

function callripemd() public pure returns (bytes20 result){

  return ripemd160("BLOCKCHAIN");

  }
}
```

OUTPUT:



```
Balance: 0 ETH

0: bytes32: result 0xedb146af3dfbb7f995ec6
   5b249dd88d2d54ca0707a84f08c25e4cc0
   8f5168aea

0: bytes20: result 0x638cf4481022e8be8fab4
   3fa5f76ccffc62f2a09

0: bytes32: result 0xdffdca1f7dd5c94afea29
   36253a2463a26aad06fa9b5f36b5affc8851
   e8c8d42
```

PRACTICAL 4

AIM: Implement and demonstrate the use of the following in Solidity:

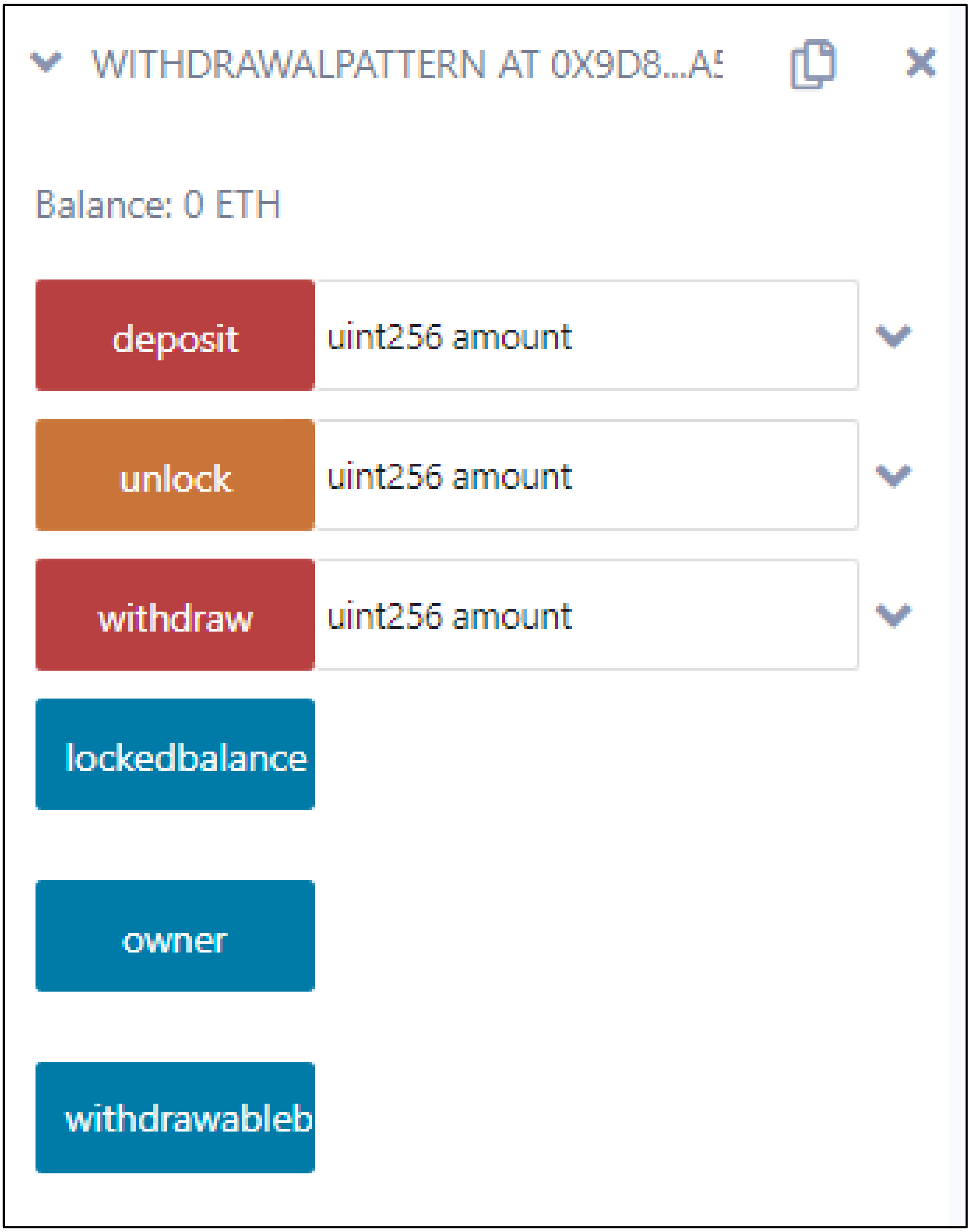
[A] Withdrawal Pattern, Restricted Access.

[I] Withdrawal Pattern

CODE:

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

contract WithdrawalPattern {
  address public owner;
  uint256 public lockedbalance;
  uint256 public withdrawablebalance;

  constructor() {
    owner = msg.sender;
  }

  modifier onlyowner() {
    require(msg.sender == owner, "Only the owner can call this function");
    _;
  }

  function deposit(uint256 amount) public payable {
    require(amount > 0, "Amount must be greater than zero");
    lockedbalance += amount;
  }

  function withdraw(uint256 amount) public payable onlyowner {
    require(
      amount <= withdrawablebalance,
      "Insufficient withdrawable balance"
    );
    withdrawablebalance -= amount;
    payable(msg.sender).transfer(amount);
  }

  function unlock(uint256 amount) public onlyowner {
    require(amount <= lockedbalance, "Insufficient locked balance");
    lockedbalance -= amount;
    withdrawablebalance += amount;
  }
}
```

OUTPUT:



WITHDRAWALPATTERN AT 0X9D8...A5

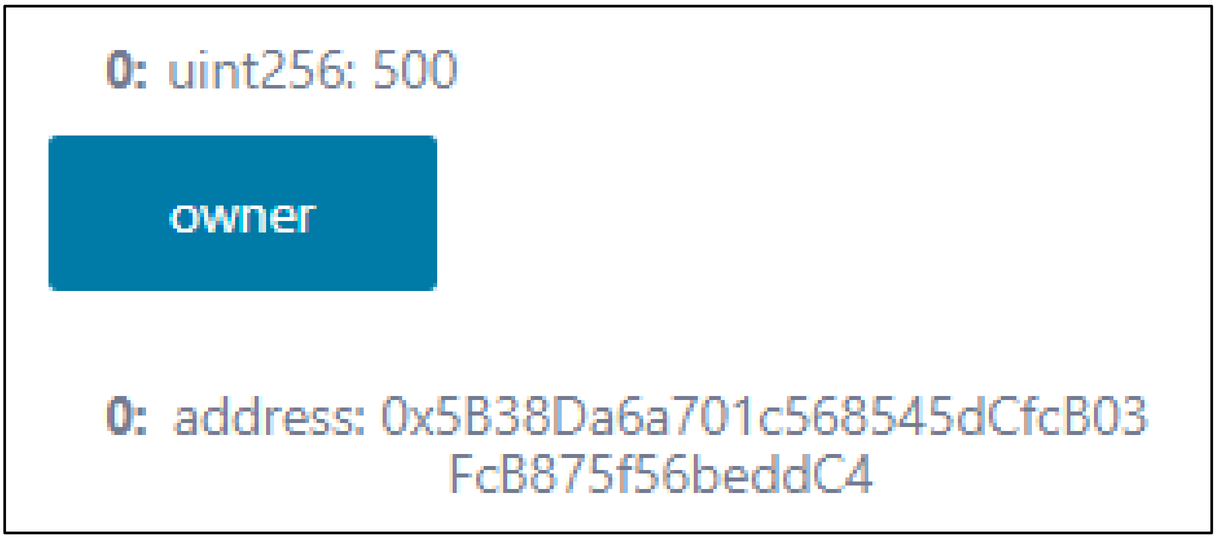
Balance: 0 ETH

deposit uint256 amount

unlock uint256 amount

withdraw uint256 amount

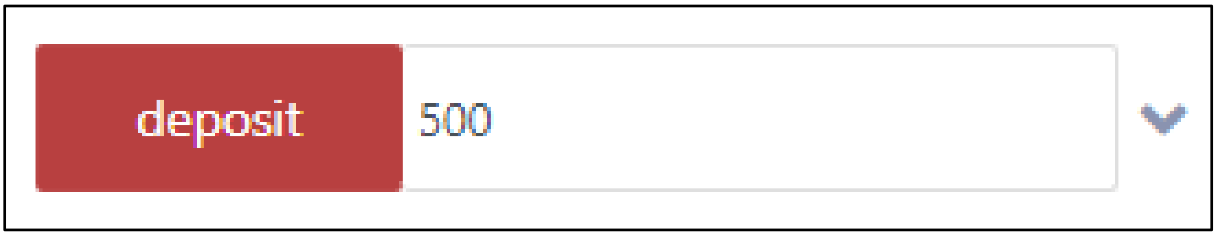
Step 1: Click on owner to create the owner object.



0: uint256: 500

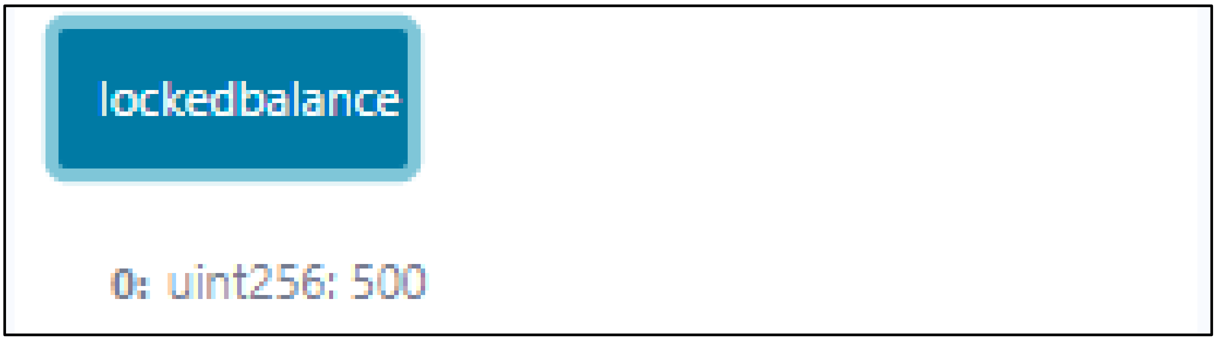
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

Step 2: Enter an amount and click on deposit.



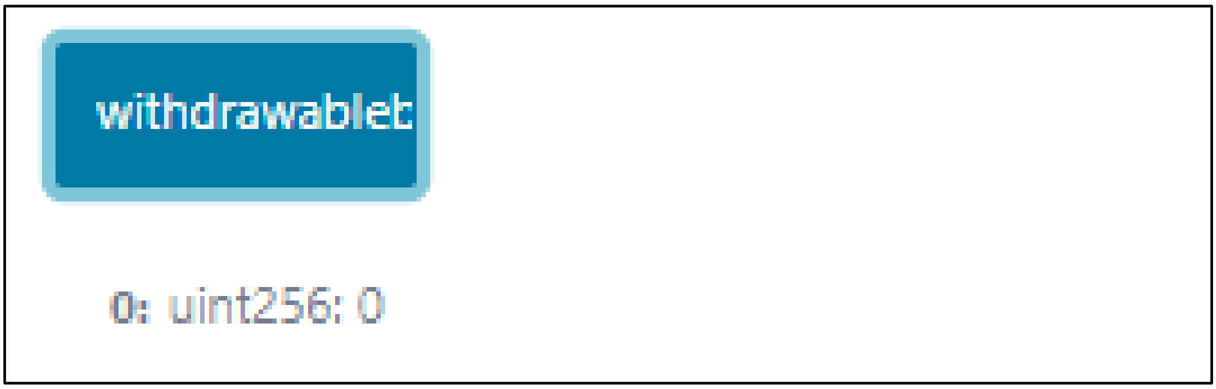
500

Step 3: Click on locked balance button to display the locked amount in the account.



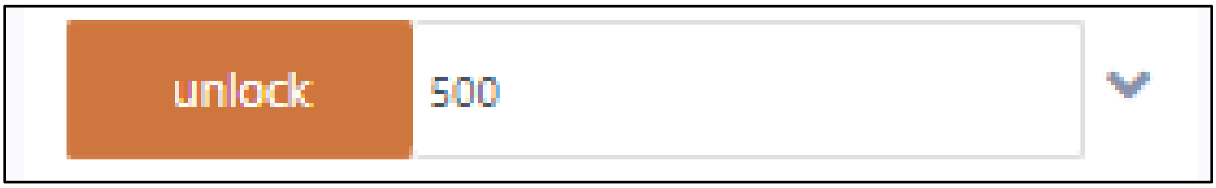
0: uint256: 500

Step 4: Click on withdrawable balance button.

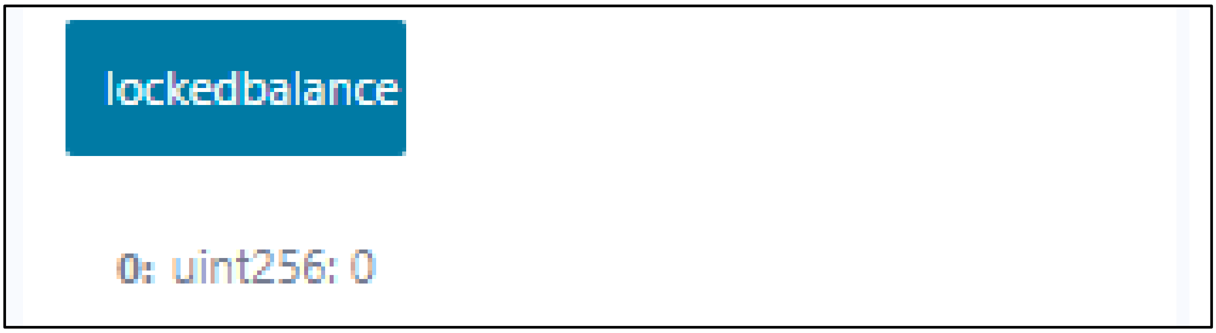


```
0: uint256: 0
```

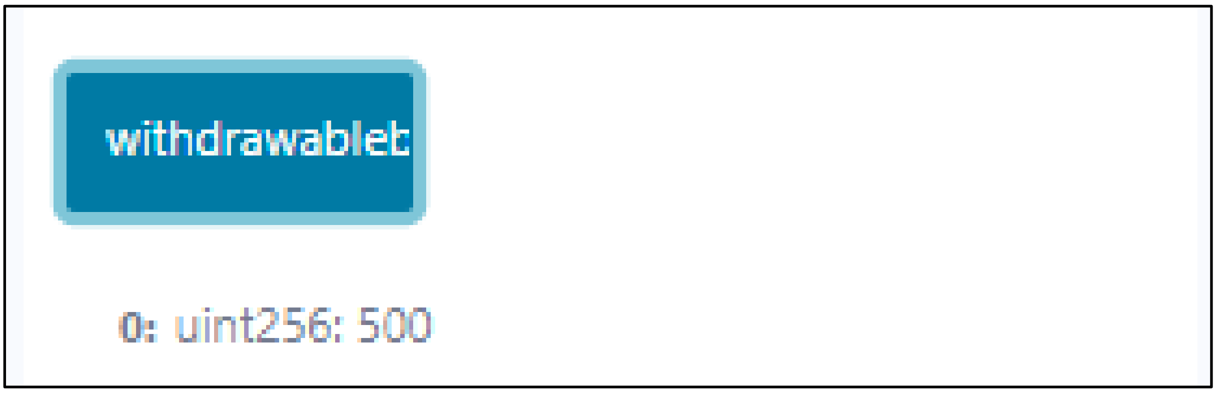
Step 5: Click on unlock button and enter any amount to transfer amount to withdrawable balance. Check locked balance and withdrawable balance.



500

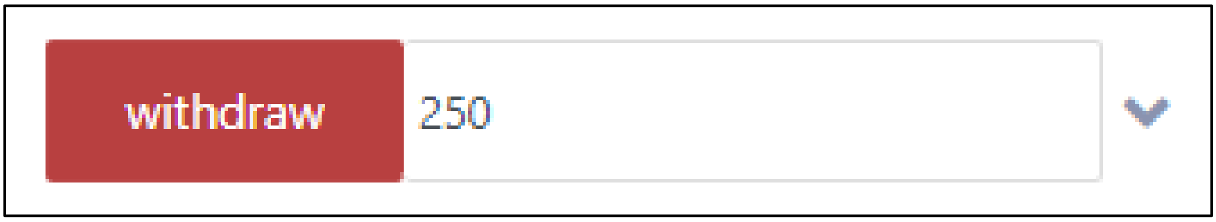


```
0: uint256: 0
```

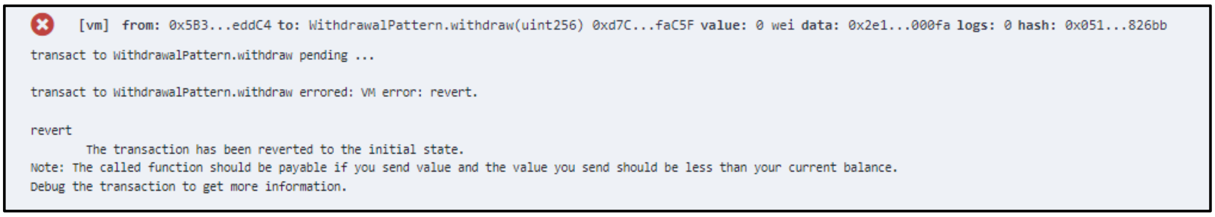


```
0: uint256: 500
```

Step 6: Enter any amount you want to withdraw and click the withdraw button. You should get an error and the transaction should be reverted.



