University of Mumbai

PAPER - I

BLOCKCHAIN

ELECTIVE - I

NATURAL LANGUAGE PROCESSING

ELECTIVE - II

DEEP LEARNING

SUBMITTED

BY

KHAN HAJRA MOHAMMED RASHID

SEAT NUMBER: 40150

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR QUALIFYING

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

PART 2 (SEMESTER – IV) EXAMINATION 2022 – 2023.

DEPARTMENT OF INFORMATION TECHNOLOGY

3RD FLOOR, DR. SHANKAR DAYALSHARMA BHAVAN,

IDOL BUILDING, VIDYANAGRI, SANTACRUZ (EAST), MUMBAI – 400098.

University of Mumbai

PRACTICAL JOURNAL – PAPER I

PSIT4P1a

Blockchain

SUBMITTED BY

Khan Hajra Mohammed Rashid

SEAT NO 40150

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 Ms. **Khan Hajra Mohammed Rashid Seat No. 40150** 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

5	23/05/2023	Install hyperledger fabric and composer. Deploy and execute the application.	105	
6	22/06/2023	Write a program to demonstrate mining of Ether.	113	
7	22/06/2023	Demonstrate the running of the blockchain node.	115	
8	22/06/2023	Demonstrate the use of Bitcoin Core API.	117	
9	23/06/2023	Create your own blockchain and demonstrate its use.	120	

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 builtin 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("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")


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:
 30819f300d06092a864886f70d010101050003818d0030818902818100c383a3fc7
 248a85bb871be70bdad857db28da890de9de76f3df85950fa0452ab1615bf6b7967f
 28c642e433a04335befdb4232c0735d271a0949a28e51e9154f327c7edd6be6e5588
 73c12afbd66a48c6fc726515789d1109438f75446829b3832be14e894a40f27e1331
 cc48e536a30cc47a1039b4d87364a3ac2c3f7553110203010001
-----
recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100a597e8496e
 f09e903eac0b9f97d8bfde76565f58599206bbd47fb020b0d0daef0568449ef7cb44
 68abe58a30e7877276404a5264840f4e0ddda570cbefec408c459d58429573427694
 382caa972f3878f7ae04ff27bbea057bf9360dd65e193eaf8301c5168840e6a55812
 867b746de1da1695c5130d49cbee519c6cb23698710203010001
-----
value: 15.0
-----
time: 2023-06-25 18:38:53.536480
-----
```

```
sender:
 30819f300d06092a864886f70d01010105000381d0030818902818100c383a3fc7
 248a85bb871be70bdad857db28da890de9de76f3df85950fa0452ab1615bf6b7967f
 28c642e433a04335befdb4232c0735d271a0949a28e51e9154f327c7edd6be6e5588
 73c12afbd66a48c6fc726515789d1109438f75446829b3832be14e894a40f27e1331
 cc48e536a30cc47a1039b4d87364a3ac2c3f7553110203010001
-----
recipient:
 30819f300d06092a864886f70d01010105000381d0030818902818100cfc0b7e441
 5f62ea994a001e1d30f8d23d89ec17062f162c5f2ac56c4c5883ace946a59854562d
 116efa15269b50a0d180e8b46db923ec942c347826037007c3b638bd666a4949e773
 bc98ba3ff5e0f4fa06e97f0d31986ee2090e02a4ba9dd60951d48e8253763a08730b
 0c4e4f3aae04ac568b9c9708406a5c9564d03ace250203010001
-----
value: 6.0
-----
time: 2023-06-25 18:38:53.538479
-----
```

```
sender:
 30819f300d06092a864886f70d010101050003818d003081902818100a597e8496e
 f09e903eac0b9f97d8bfde76565f58599206bbd47fb020b0d0daef0568449ef7cb44
 68abe58a30e7877276404a5264840f4e0ddda570cbefec408c459d58429573427694
 382caa972f3878f7ae04ff27bbea057bf9360dd65e193eaf8301c5168840e6a55812
 867b746de1da1695c5130d49cbee519c6cb23698710203010001
-----
recipient:
 30819f300d06092a864886f70d010101050003818d003081902818100a51bb5b3ba
 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.