250



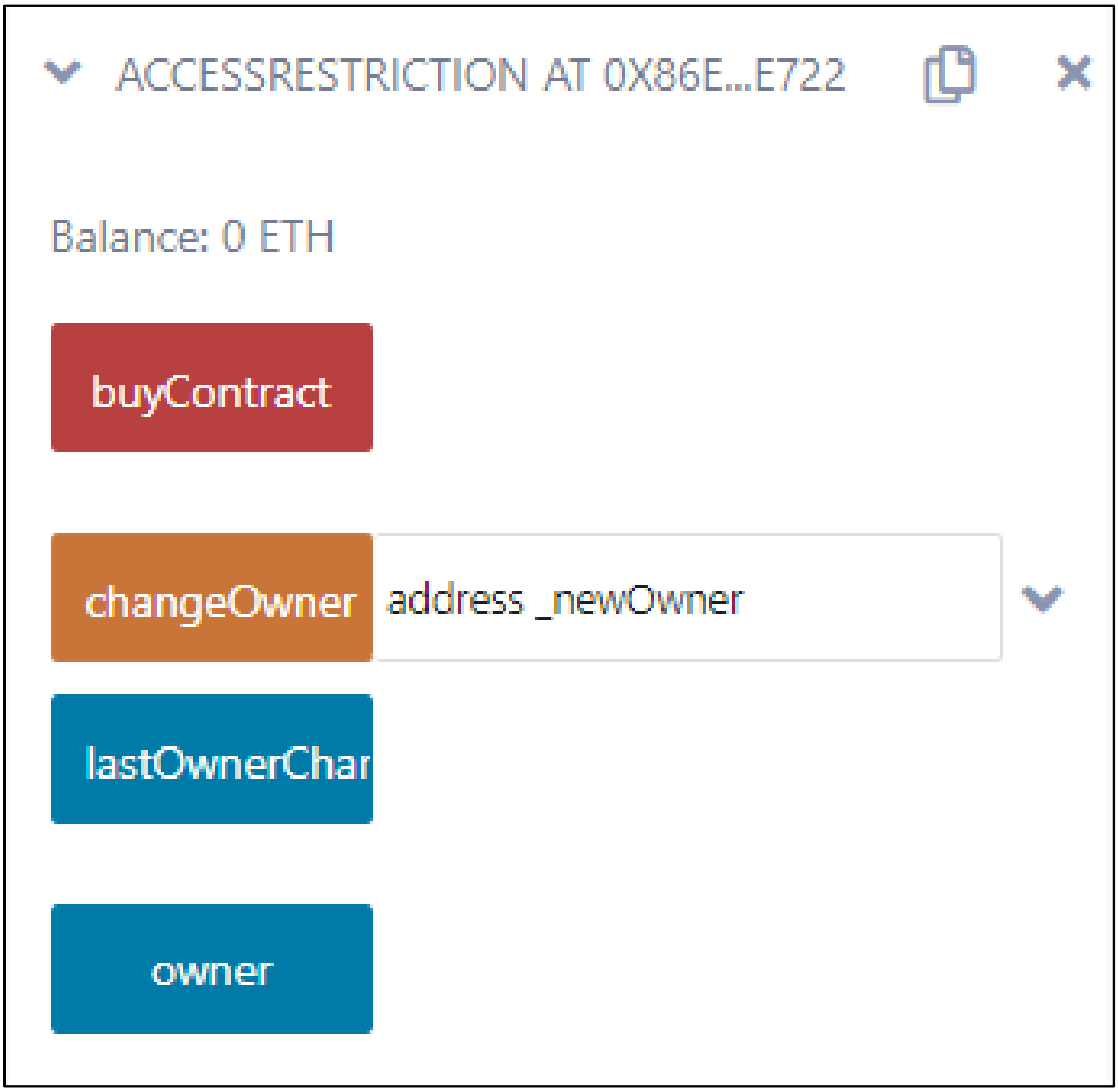
```
[vm]  from: 0x5B3...eddC4 to: WithdrawalPattern.withdraw(uint256) 0xd7C...faC5F value: 0 wei data: 0x2e1...000fa logs: 0 hash: 0x051...826bb
transact to WithdrawalPattern.withdraw pending ...

transact to WithdrawalPattern.withdraw errored: VM error: revert.

revert
	The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
Debug the transaction to get more information.
```

[II] Restricted Access

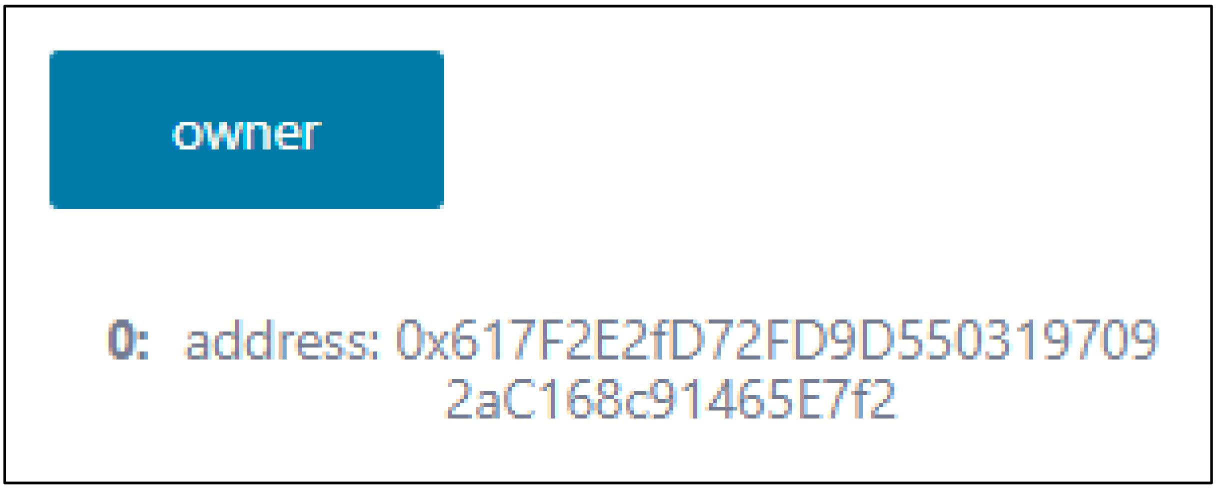
CODE:

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

contract AccessRestriction {

    address public owner = msg.sender;
    uint public lastOwnerChange = now;

    modifier onlyBy(address _account) {
        require(msg.sender == _account);
        _;
    }

    modifier onlyAfter(uint _time) {
        require(now >= _time);
        _;
    }

    modifier costs(uint _amount) {
        require(msg.value >= _amount);
        _;
        if (msg.value > _amount) {
            msg.sender.transfer(msg.value - _amount);
        }
    }

    function changeOwner(address _newOwner) public onlyBy(owner) {
        owner = _newOwner;
    }

    function buyContract() public payable onlyAfter(lastOwnerChange +
4 weeks) costs(1 ether) {
        owner = msg.sender;
        lastOwnerChange = now;
    }
}
```

OUTPUT:

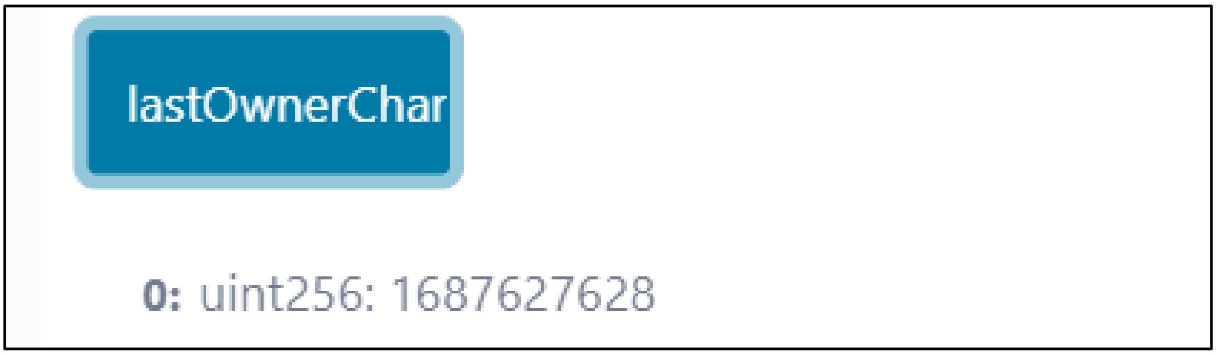

ACCESSRESTRICTION AT 0X86E...E722
Balance: 0 ETH
buyContract
changeOwner
address _newOwner
lastOwnerChar
owner

Step 1: Click on owner to create the owner object.



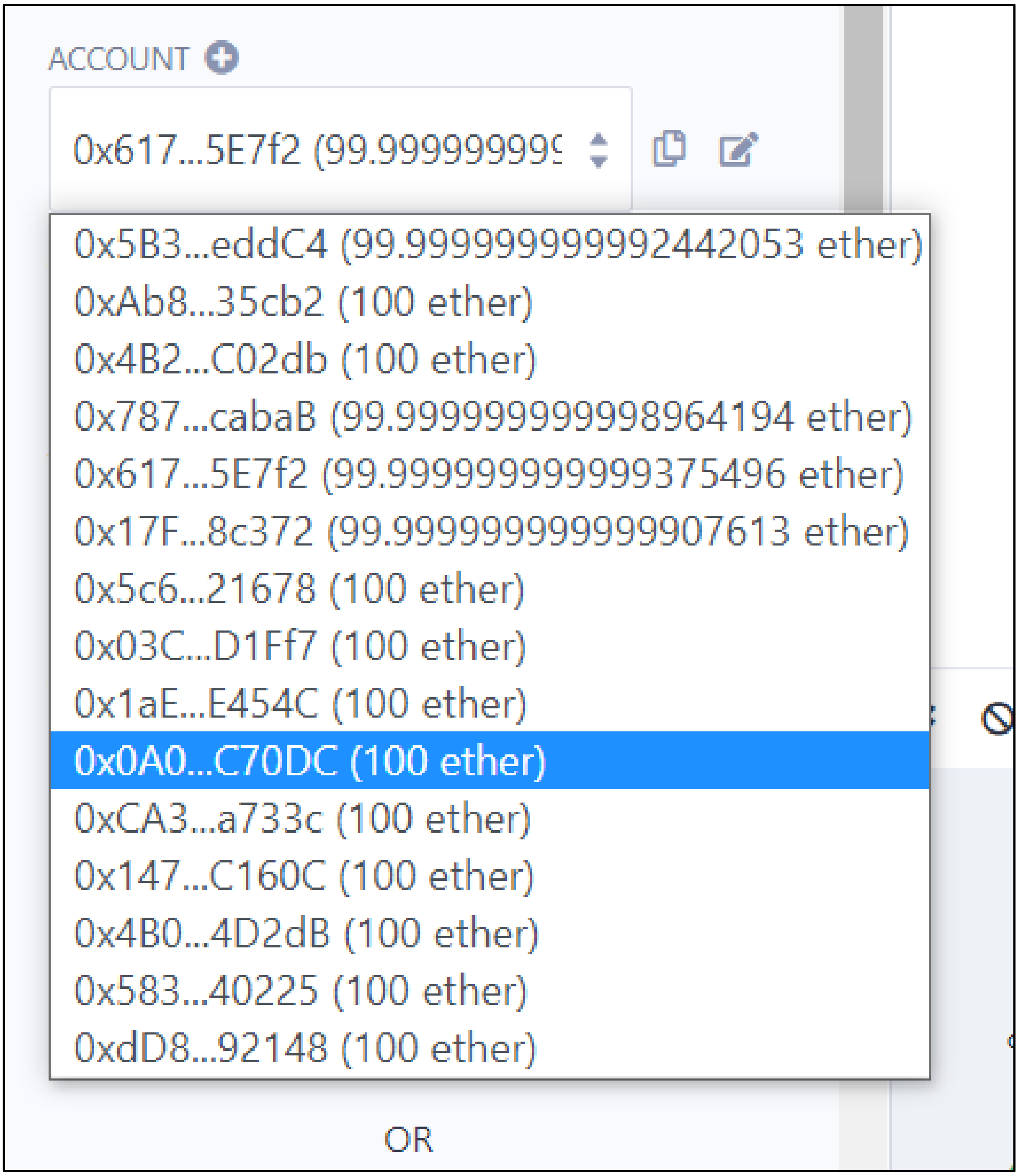
0: address: 0x617F2E2fD72FD9D5503197092aC168c91465E7f2

Step 2: Click on lastOwnerChange button.



0: uint256: 1687627628

Step 3: Change the address of the account from Account dropdown in Deploy tab of Remix IDE.



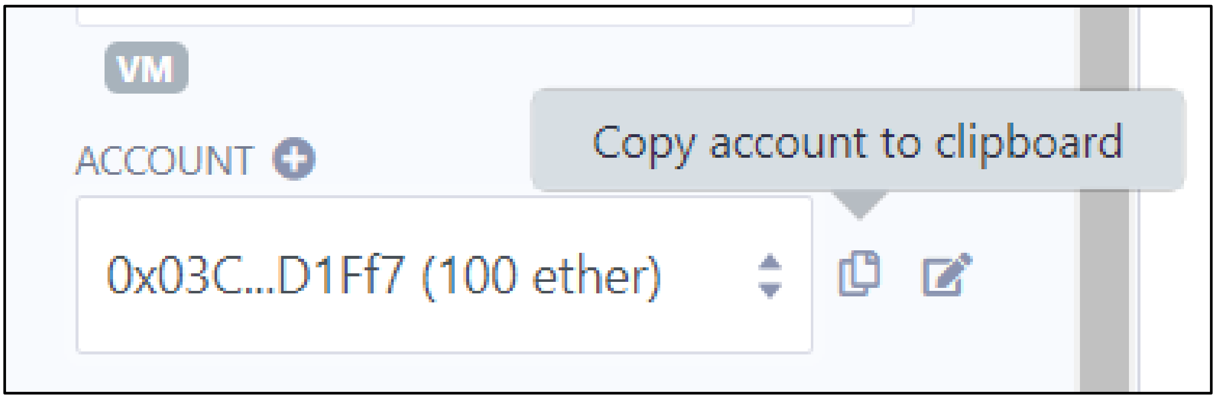
ACCOUNT

0x617...5E7f2 (99.99999999

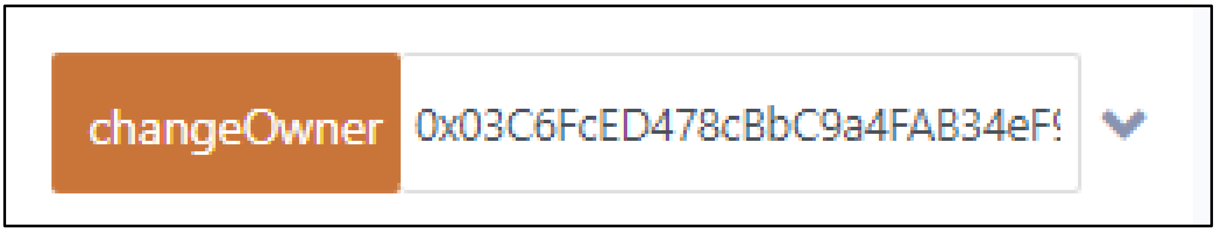
0x5B3...eddC4 (99.9999999999992442053 ether)
0xAb8...35cb2 (100 ether)
0x4B2...C02db (100 ether)
0x787...cabaB (99.9999999999998964194 ether)
0x617...5E7f2 (99.9999999999999375496 ether)
0x17F...8c372 (99.9999999999999907613 ether)
0x5c6...21678 (100 ether)
0x03C...D1Ff7 (100 ether)
0x1aE...E454C (100 ether)
0x0A0...C70DC (100 ether)
0xCA3...a733c (100 ether)
0x147...C160C (100 ether)
0x4B0...4D2dB (100 ether)
0x583...40225 (100 ether)
0xdD8...92148 (100 ether)

OR

Step 4: Copy and paste the address in changeOwner input and click on changeOwner.

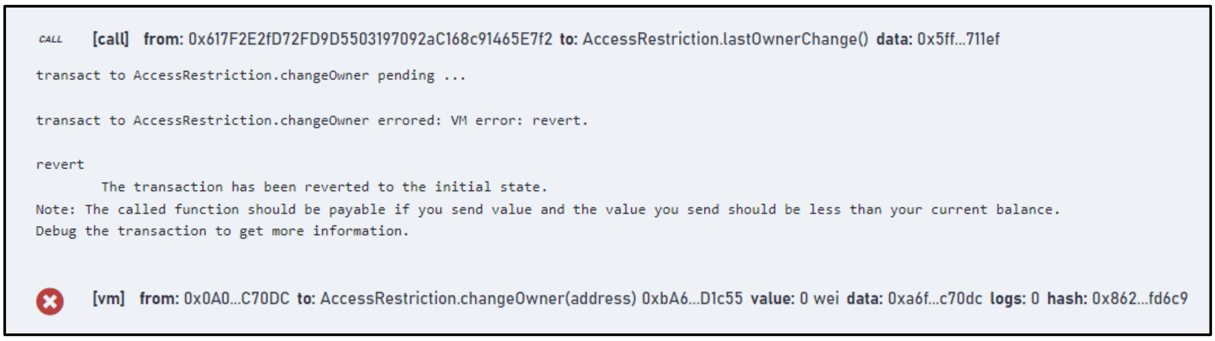


ACCOUNT: 0x03C...D1Ff7 (100 ether)



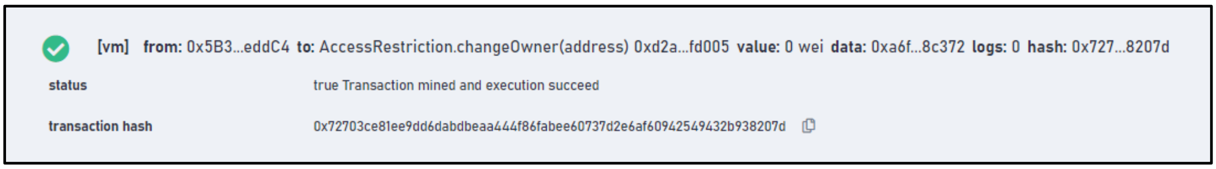
changeOwner: 0x03C6FcED478cBbC9a4FAB34eF!

Step 5: You should get an error as following.

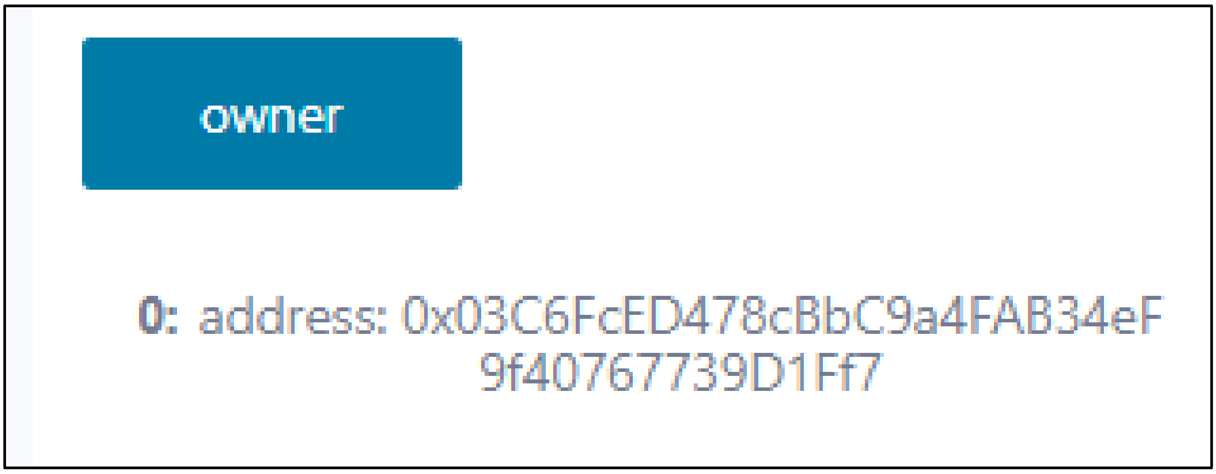


```
CALL   [call] from: 0x617F2E2fD72FD9D5503197092aC168c91465E7f2 to: AccessRestriction.lastOwnerChange() data: 0x5ff...711ef
transact to AccessRestriction.changeOwner pending ...

transact to AccessRestriction.changeOwner errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
Debug the transaction to get more information.

       [vm] from: 0x0A0...C70DC to: AccessRestriction.changeOwner(address) 0xbA6...D1c55 value: 0 wei data: 0xa6f...c70dc logs: 0 hash: 0x862...fd6c9
```

Step 6: Now, change back to the actual address of the account and click on changeOwner.



```
[vm] from: 0x5B3...eddC4 to: AccessRestriction.changeOwner(address) 0xd2a...fd005 value: 0 wei data: 0xa6f...8c372 logs: 0 hash: 0x727...8207d
status              true Transaction mined and execution succeed
transaction hash    0x72703ce81ee9dd6dabdbeaa444f86fabee60737d2e6af60942549432b938207d
```



```
0: address: 0x03C6FcED478cBbC9a4FAB34eF
9f40767739D1Ff7
```

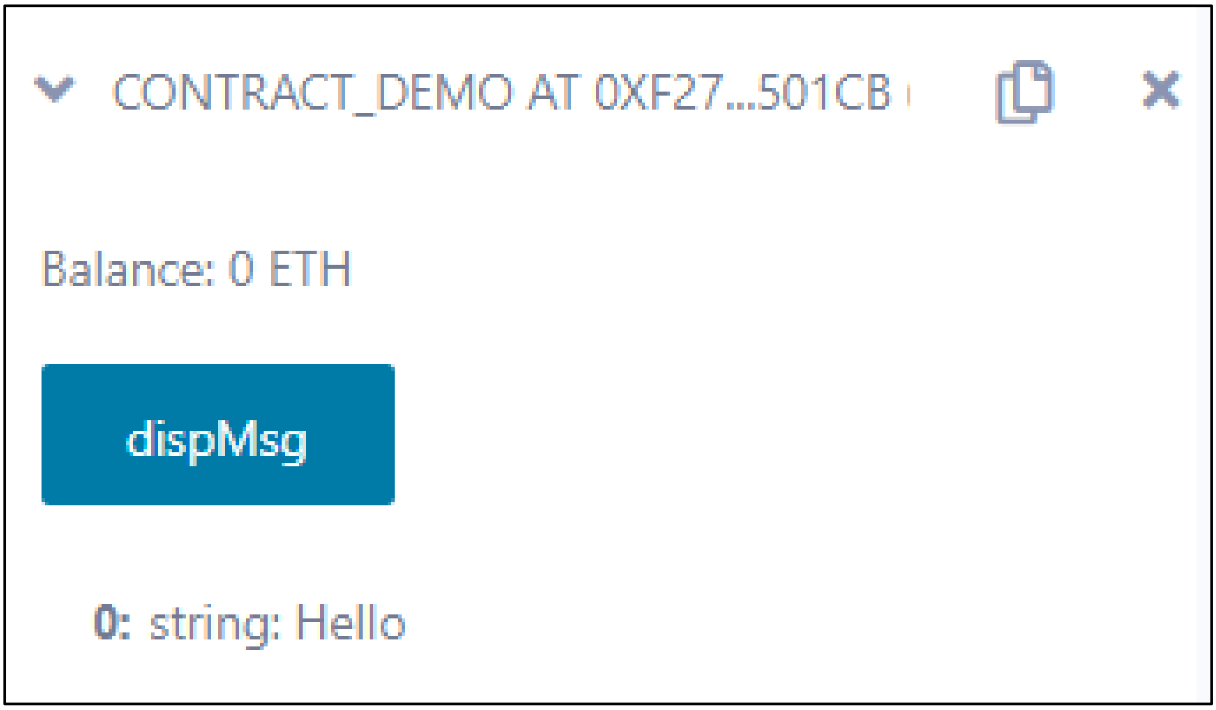
[B] Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.

[I] Contracts

CODE:

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

contract Contract_demo {
  string message = "Hello";

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

OUTPUT:



0: string: Hello

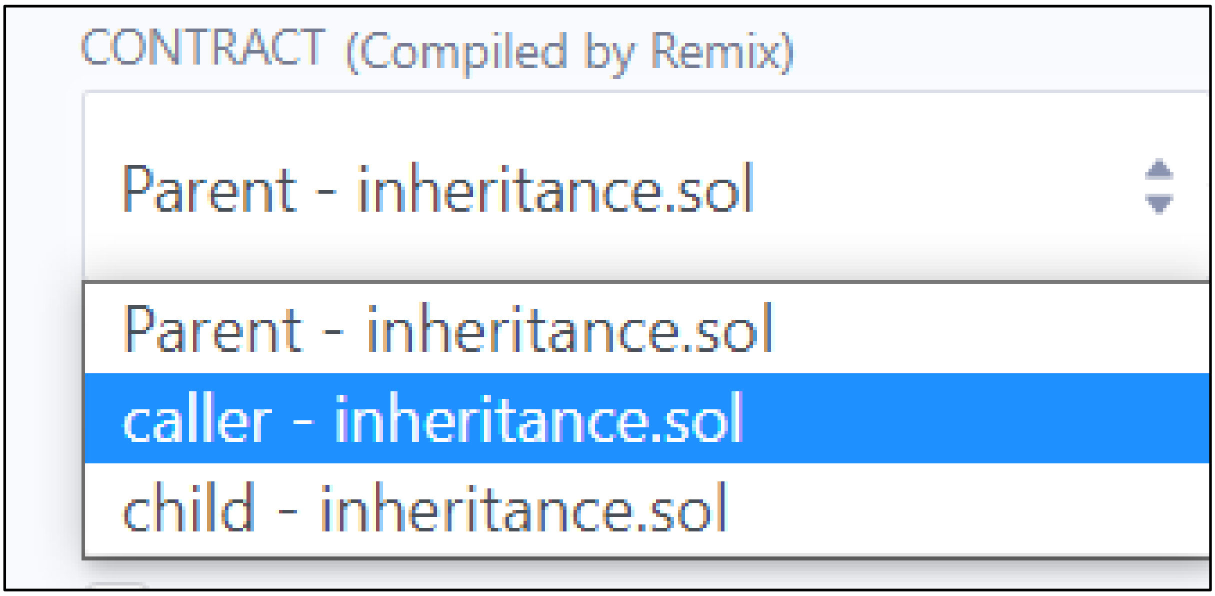
[II] Inheritance

CODE:

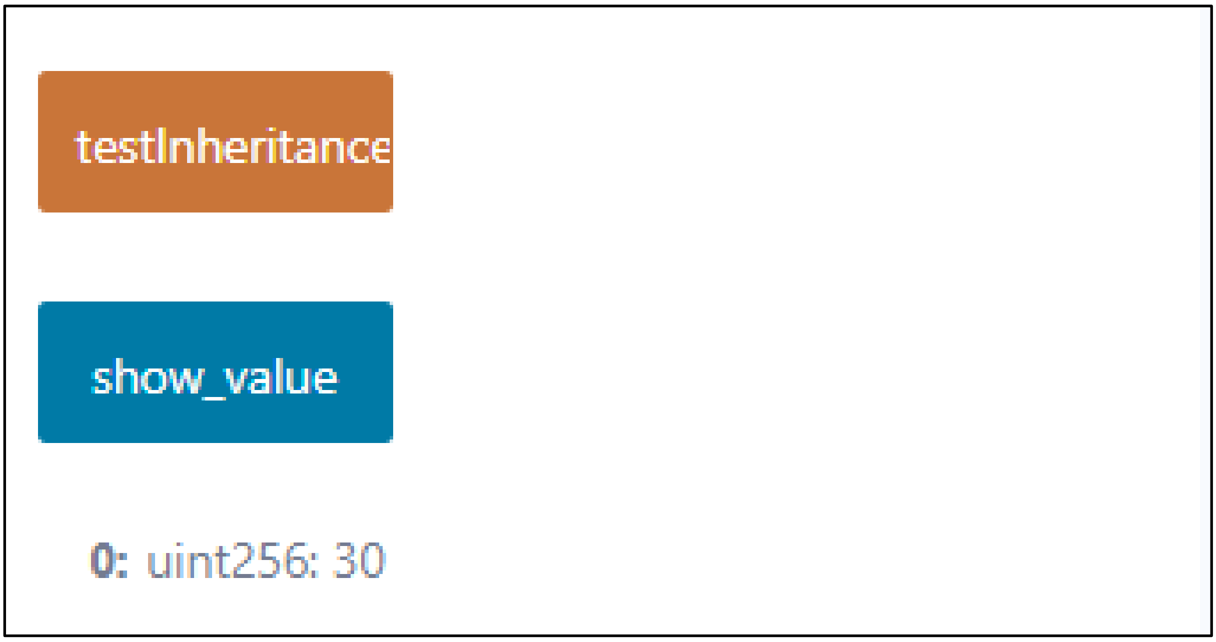
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

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 show_value() public view returns (uint256) {
    return cc.getValue();
  }
}
```

OUTPUT:

Step 1: Select caller contract and deploy.



Step 2: Click test Inheritance and then click on show_value to view value.



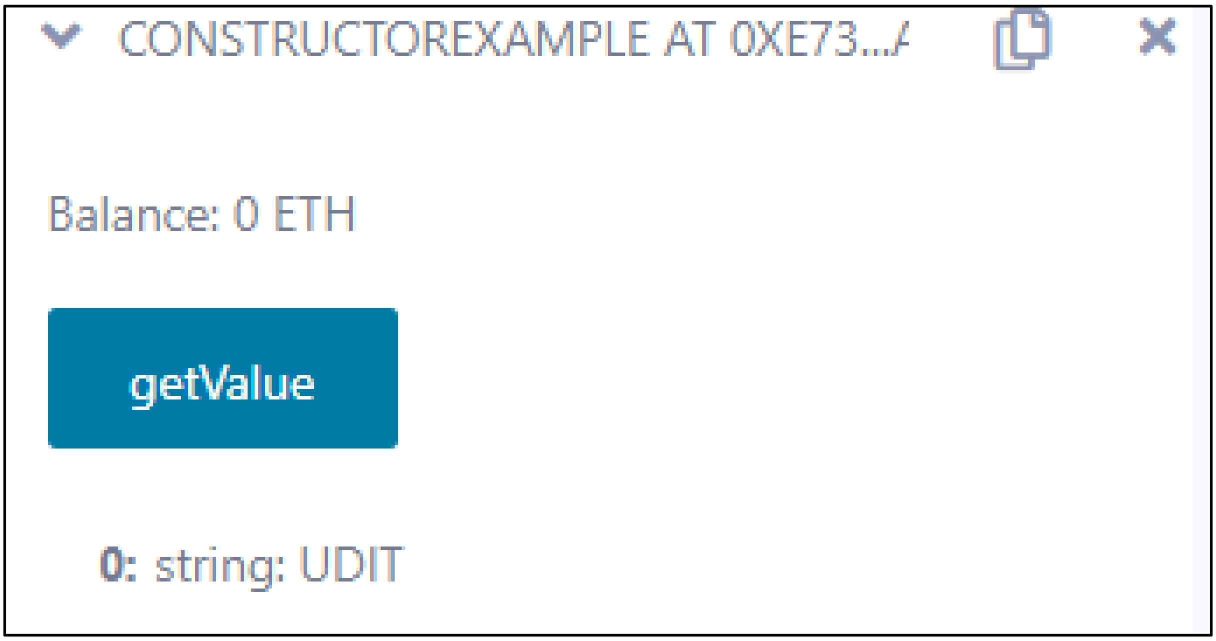
```
0: uint256: 30
```

[III] Constructors

CODE:

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

// Creating a contract
contract constructorExample {
  string str;

  constructor() public {
    str = "UDIT";
  }

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

OUTPUT:



Balance: 0 ETH

0: string: UDIT

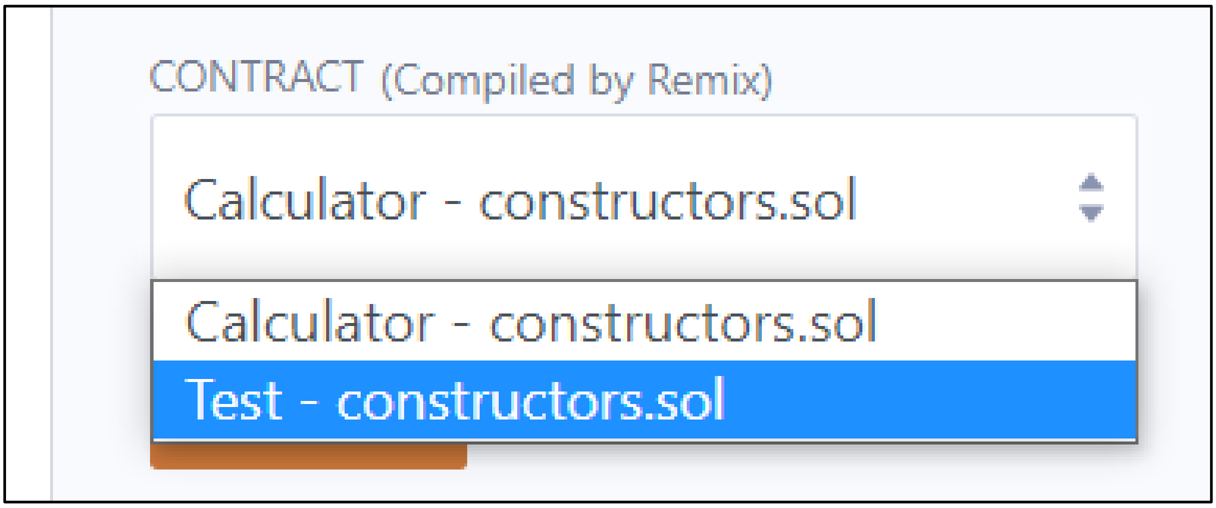
[IV] Abstract Contracts

CODE:

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

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:

Step 1: Select Test contract and deploy.



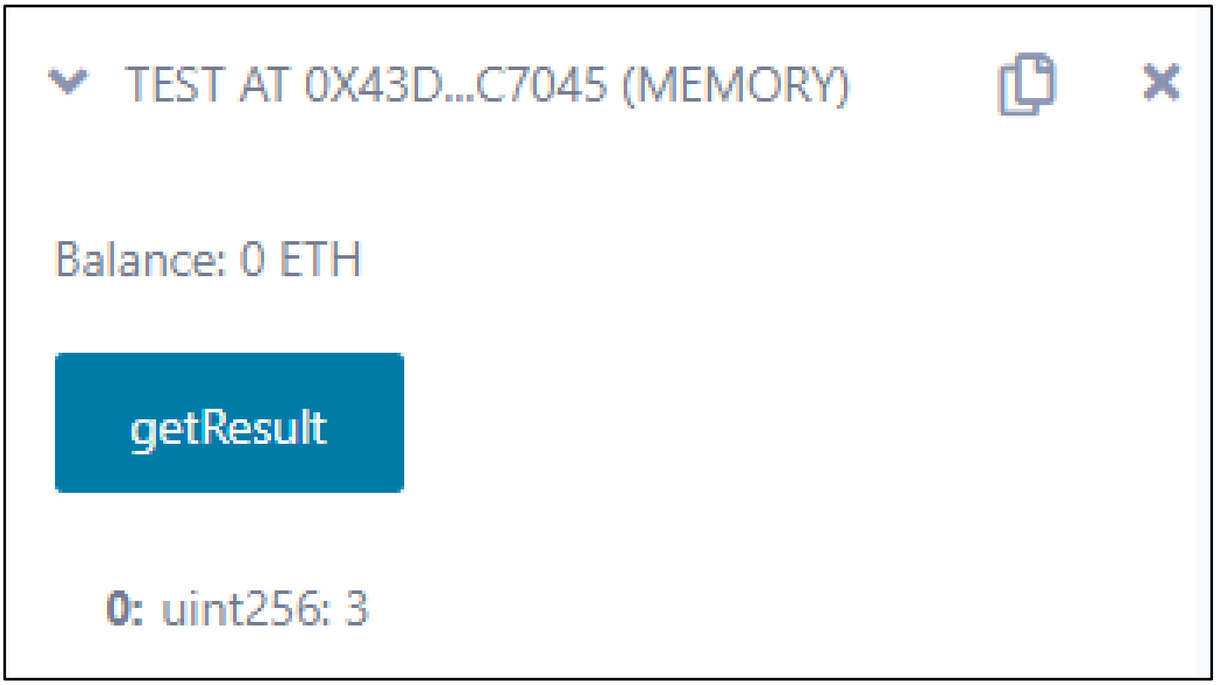
CONTRACT (Compiled by Remix)

Calculator - constructors.sol

Calculator - constructors.sol

Test - constructors.sol

Step 2: Click on getResult to get sum of a+b.