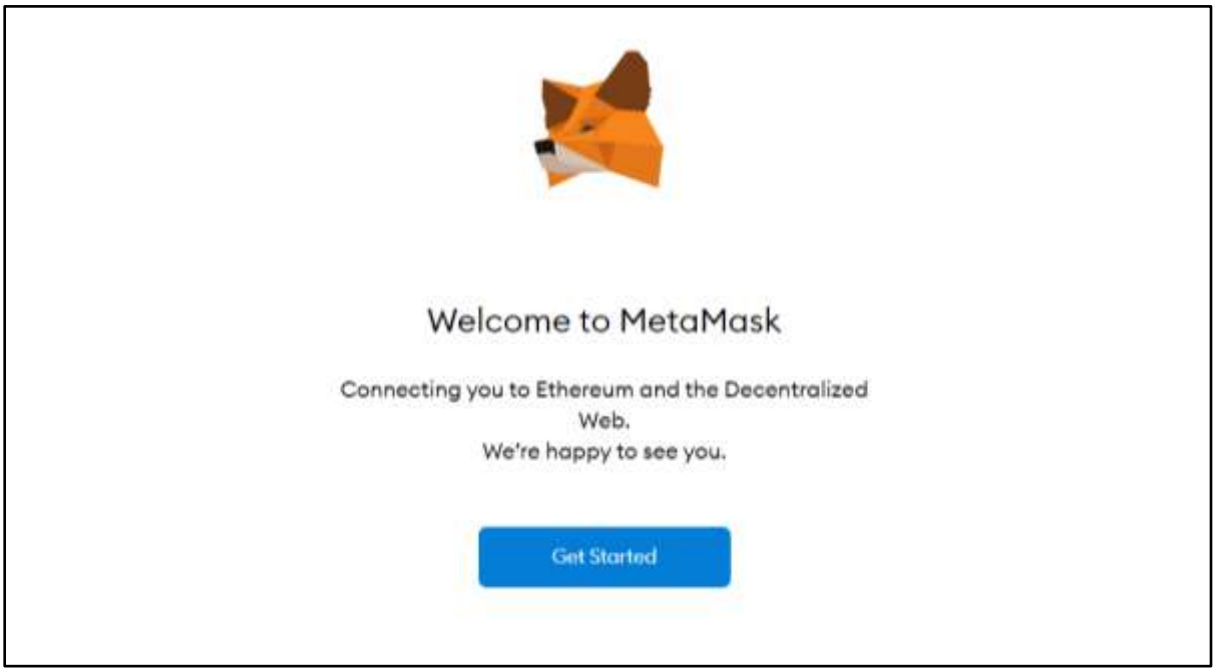
Step 1: Go to Chrome Web Store Extensions Section.

Step 2: Search MetaMask.

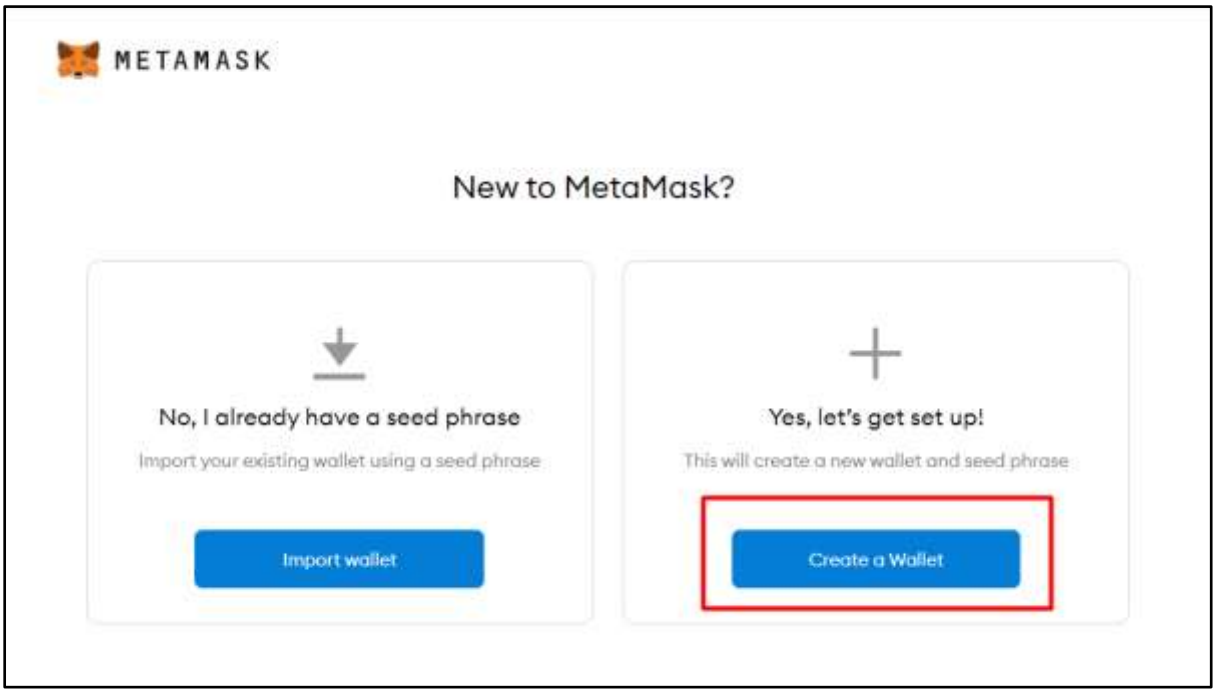
Step 3: Check the number of downloads to make sure that the legitimate MetaMask is being installed, as hackers might try to make clones of it.