TEST AT 0X43D...C7045 (MEMORY)

Balance: 0 ETH

```
0: uint256: 3
```

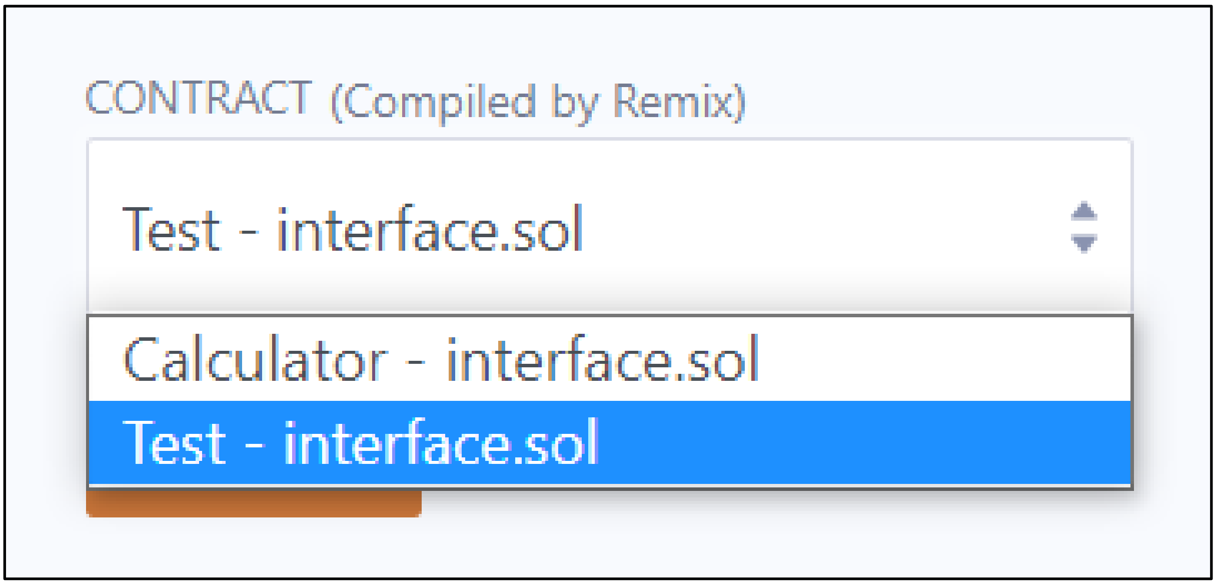
[V] Interfaces

CODE:

```
// SPDX-License-Identifier: MIT
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:

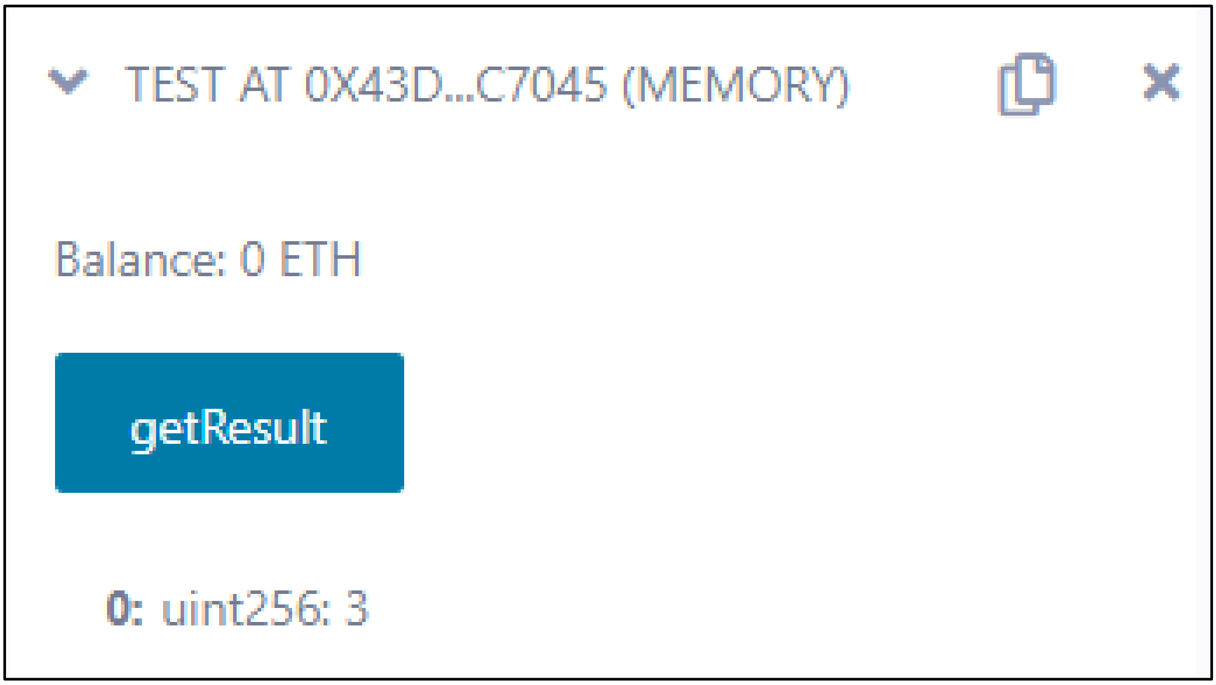
Step 1: Select Test interface contract and deploy.



CONTRACT (Compiled by Remix)

Test - interface.sol

Step 2: Click on getResult to get sum of a+b.



TEST AT 0X43D...C7045 (MEMORY)

Balance: 0 ETH

0: uint256: 3

[C] Libraries, Assembly, Events, Error handling.

[I] Libraries

CODE:

Create library myLib.sol

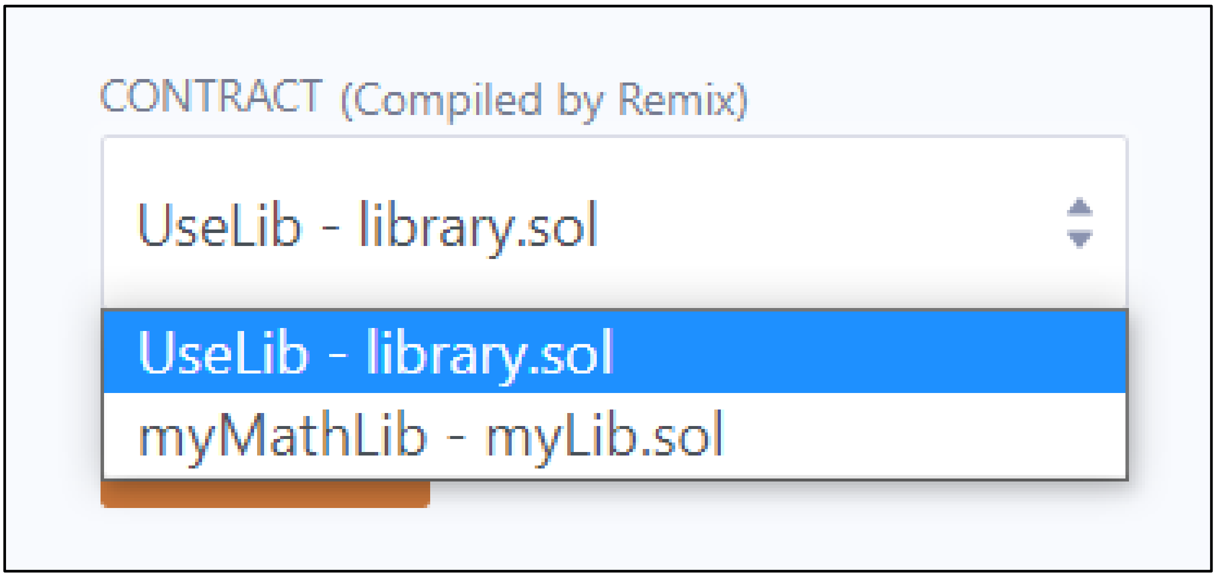
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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;
  }
}
```

Using the library myLib.sol

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

import "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:

Step 1: Change contract to UseLib and deploy.



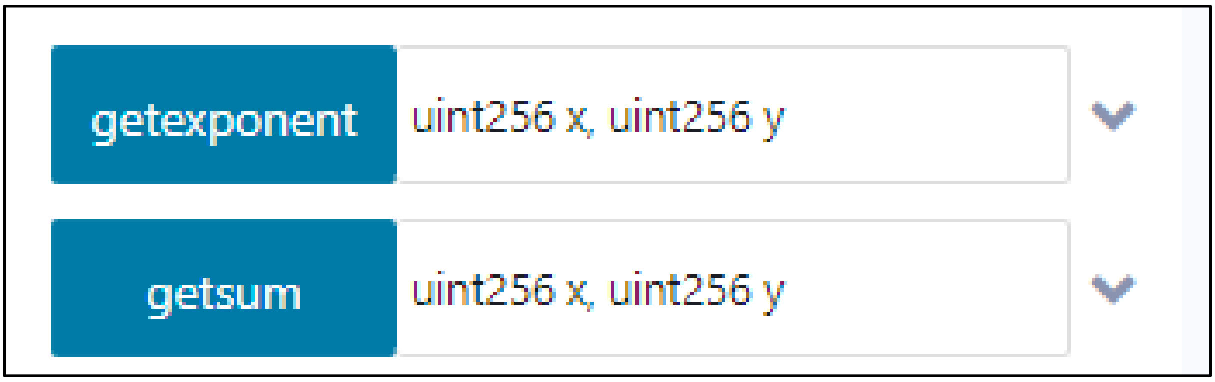
CONTRACT (Compiled by Remix)

UseLib - library.sol

UseLib - library.sol

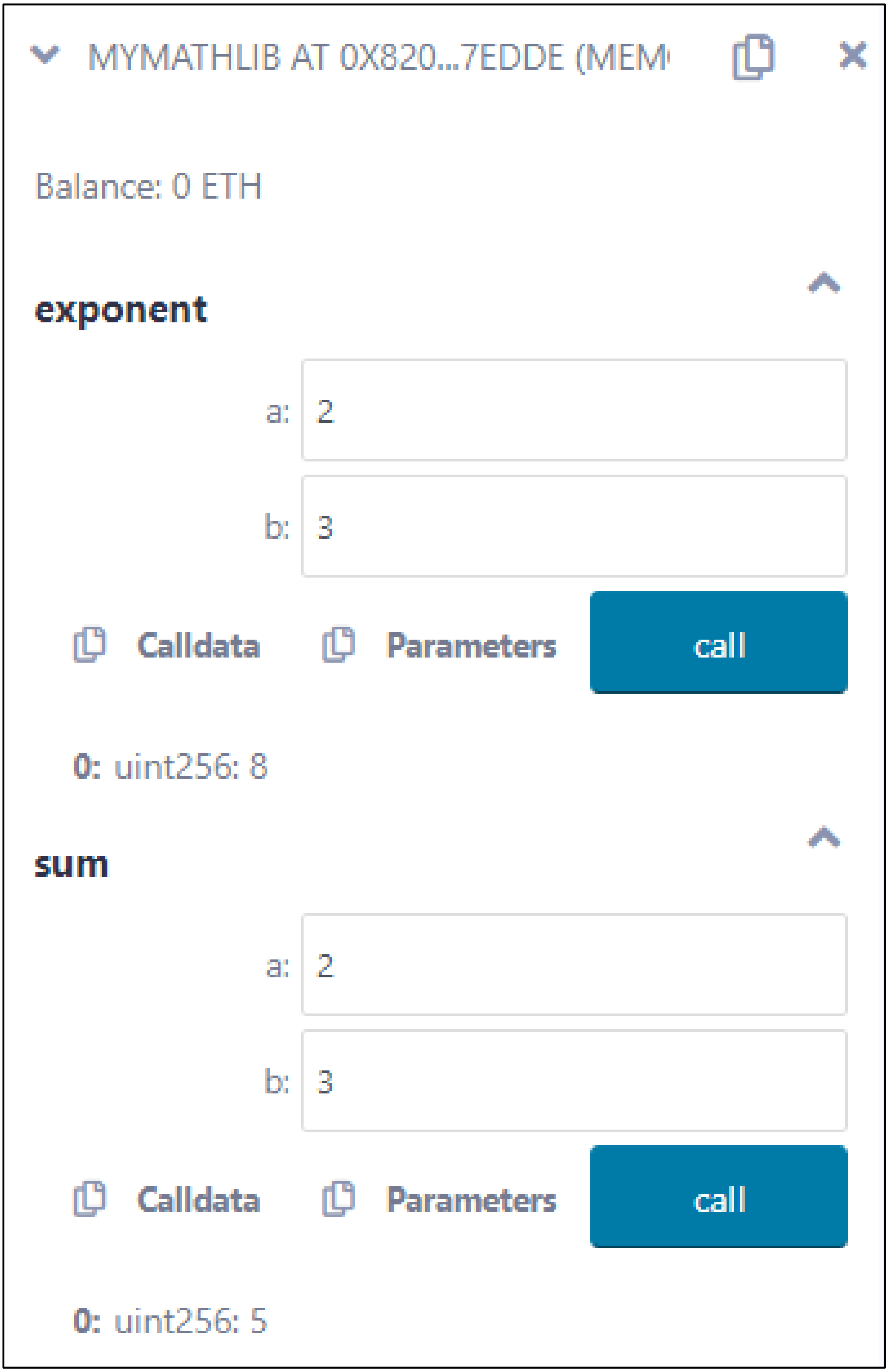
myMathLib - myLib.sol

Step 2: Input values to both getexponent and getsum functions and get their values as below.



getexponent uint256 x, uint256 y

getsum uint256 x, uint256 y



MYMATHLIB AT 0X820...7EDDE (MEM

Balance: 0 ETH

exponent

a: 2

b: 3

0: uint256: 8

sum

a: 2

b: 3

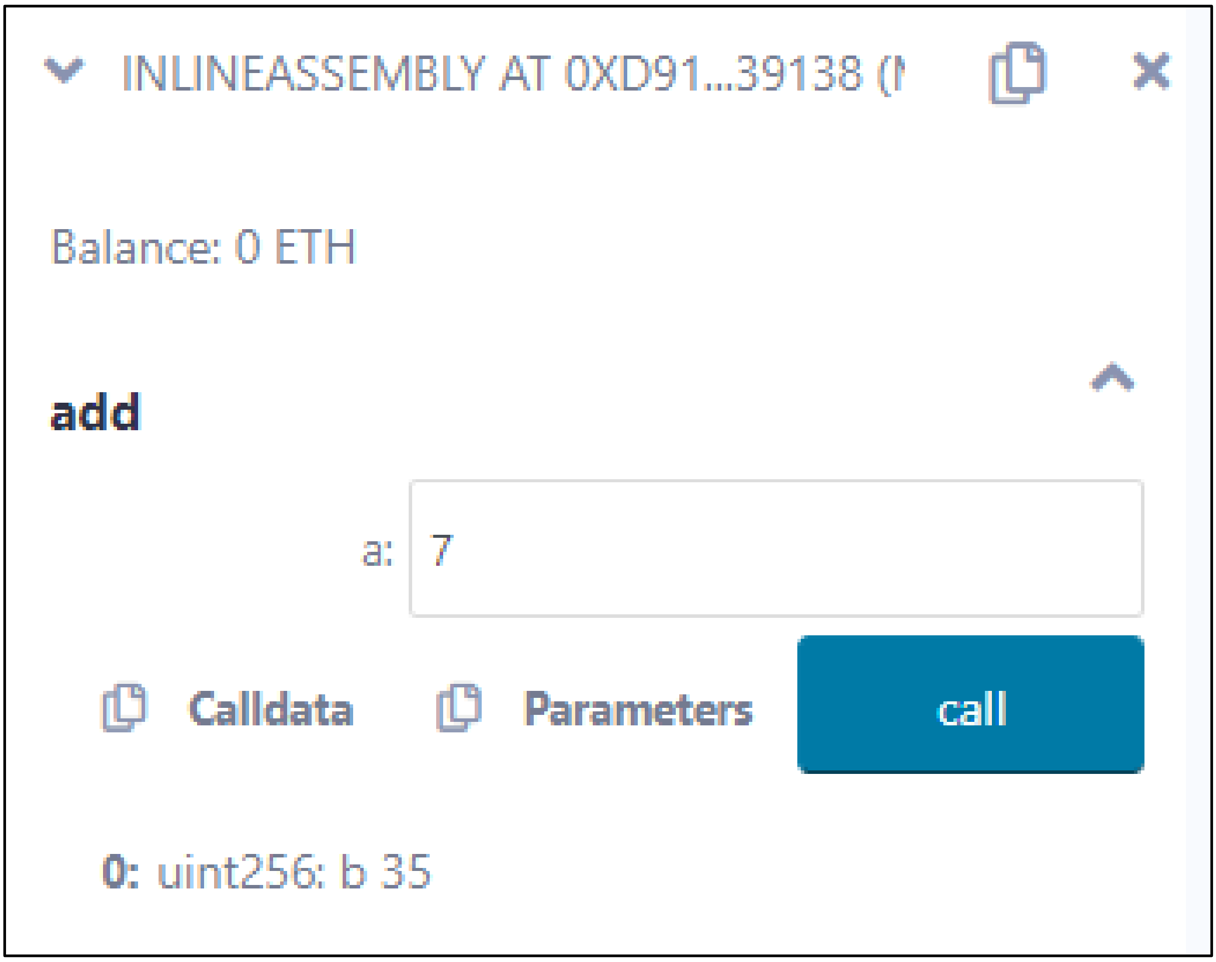
0: uint256: 5

[II] Assembly

CODE:

```
// SPDX-License-Identifier: MIT
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

add

a: 7

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

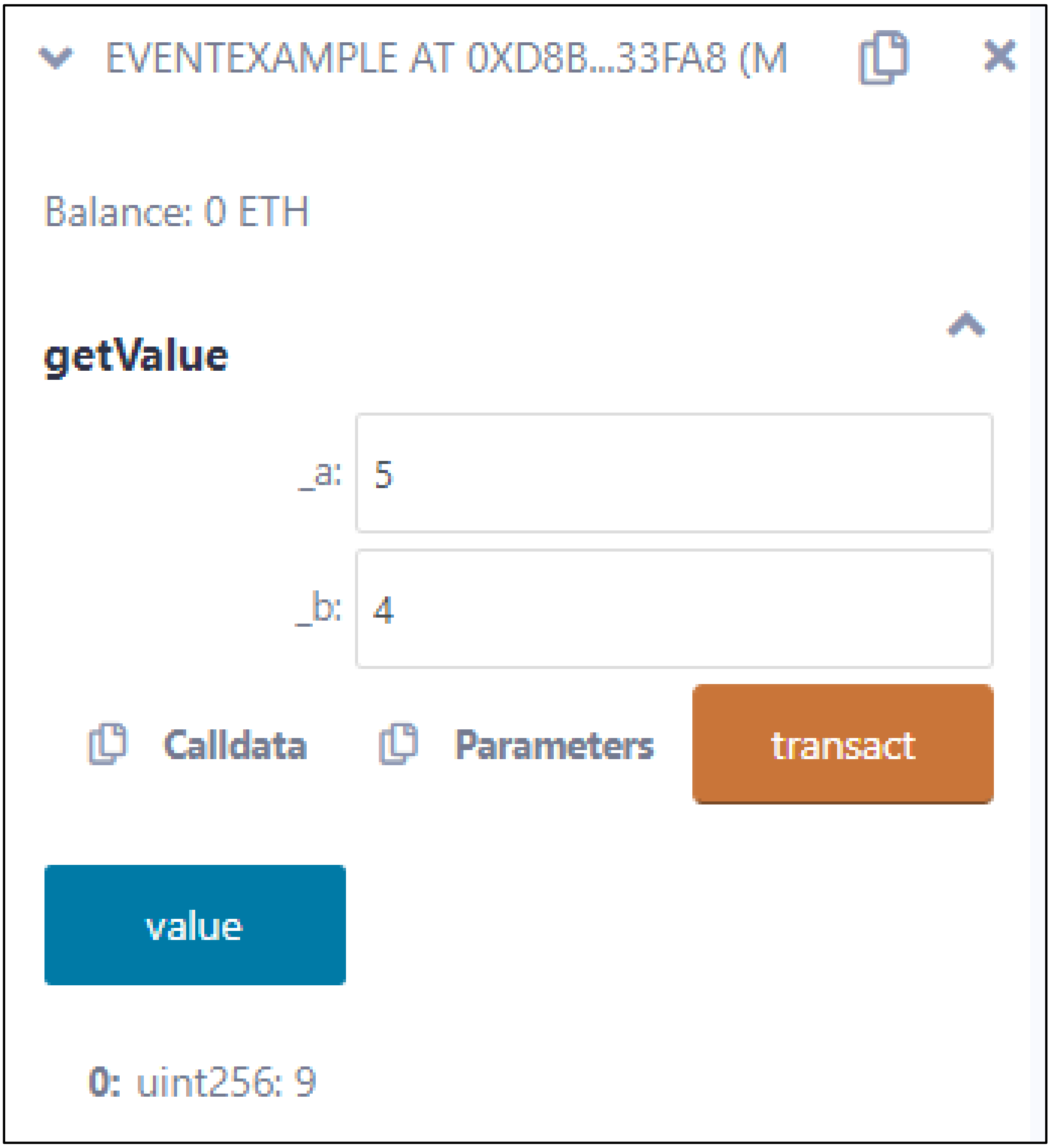
[III] Events

CODE:

```
// 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);
    value = _a + _b;
  }
}
```

OUTPUT:

Step 1: Input values to getValue function and get the result by clicking on the value button.



EVENTEXAMPLE AT 0XD8B...33FA8 (M

Balance: 0 ETH

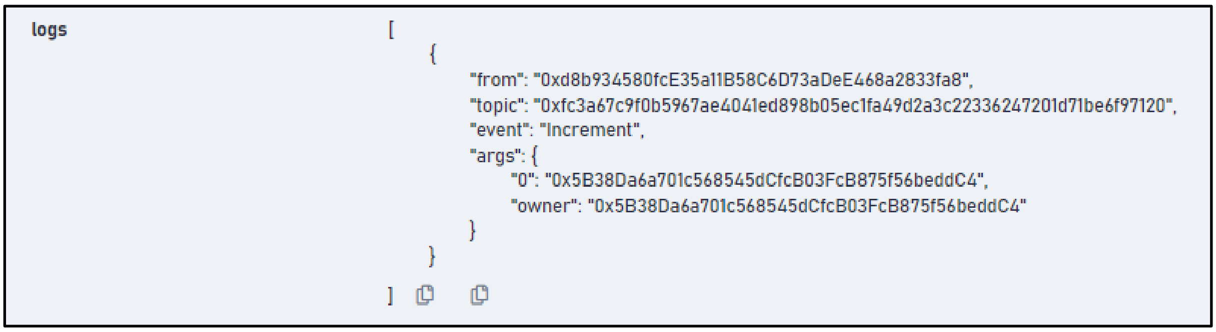
getValue

_a: 5

_b: 4

0: uint256: 9

Step 2: In the terminal, check for logs.



```
logs                                [
                                        {
                                            "from": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
                                            "topic": "0xfc3a67c9f0b5967ae4041ed898b05ec1fa49d2a3c22336247201d71be6f97120",
                                            "event": "Increment",
                                            "args": {
                                                "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "owner": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                            }
                                        }
                                    ]
```

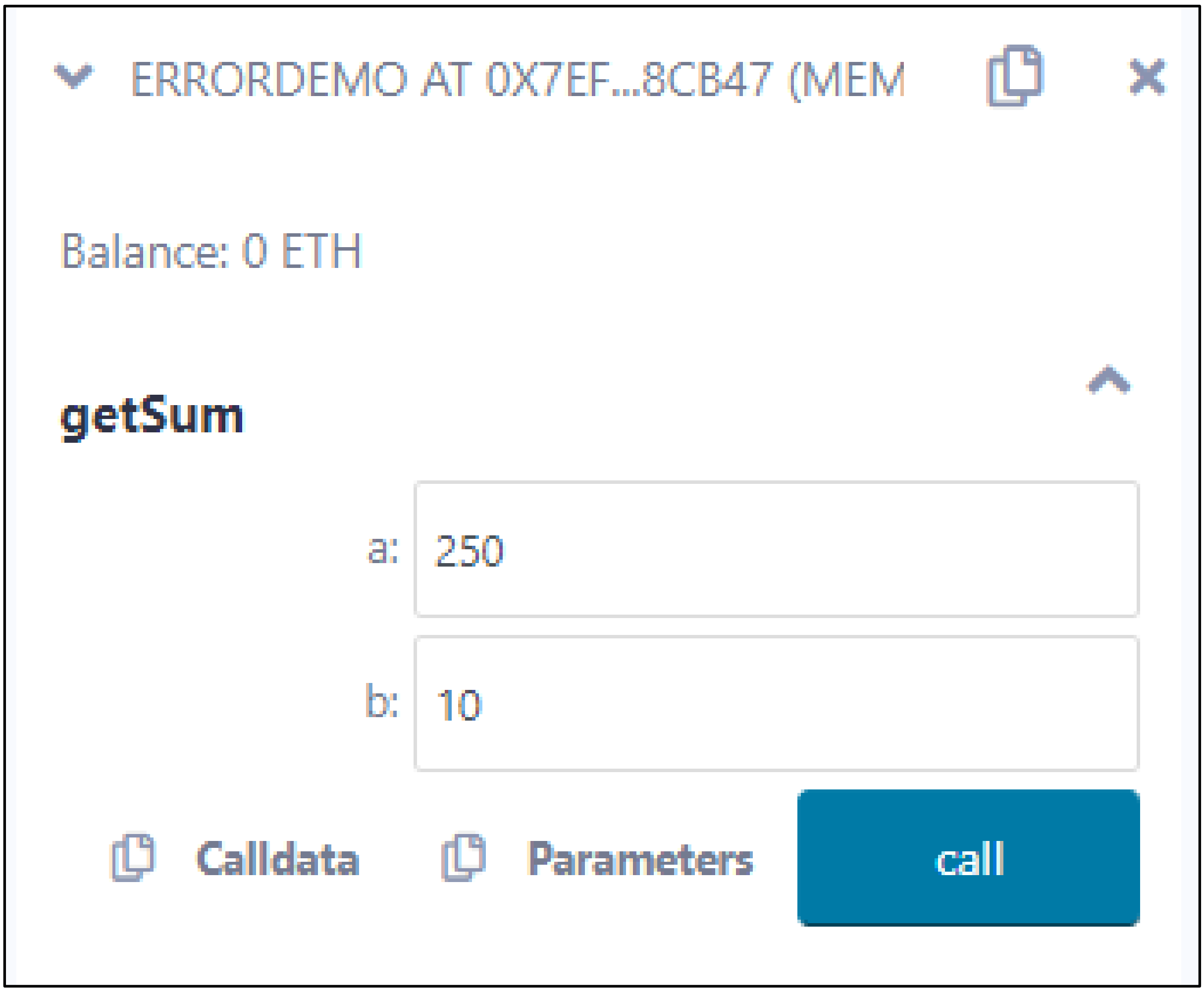
[IV] Error handling

CODE:

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

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:

Step 1: Input values to the getSum function.



ERRORDEMO AT 0X7EF...8CB47 (MEM

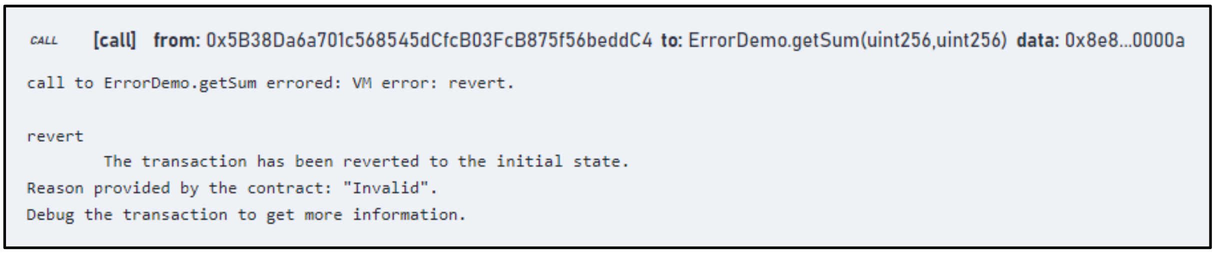
Balance: 0 ETH

getSum

a: 250

b: 10

Step 2: Check terminal panel.



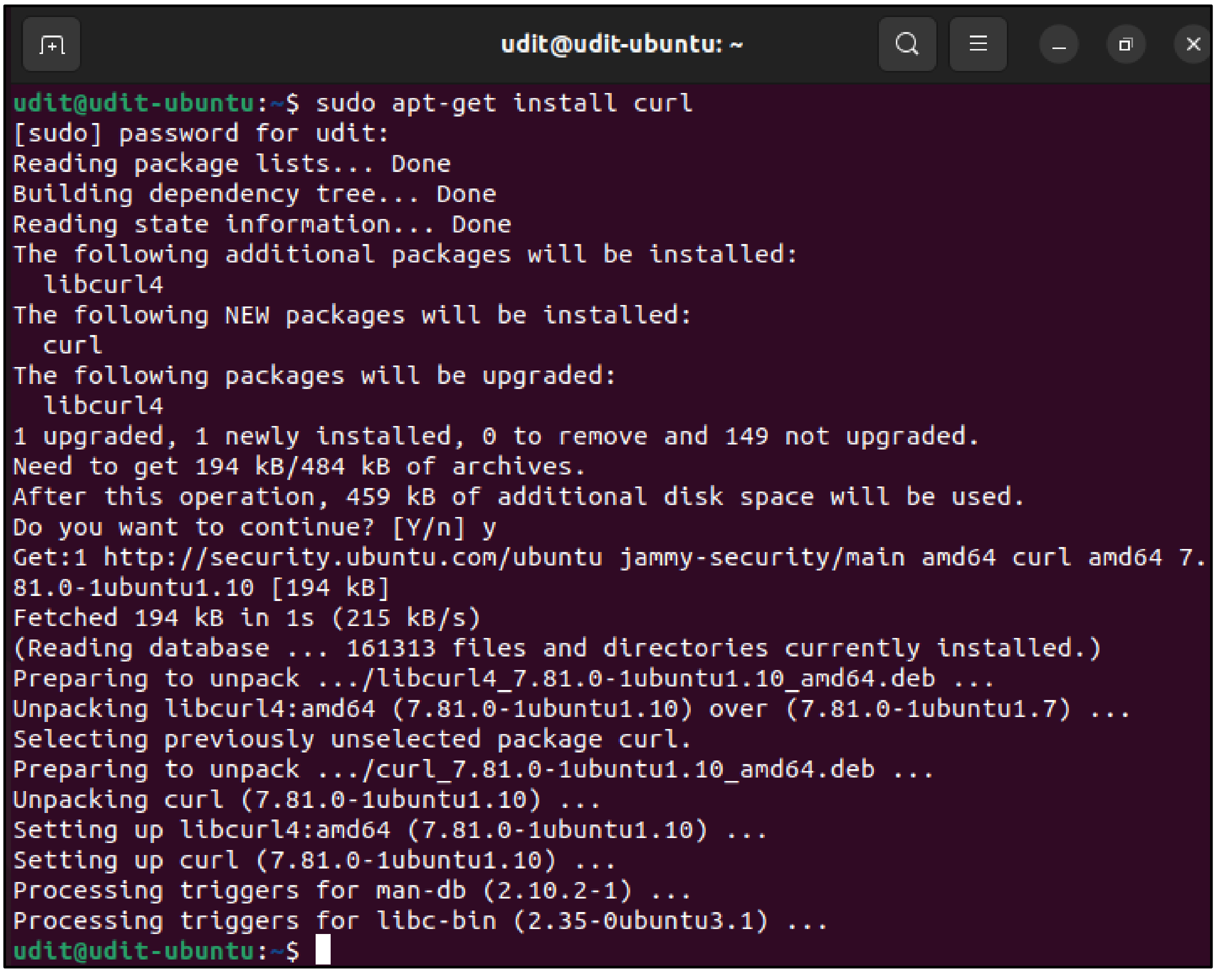
```
CALL    [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorDemo.getSum(uint256,uint256) data: 0x8e8...0000a
call to ErrorDemo.getSum errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Invalid".
Debug the transaction to get more information.
```

PRACTICAL 5

AIM: Install hyperledger fabric and composer. Deploy and execute the application.

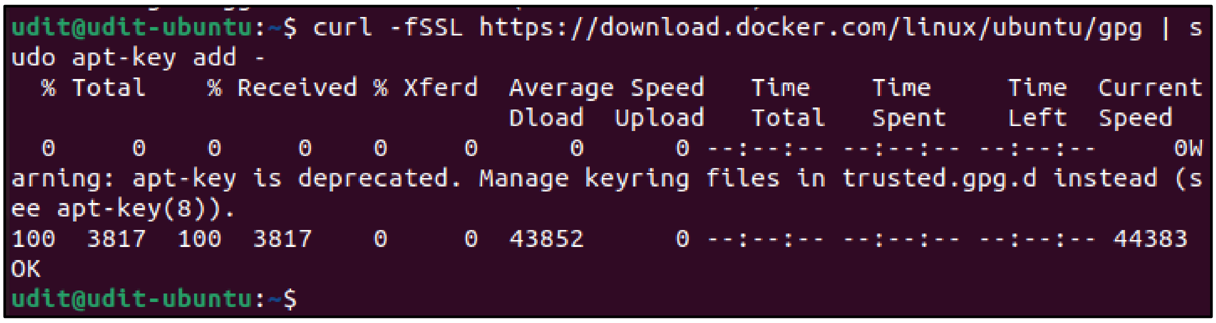
Step 1: Install Docker on Ubuntu

sudo apt-get install curl