Step 4: Click the Add to Chrome button.

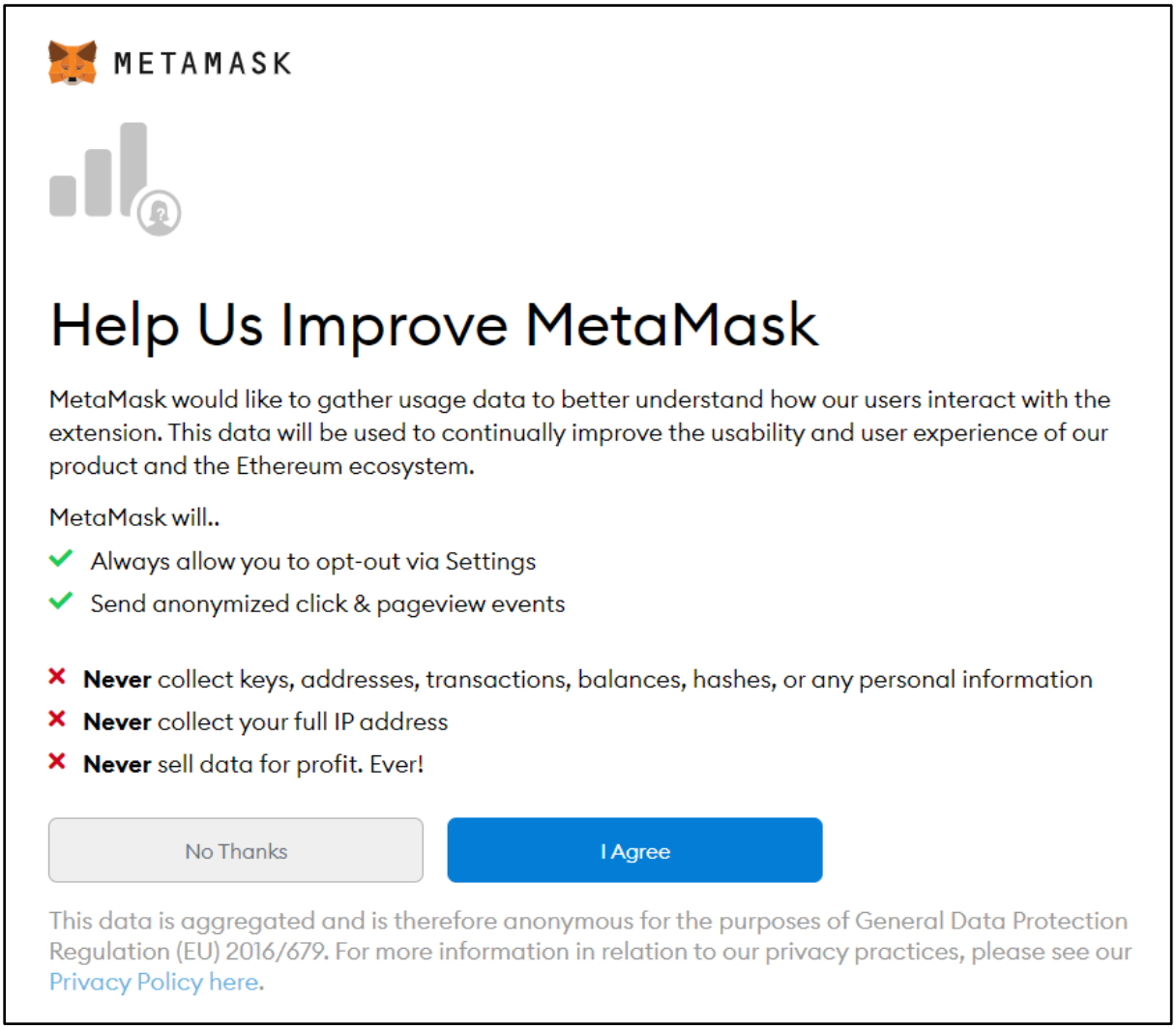
Step 5: Once installation is complete this page will be displayed. Click on the Get Started button.



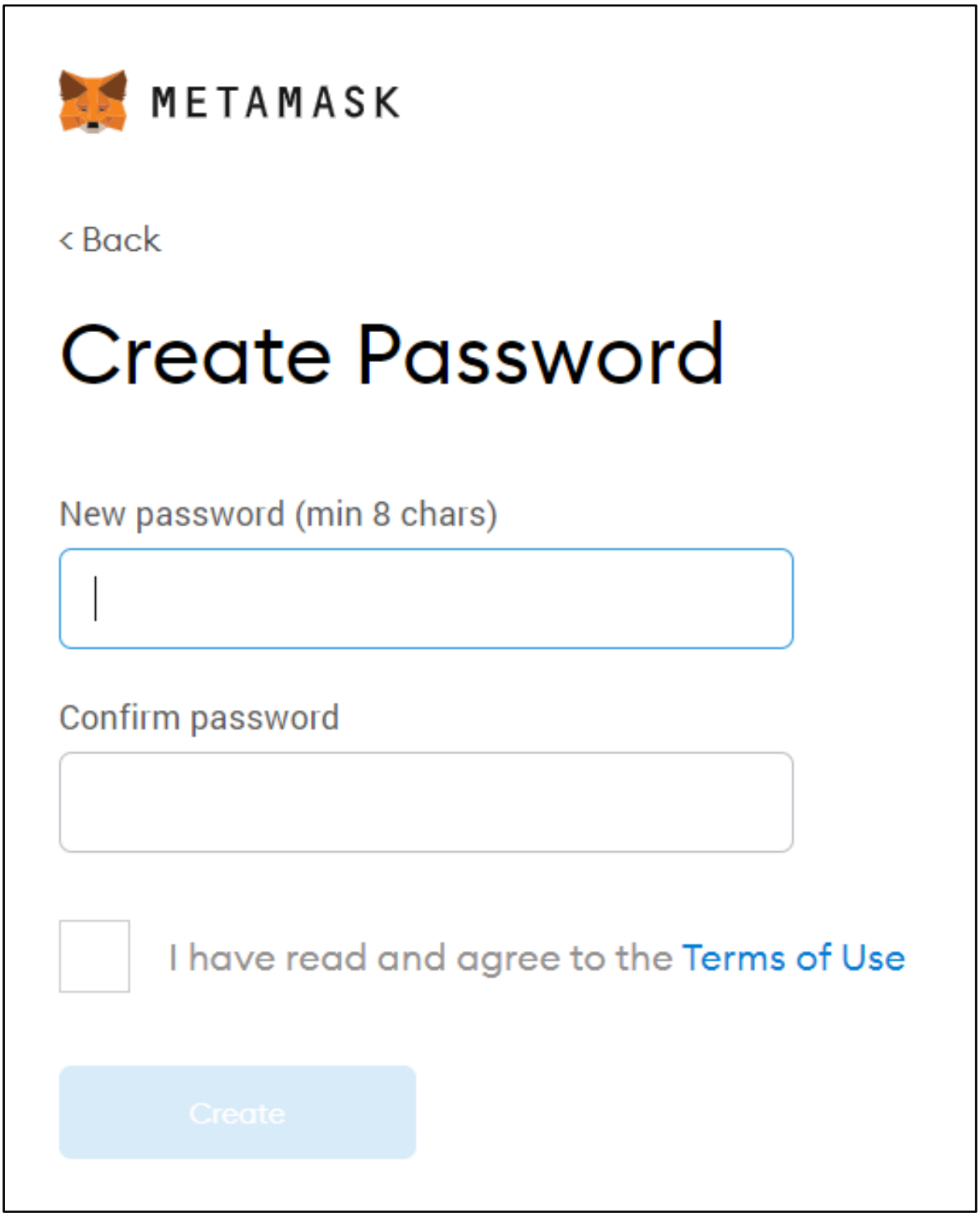
Step 6: This is the first time creating a wallet, so click the Create a Wallet button. If there is already a wallet then import the already created using the Import Wallet button.