```
udit@udit-ubuntu:~$ sudo apt-get install curl
[sudo] password for udit:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl
The following packages will be upgraded:
  libcurl4
1 upgraded, 1 newly installed, 0 to remove and 149 not upgraded.
Need to get 194 kB/484 kB of archives.
After this operation, 459 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://security.ubuntu.com/ubuntu jammy-security/main amd64 curl amd64 7.
81.0-1ubuntu1.10 [194 kB]
Fetched 194 kB in 1s (215 kB/s)
(Reading database ... 161313 files and directories currently installed.)
Preparing to unpack .../libcurl4_7.81.0-1ubuntu1.10_amd64.deb ...
Unpacking libcurl4:amd64 (7.81.0-1ubuntu1.10) over (7.81.0-1ubuntu1.7) ...
Selecting previously unselected package curl.
Preparing to unpack .../curl_7.81.0-1ubuntu1.10_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.10) ...
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.10) ...
Setting up curl (7.81.0-1ubuntu1.10) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
udit@udit-ubuntu:~$
```

curl -fSSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –



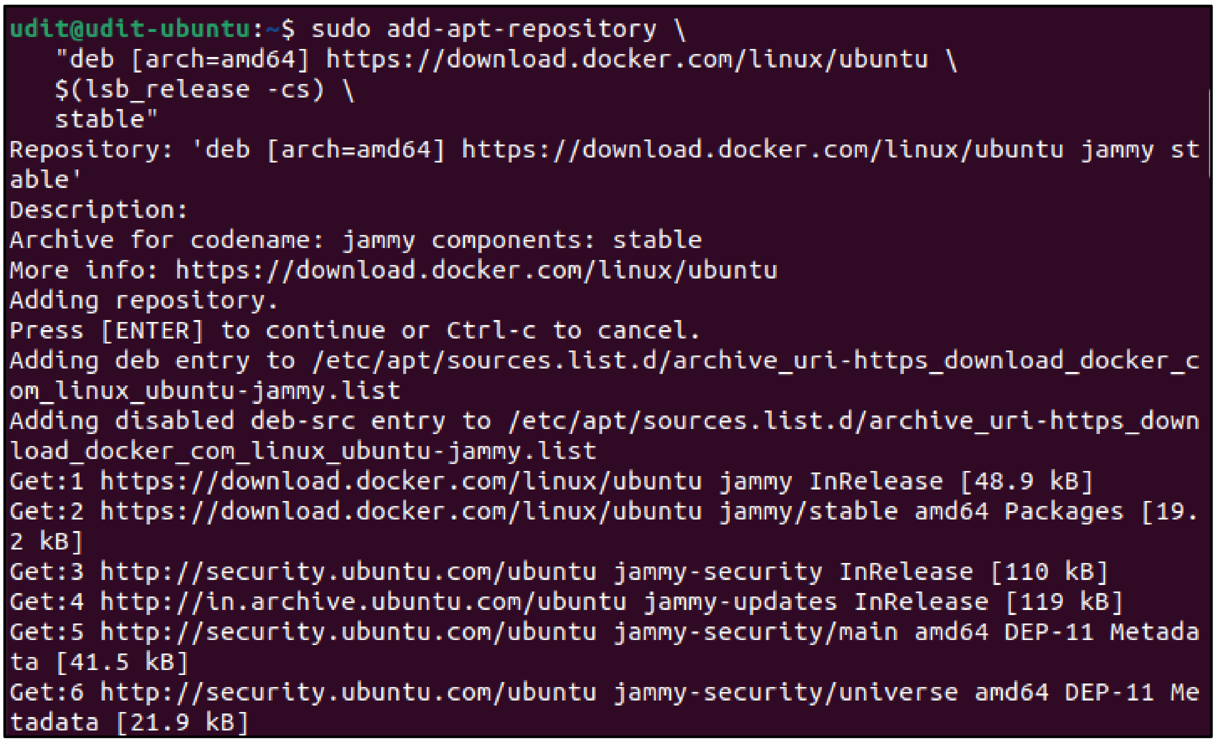
```
udit@udit-ubuntu:~$ curl -fSSL https://download.docker.com/linux/ubuntu/gpg | s
udo apt-key add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0W
arning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (s
ee apt-key(8)).
100  3817  100  3817    0     0  43852      0 --:--:-- --:--:-- --:--:-- 44383
OK
udit@udit-ubuntu:~$
```

sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

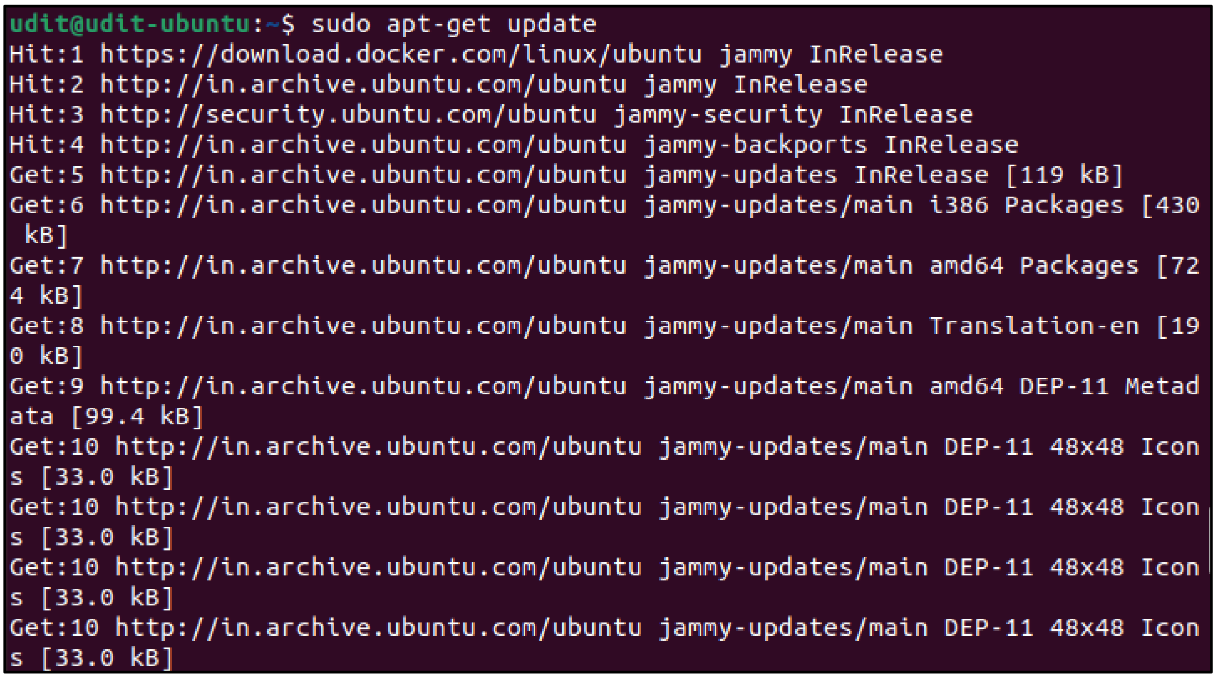
$(lsb_release -cs) \

stable"



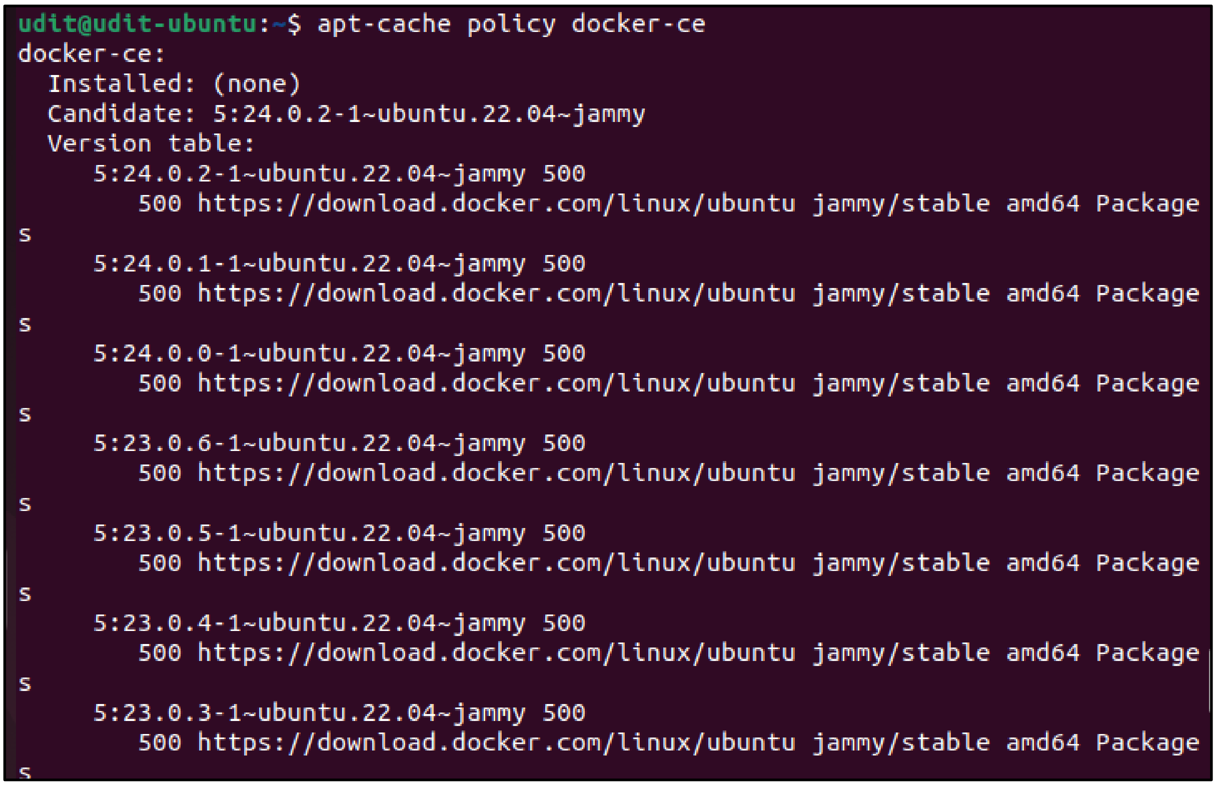
```
udit@udit-ubuntu:~$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu jammy st
able'
Description:
Archive for codename: jammy components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_c
om_linux_ubuntu-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_down
load_docker_com_linux_ubuntu-jammy.list
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.9 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [19.
2 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metada
ta [41.5 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Me
tadata [21.9 kB]
```

sudo apt-get update



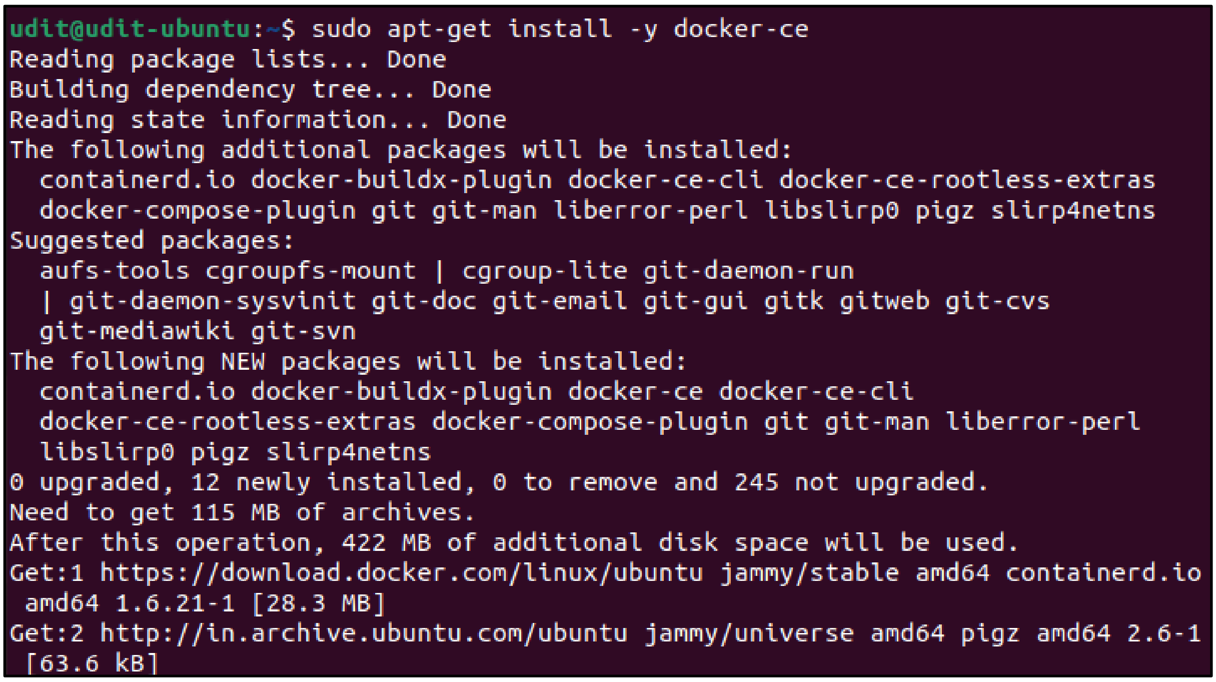
```
udit@udit-ubuntu:~$ sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [430
 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [72
4 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [19
0 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metad
ata [99.4 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 48x48 Icon
s [33.0 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 48x48 Icon
s [33.0 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 48x48 Icon
s [33.0 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 48x48 Icon
s [33.0 kB]
```

apt-cache policy docker-ce



```
udit@udit-ubuntu:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:24.0.2-1~ubuntu.22.04~jammy
  Version table:
     5:24.0.2-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Package
s
     5:24.0.1-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Package
s
     5:24.0.0-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Package
s
     5:23.0.6-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Package
s
     5:23.0.5-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Package
s
     5:23.0.4-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Package
s
     5:23.0.3-1~ubuntu.22.04~jammy 500
        500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Package
s
```

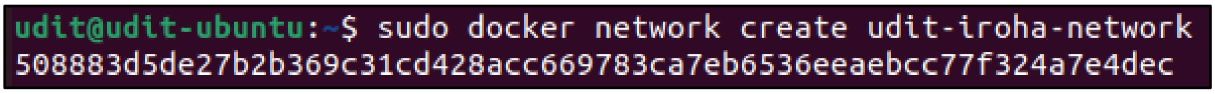
sudo apt-get install -y docker-ce



```
udit@udit-ubuntu:~$ sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin git git-man liberror-perl libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run
  | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli
  docker-ce-rootless-extras docker-compose-plugin git git-man liberror-perl
  libslirp0 pigz slirp4netns
0 upgraded, 12 newly installed, 0 to remove and 245 not upgraded.
Need to get 115 MB of archives.
After this operation, 422 MB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io
 amd64 1.6.21-1 [28.3 MB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1
 [63.6 kB]
```

Step 2: Create a docker network

sudo docker network create udit-iroha-network



```
udit@udit-ubuntu:~$ sudo docker network create udit-iroha-network
508883d5de27b2b369c31cd428acc669783ca7eb6536eeaebcc77f324a7e4dec
```

Step 3: Add PostgreSQL to the created docker network.

sudo docker run --name some-postgres \

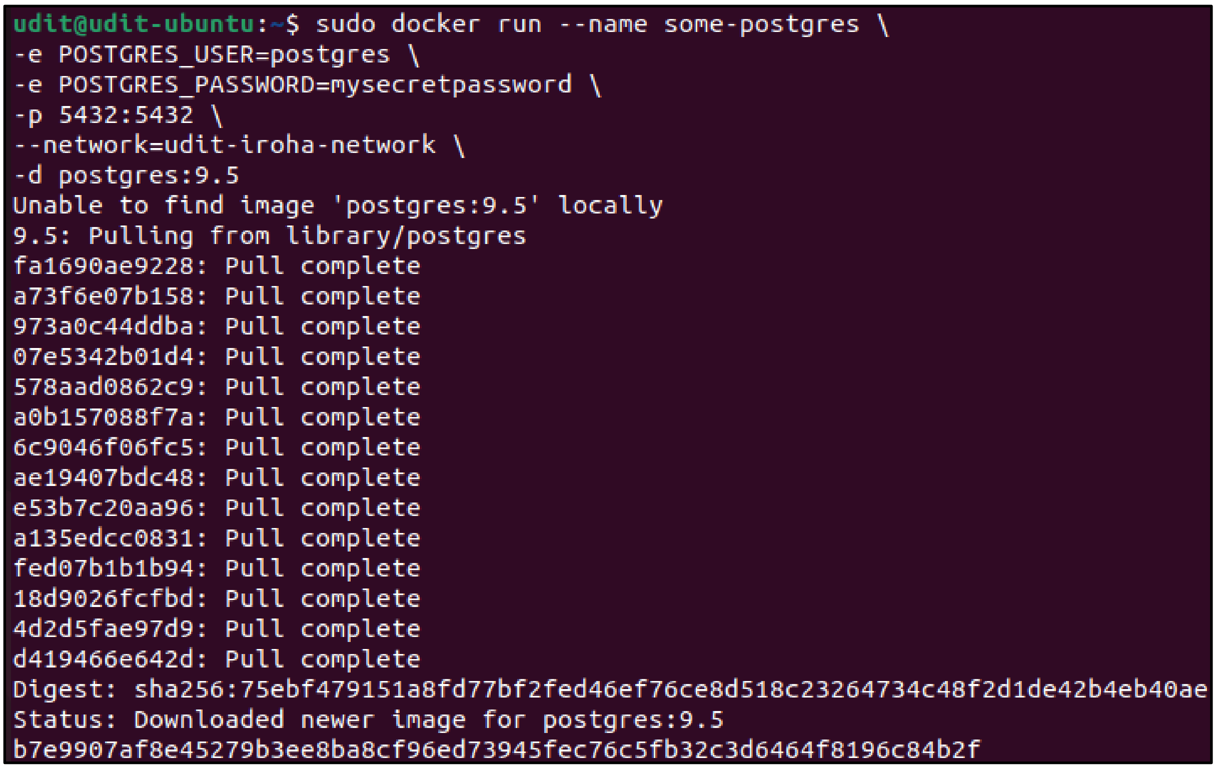
-e POSTGRES_USER=postgres \

-e POSTGRES_PASSWORD=mysecretpassword \

-p 5432:5432 \

--network=udit-iroha-network \

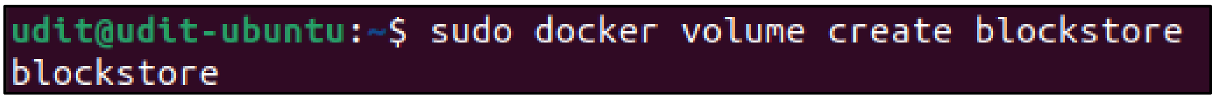
-d postgres:9.5



```
udit@udit-ubuntu:~$ sudo docker run --name some-postgres \
-e POSTGRES_USER=postgres \
-e POSTGRES_PASSWORD=mysecretpassword \
-p 5432:5432 \
--network=udit-iroha-network \
-d postgres:9.5
Unable to find image 'postgres:9.5' locally
9.5: Pulling from library/postgres
fa1690ae9228: Pull complete
a73f6e07b158: Pull complete
973a0c44ddba: Pull complete
07e5342b01d4: Pull complete
578aad0862c9: Pull complete
a0b157088f7a: Pull complete
6c9046f06fc5: Pull complete
ae19407bdc48: Pull complete
e53b7c20aa96: Pull complete
a135edcc0831: Pull complete
fed07b1b1b94: Pull complete
18d9026fcfbd: Pull complete
4d2d5fae97d9: Pull complete
d419466e642d: Pull complete
Digest: sha256:75ebf479151a8fd77bf2fed46ef76ce8d518c23264734c48f2d1de42b4eb40ae
Status: Downloaded newer image for postgres:9.5
b7e9907af8e45279b3ee8ba8cf96ed73945fec76c5fb32c3d6464f8196c84b2f
```

Step 4: Create a volume of persistant storage named "blockstore" to store the blocks on blockchain.

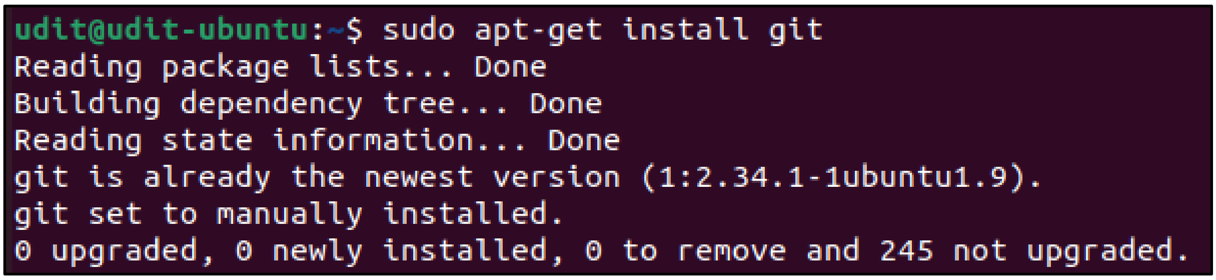
sudo docker volume create blockstore