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.

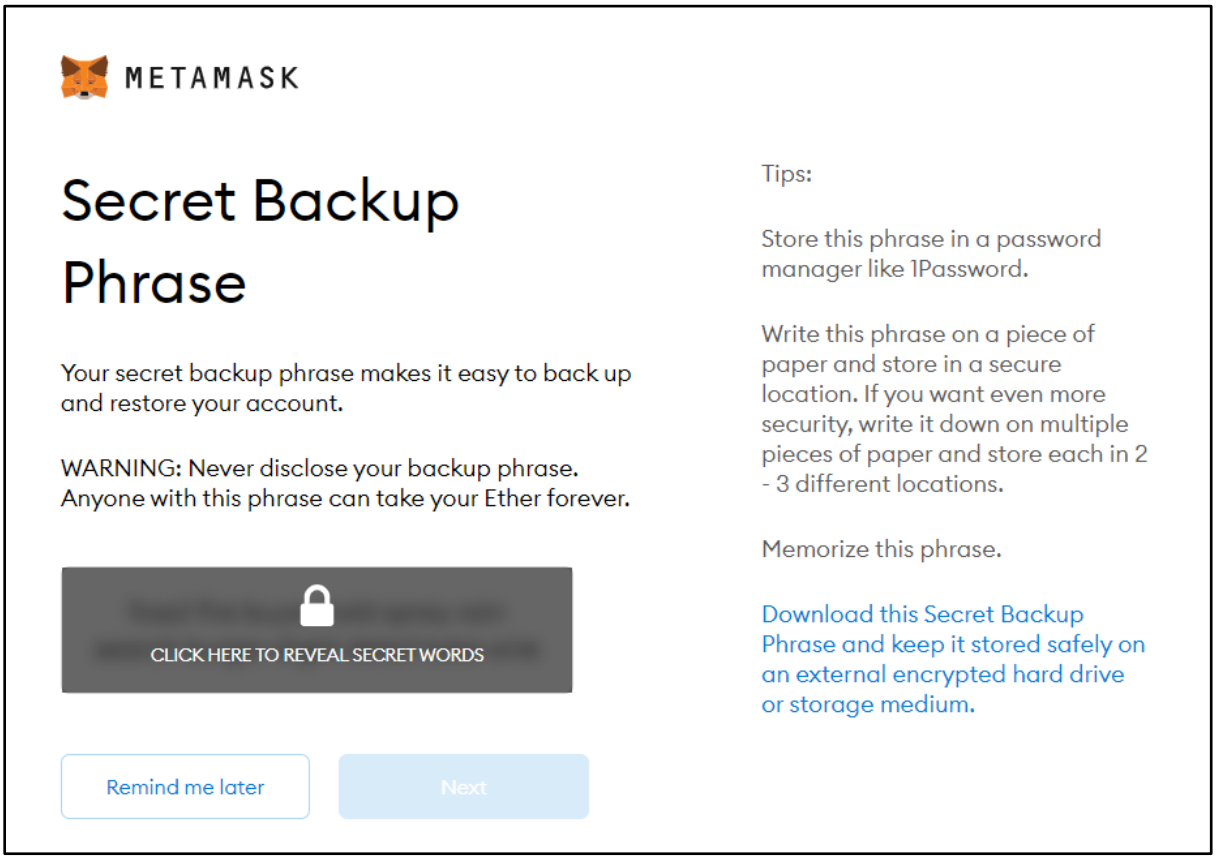


Step 8: Create a password for your wallet. This password is to be entered every time the browser is launched and wants to use MetaMask. A new password needs to be created if chrome is uninstalled or if there is a switching of browsers. In that case, go through the Import Wallet button. This is because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

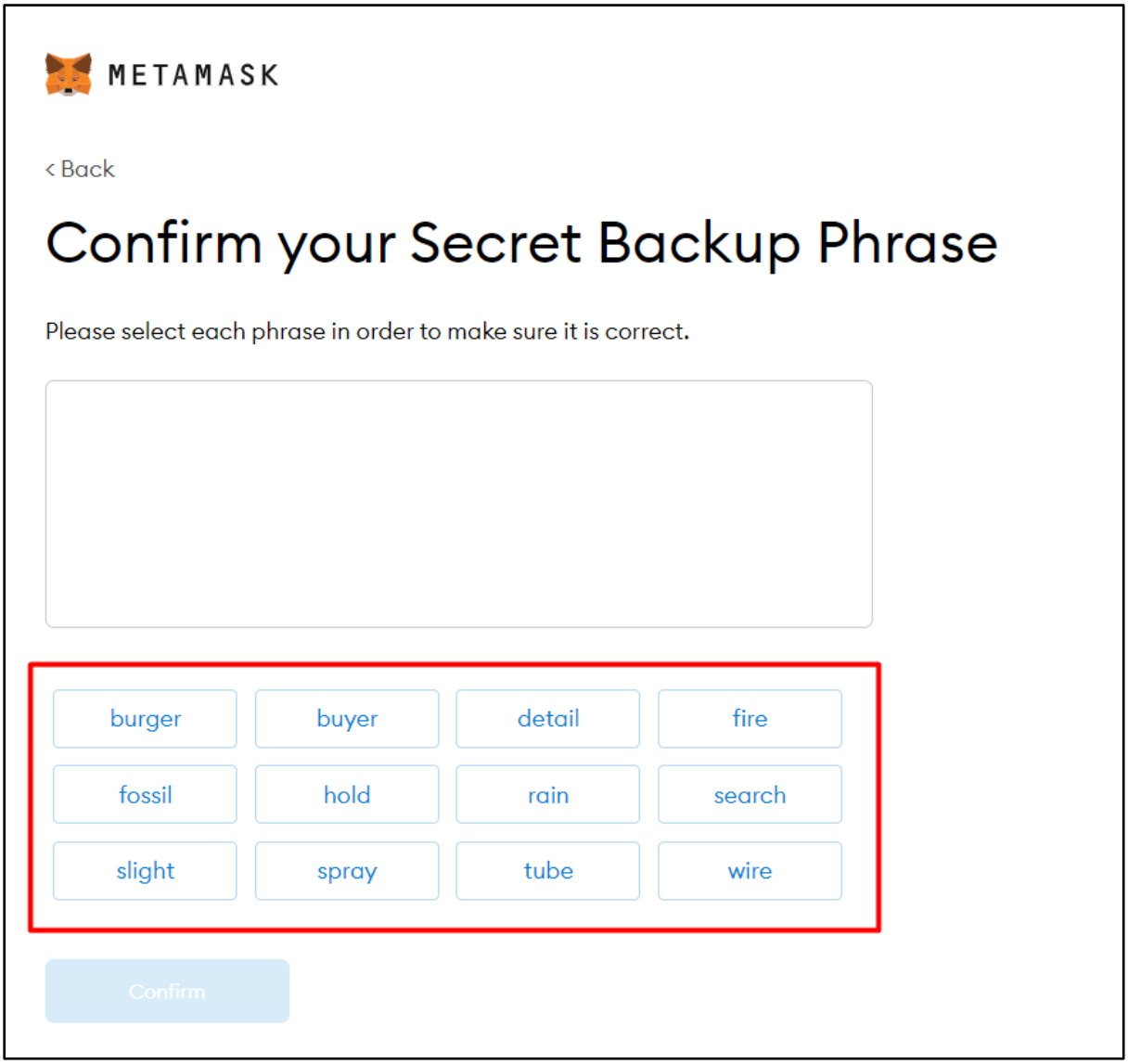


Step 9: Click on the dark area which says Click here to reveal secret words to get your secret phrase.