```
udit@udit-ubuntu:~$ sudo docker volume create blockstore
blockstore
```

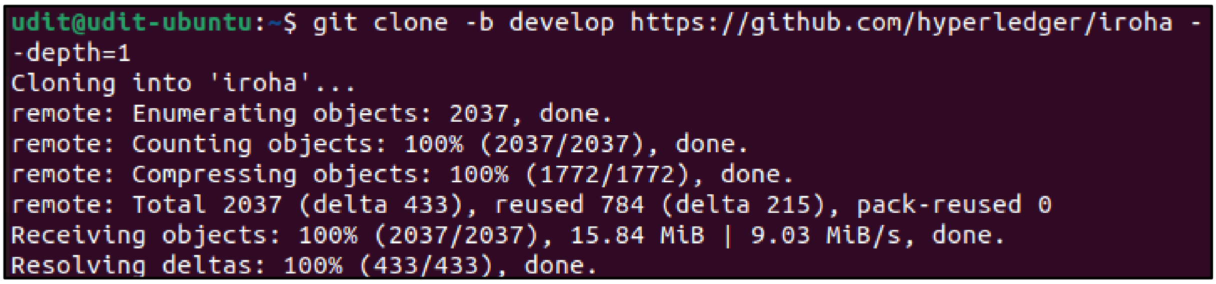
Step 5: Configure Iroha on the network. Download the Iroha code from github and install git.

sudo apt-get install git



```
udit@udit-ubuntu:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.9).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 245 not upgraded.
```

git clone -b develop https://github.com/hyperledger/iroha --depth=1



```
udit@udit-ubuntu:~$ git clone -b develop https://github.com/hyperledger/iroha -
-depth=1
Cloning into 'iroha'...
remote: Enumerating objects: 2037, done.
remote: Counting objects: 100% (2037/2037), done.
remote: Compressing objects: 100% (1772/1772), done.
remote: Total 2037 (delta 433), reused 784 (delta 215), pack-reused 0
Receiving objects: 100% (2037/2037), 15.84 MiB | 9.03 MiB/s, done.
Resolving deltas: 100% (433/433), done.
```

Step 6: Run the Iroha docker container.

sudo docker run -it --name iroha \

-p 50051:50051 \

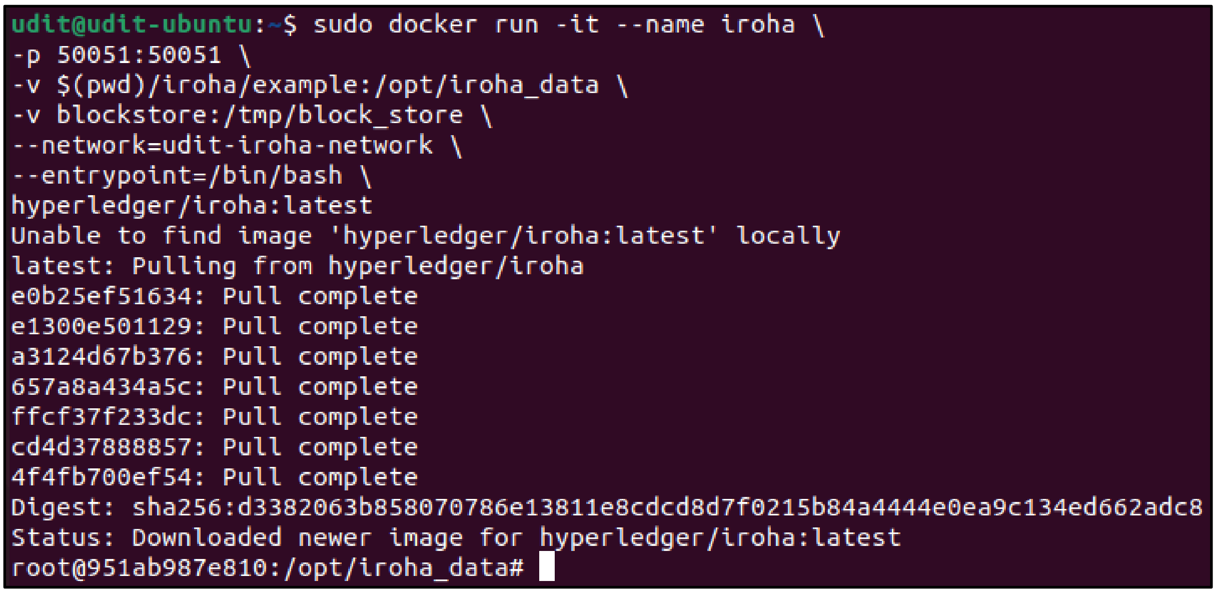
-v $(pwd)/iroha/example:/opt/iroha_data \

-v blockstore:/tmp/block_store \

--network=udit-iroha-network \

--entrypoint=/bin/bash \

hyperledger/iroha:latest



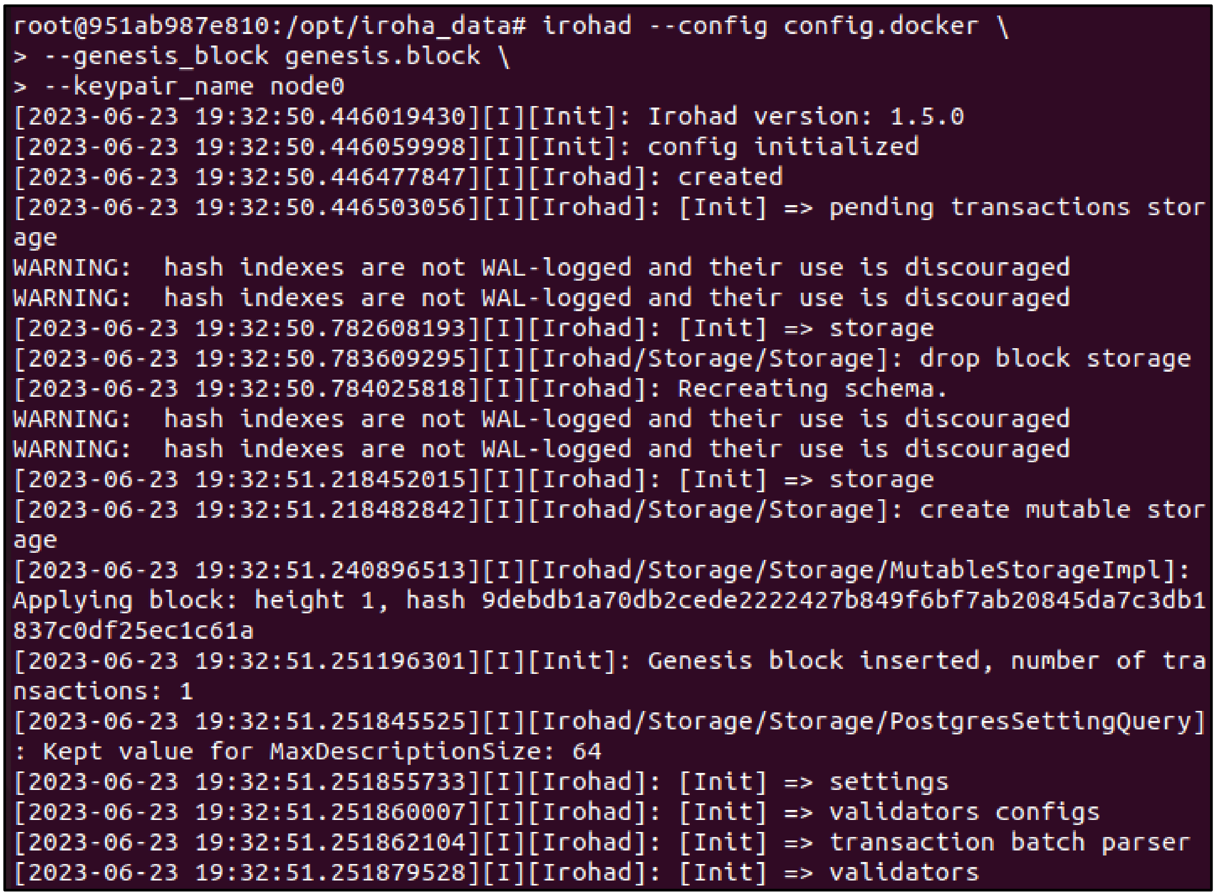
```
udit@udit-ubuntu:~$ sudo docker run -it --name iroha \
-p 50051:50051 \
-v $(pwd)/iroha/example:/opt/iroha_data \
-v blockstore:/tmp/block_store \
--network=udit-iroha-network \
--entrypoint=/bin/bash \
hyperledger/iroha:latest
Unable to find image 'hyperledger/iroha:latest' locally
latest: Pulling from hyperledger/iroha
e0b25ef51634: Pull complete
e1300e501129: Pull complete
a3124d67b376: Pull complete
657a8a434a5c: Pull complete
ffcf37f233dc: Pull complete
cd4d37888857: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:d3382063b858070786e13811e8cdcd8d7f0215b84a4444e0ea9c134ed662adc8
Status: Downloaded newer image for hyperledger/iroha:latest
root@951ab987e810:/opt/iroha_data# 
```

Step 7: Run Iroha.

irohad --config config.docker \

--genesis_block genesis.block \

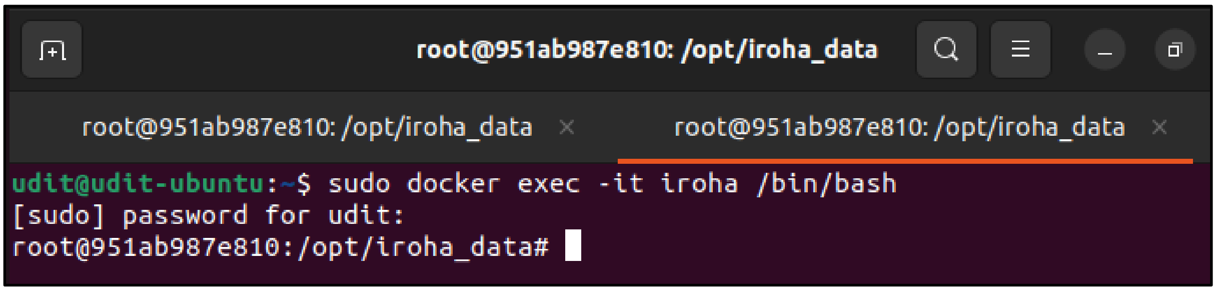
--keypair_name node0



```
root@951ab987e810:/opt/iroha_data# irohad --config config.docker \
> --genesis_block genesis.block \
> --keypair_name node0
[2023-06-23 19:32:50.446019430][I][Init]: Irohad version: 1.5.0
[2023-06-23 19:32:50.446059998][I][Init]: config initialized
[2023-06-23 19:32:50.446477847][I][Irohad]: created
[2023-06-23 19:32:50.446503056][I][Irohad]: [Init] => pending transactions stor
age
WARNING:  hash indexes are not WAL-logged and their use is discouraged
WARNING:  hash indexes are not WAL-logged and their use is discouraged
[2023-06-23 19:32:50.782608193][I][Irohad]: [Init] => storage
[2023-06-23 19:32:50.783609295][I][Irohad/Storage/Storage]: drop block storage
[2023-06-23 19:32:50.784025818][I][Irohad]: Recreating schema.
WARNING:  hash indexes are not WAL-logged and their use is discouraged
WARNING:  hash indexes are not WAL-logged and their use is discouraged
[2023-06-23 19:32:51.218452015][I][Irohad]: [Init] => storage
[2023-06-23 19:32:51.218482842][I][Irohad/Storage/Storage]: create mutable stor
age
[2023-06-23 19:32:51.240896513][I][Irohad/Storage/Storage/MutableStorageImpl]:
Applying block: height 1, hash 9debdb1a70db2cede2222427b849f6bf7ab20845da7c3db1
837c0df25ec1c61a
[2023-06-23 19:32:51.251196301][I][Init]: Genesis block inserted, number of tra
nsactions: 1
[2023-06-23 19:32:51.251845525][I][Irohad/Storage/Storage/PostgresSettingQuery]
: Kept value for MaxDescriptionSize: 64
[2023-06-23 19:32:51.251855733][I][Irohad]: [Init] => settings
[2023-06-23 19:32:51.251860007][I][Irohad]: [Init] => validators configs
[2023-06-23 19:32:51.251862104][I][Irohad]: [Init] => transaction batch parser
[2023-06-23 19:32:51.251879528][I][Irohad]: [Init] => validators
```

Step 8: Open new terminal and attach the docker container to our terminal.

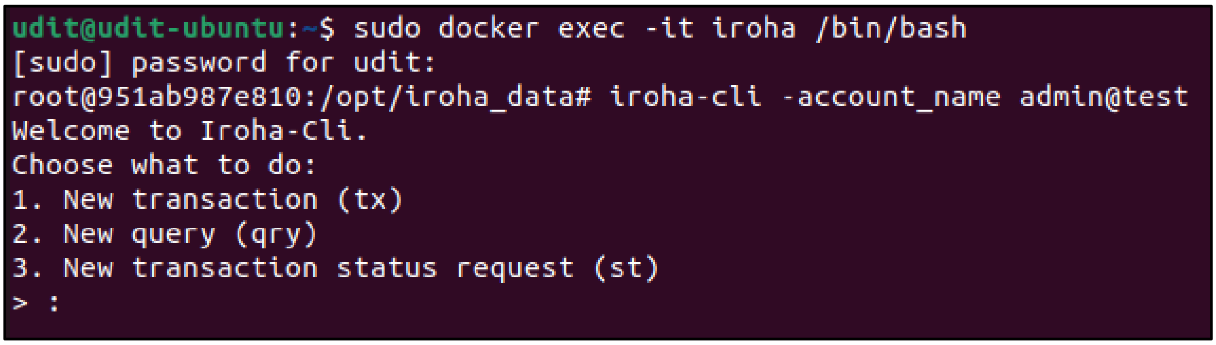
sudo docker exec -it iroha /bin/bash


root@951ab987e810: /opt/iroha_data
root@951ab987e810: /opt/iroha_data
root@951ab987e810: /opt/iroha_data

```
udit@udit-ubuntu:~$ sudo docker exec -it iroha /bin/bash
[sudo] password for udit:
root@951ab987e810:/opt/iroha_data# 
```

Step 9: Launch the iroha-cli tool and login as admin@test.

iroha-cli -account_name admin@test



```
udit@udit-ubuntu:~$ sudo docker exec -it iroha /bin/bash
[sudo] password for udit:
root@951ab987e810:/opt/iroha_data# iroha-cli -account_name admin@test
Welcome to Iroha-Cli.
Choose what to do:
1. New transaction (tx)
2. New query (qry)
3. New transaction status request (st)
> :
```

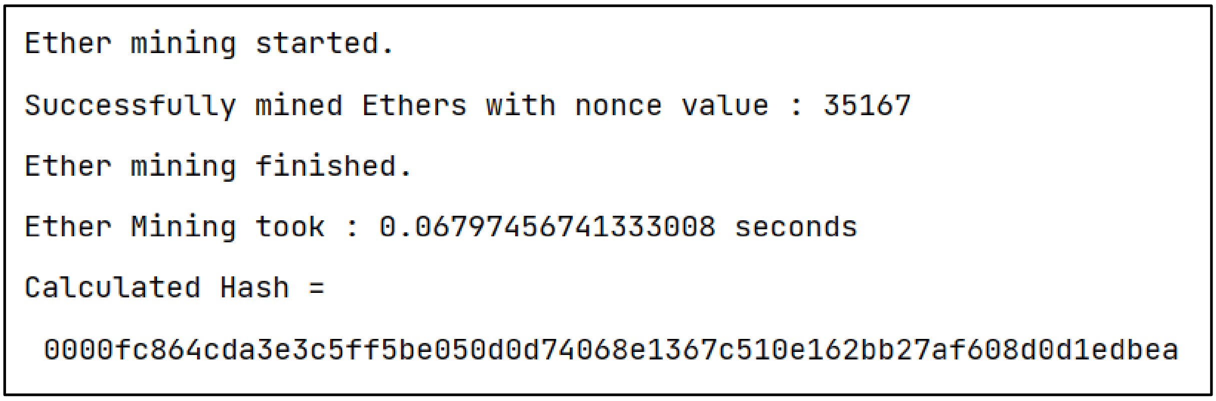
PRACTICAL 6

AIM: Write a program to demonstrate mining of Ether.

CODE:

```
from hashlib import sha256
import time

MAX_NONCE = 100000000000

def hashGenerator(text):
    return sha256(text.encode("ascii")).hexdigest()

def mine(block_number, transactions, previous_hash, prefix_zeros):
    prefix_str = '0'*prefix_zeros
    for nonce in range(MAX_NONCE):
        text = str(block_number) + transactions + previous_hash +
str(nonce)
        new_hash = hashGenerator(text)
        if new_hash.startswith(prefix_str):
            print(f"Successfully mined Ethers with nonce value :
{nonce}")
            return new_hash

    raise BaseException(f"Couldn't find correct hash after trying
{MAX_NONCE} times")

if __name__ == '__main__':
    transactions = '''
    Prateek->Hajra->77,
    Kunal->Soham->18
    '''
    difficulty = 4
    start = time.time()
    print("Ether mining started.")
    new_hash                                                            =
mine(5,transactions,'0000000xa036944e29568d0cff17edbe038f81208fecf9a
66be9a2b8321c6ec7', difficulty)
    total_time = str((time.time() - start))
    print(f"Ether mining finished.")
    print(f"Ether Mining took : {total_time} seconds")
    print(f"Calculated Hash = {new_hash}")
```

OUTPUT:



```
Ether mining started.
Successfully mined Ethers with nonce value : 35167
Ether mining finished.
Ether Mining took : 0.06797456741333008 seconds
Calculated Hash =
 0000fc864cda3e3c5ff5be050d0d74068e1367c510e162bb27af608d0d1edbea
```

PRACTICAL 7

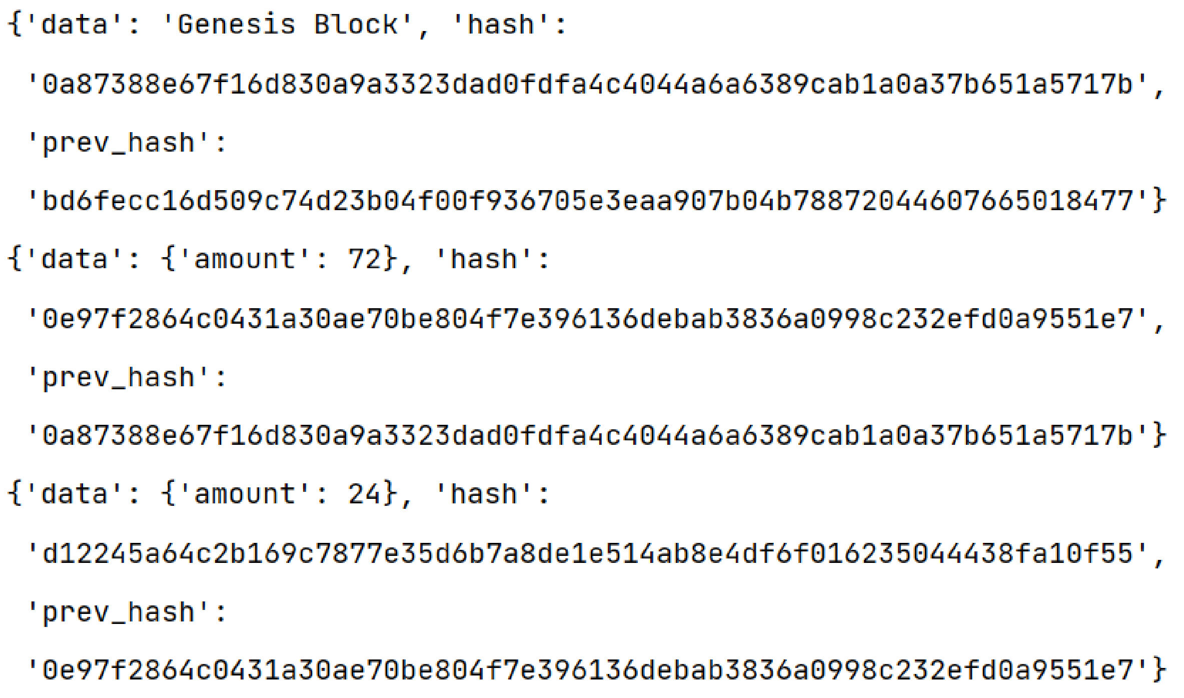
AIM: Demonstrate the running of the blockchain node.

CODE:

```
from hashlib import sha256

def hashGenerator(text):
    return sha256(text.encode("ascii")).hexdigest()

class Block:
    def __init__(self, data, hash, prev_hash):
        self.data = data
        self.hash = hash
        self.prev_hash = prev_hash

class Blockchain:
    def __init__(self):
        hashLast = hashGenerator('gen_last')
        hashStart = hashGenerator('gen_hash')

        genesis = Block('Genesis Block', hashStart, hashLast)
        self.chain = [genesis]

    def add_block(self, new_block):
        new_block.prev_hash = self.chain[-1].hash
        new_block.hash = hashGenerator(str(new_block.data) +
new_block.prev_hash)
        self.chain.append(new_block)

my_coin = Blockchain()
my_coin.add_block(Block({"amount": 72}, "", ""))
my_coin.add_block(Block({"amount": 24}, "", ""))
```

```
for block in my_coin.chain:
    print(block.__dict__)
```

OUTPUT:



```
{'data': 'Genesis Block', 'hash':
 '0a87388e67f16d830a9a3323dad0fdfa4c4044a6a6389cab1a0a37b651a5717b',
 'prev_hash':
 'bd6fecc16d509c74d23b04f00f936705e3eaa907b04b78872044607665018477'}
{'data': {'amount': 72}, 'hash':
 '0e97f2864c0431a30ae70be804f7e396136debab3836a0998c232efd0a9551e7',
 'prev_hash':
 '0a87388e67f16d830a9a3323dad0fdfa4c4044a6a6389cab1a0a37b651a5717b'}
{'data': {'amount': 24}, 'hash':
 'd12245a64c2b169c7877e35d6b7a8de1e514ab8e4df6f016235044438fa10f55',
 'prev_hash':
 '0e97f2864c0431a30ae70be804f7e396136debab3836a0998c232efd0a9551e7'}
```

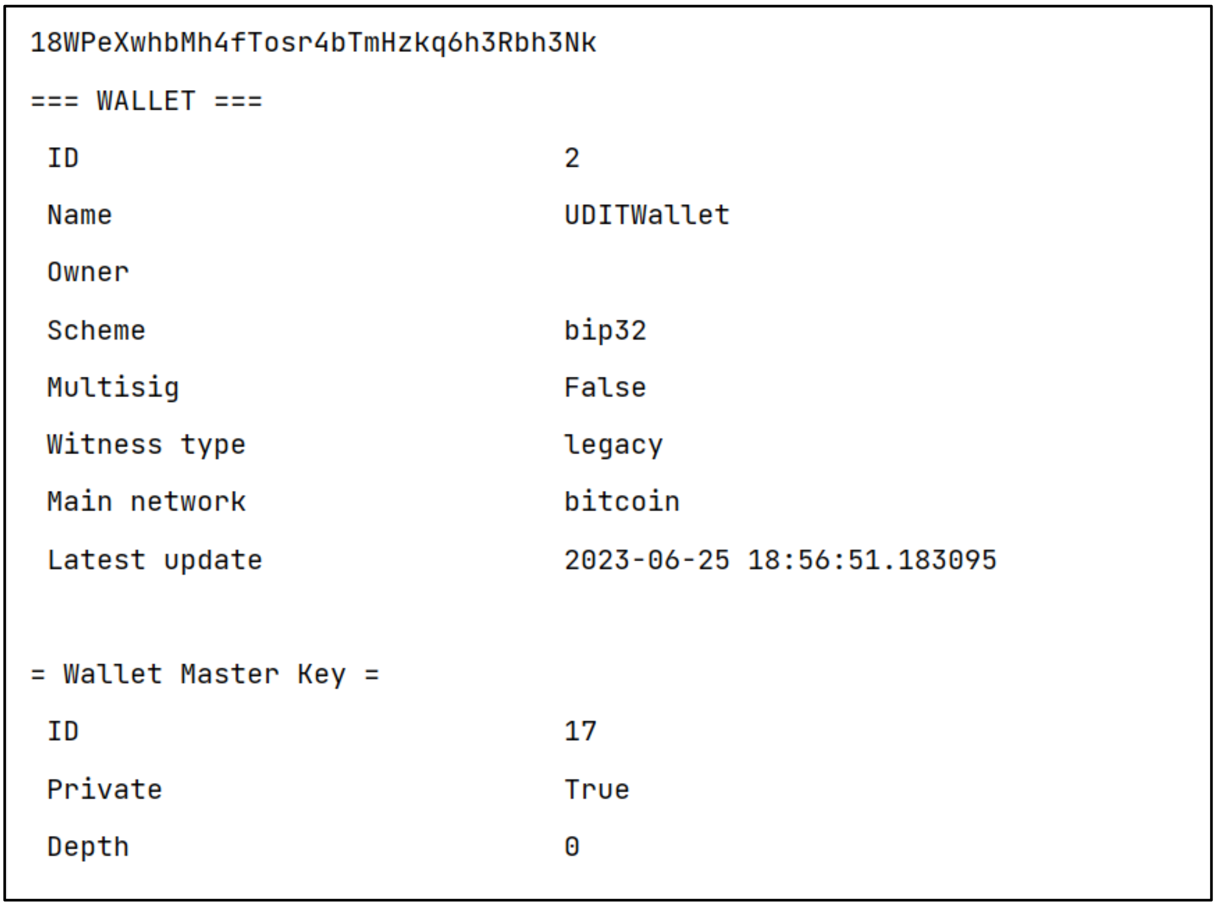
PRACTICAL 8

AIM: Demonstrate the use of Bitcoin Core API.

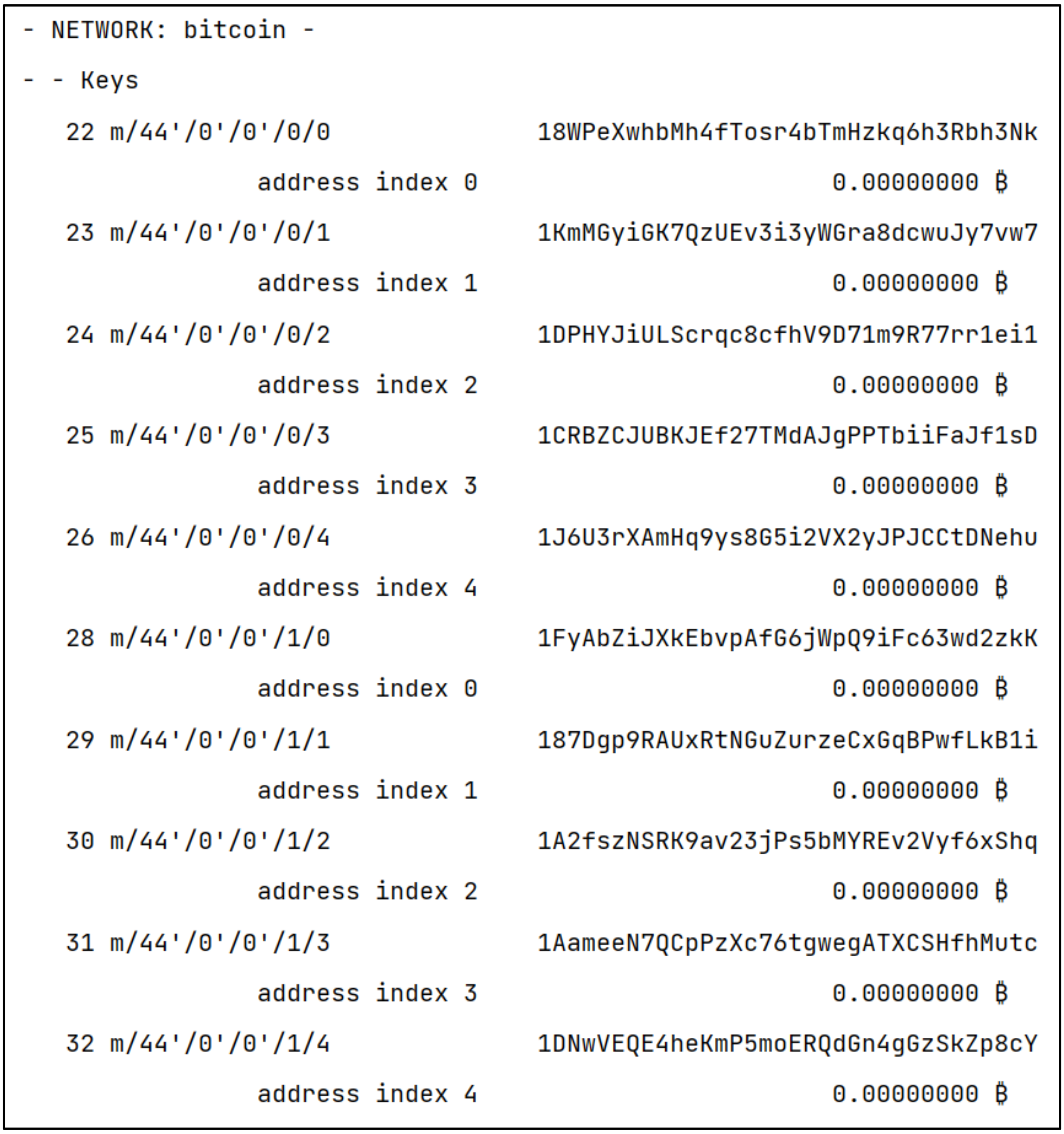
CODE:

```
from bitcoinlib.wallets import Wallet

w = Wallet.create("UDITWallet")

key1 = w.get_key()

print(key1.address)

w.scan()

print(w.info())
```

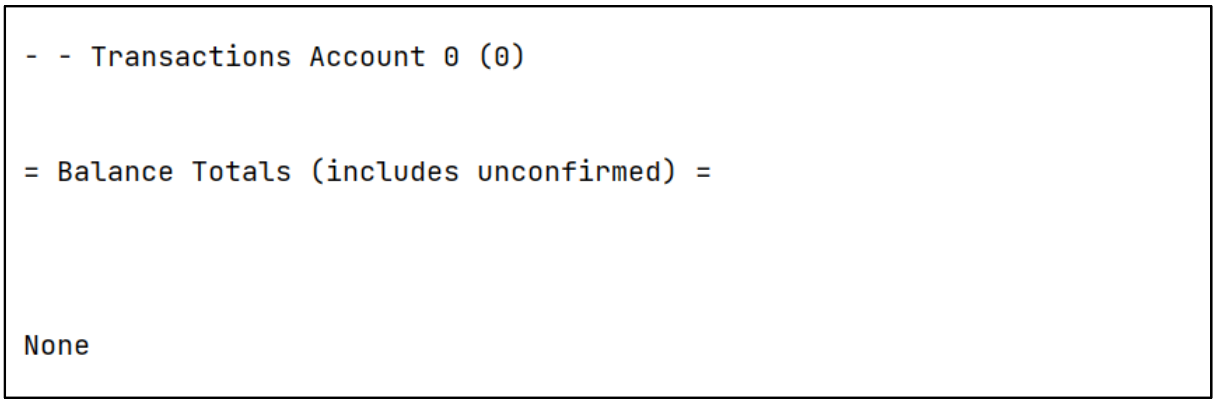
OUTPUT:



```
18WPeXwhbMh4fTosr4bTmHzkq6h3Rbh3Nk
=== WALLET ===
 ID                             2
 Name                           UDITWallet
 Owner
 Scheme                         bip32
 Multisig                       False
 Witness type                   legacy
 Main network                   bitcoin
 Latest update                  2023-06-25 18:56:51.183095

= Wallet Master Key =
 ID                             17
 Private                        True
 Depth                          0
```



```
- NETWORK: bitcoin -
- - Keys
   22 m/44'/0'/0'/0/0            18WPeXwhbMh4fTosr4bTmHzkq6h3Rbh3Nk
                address index 0                        0.00000000 ฿
   23 m/44'/0'/0'/0/1            1KmMGyiGK7QzUEv3i3yWGra8dcwuJy7vw7
                address index 1                        0.00000000 ฿
   24 m/44'/0'/0'/0/2            1DPHYJiULScrqc8cfhV9D71m9R77rr1ei1
                address index 2                        0.00000000 ฿
   25 m/44'/0'/0'/0/3            1CRBZCJUBKJEf27TMdAJgPPTbiiFaJf1sD
                address index 3                        0.00000000 ฿
   26 m/44'/0'/0'/0/4            1J6U3rXAmHq9ys8G5i2VX2yJPJCCtDNehu
                address index 4                        0.00000000 ฿
   28 m/44'/0'/0'/1/0            1FyAbZiJXkEbvpAfG6jWpQ9iFc63wd2zkK
                address index 0                        0.00000000 ฿
   29 m/44'/0'/0'/1/1            187Dgp9RAUxRtNGuZurzeCxGqBPwfLkB1i
                address index 1                        0.00000000 ฿
   30 m/44'/0'/0'/1/2            1A2fszNSRK9av23jPs5bMYREv2Vyf6xShq
                address index 2                        0.00000000 ฿
   31 m/44'/0'/0'/1/3            1AameeN7QCpPzXc76tgwegATXCSHfhMutc
                address index 3                        0.00000000 ฿
   32 m/44'/0'/0'/1/4            1DNwVEQE4heKmP5moERQdGn4gGzSkZp8cY
                address index 4                        0.00000000 ฿
```



```
- - Transactions Account 0 (0)


= Balance Totals (includes unconfirmed) =



None
```

PRACTICAL 9

AIM: Create your own blockchain and demonstrate its use.

CODE:

```
from hashlib import sha256

def hashGenerator(text):
    return sha256(text.encode("ascii")).hexdigest()

class Block:
    def __init__(self,data,hash,prev_hash):
        self.data=data
        self.hash=hash
        self.prev_hash=prev_hash

class Blockchain:
    def __init__(self):
      hashLast=hashGenerator('gen_last')
      hashStart=hashGenerator('gen_hash')

      genesis=Block('gen-data',hashStart,hashLast)
      self.chain=[genesis]

    def add_block(self,data):
        prev_hash=self.chain[-1].hash
        hash=hashGenerator(data+prev_hash)
        block=Block(data,hash,prev_hash)
        self.chain.append(block)

bc=Blockchain()
bc.add_block('1')
bc.add_block('2')
bc.add_block('3')
```

```
for block in bc.chain:
    print(block.__dict__)
```

OUTPUT:

```
{'data': 'gen-data', 'hash':
  '0a87388e67f16d830a9a3323dad0fdfa4c4044a6a6389cab1a0a37b651a5717b',
  'prev_hash':
  'bd6fecc16d509c74d23b04f00f936705e3eaa907b04b78872044607665018477'}
{'data': '1', 'hash':
  'e3e6c97161f3deaf01599fda60ba85593b07f70328bf228473d1d408f7400241',
  'prev_hash':
  '0a87388e67f16d830a9a3323dad0fdfa4c4044a6a6389cab1a0a37b651a5717b'}
{'data': '2', 'hash':
  '47e8645e3c14bd4034a498aa88ea630bc0793375207bf90ca469792a5d9484e1',
  'prev_hash':
  'e3e6c97161f3deaf01599fda60ba85593b07f70328bf228473d1d408f7400241'}
{'data': '3', 'hash':
  '82084603decb1a14a8819dacaa86197659f1e150c4a50186e68043004b5a3c06',
  'prev_hash':
  '47e8645e3c14bd4034a498aa88ea630bc0793375207bf90ca469792a5d9484e1'}
```