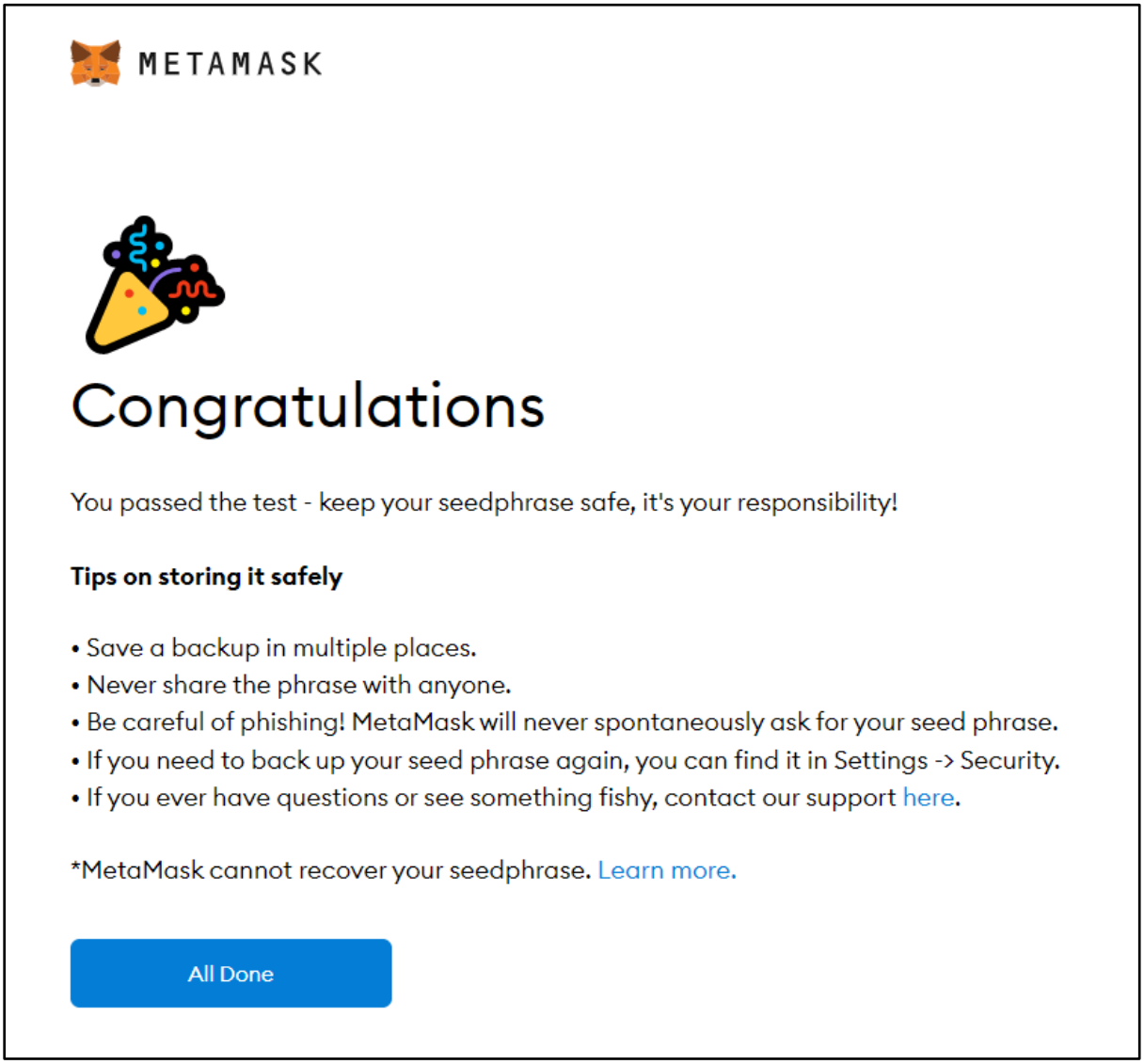
Step 10: This is the most important step. Back up your secret phrase properly. Do not store your secret phrase on your computer. Please read everything on this screen until you understand it completely before proceeding. The secret phrase is the only way to access your wallet if you forget your password.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



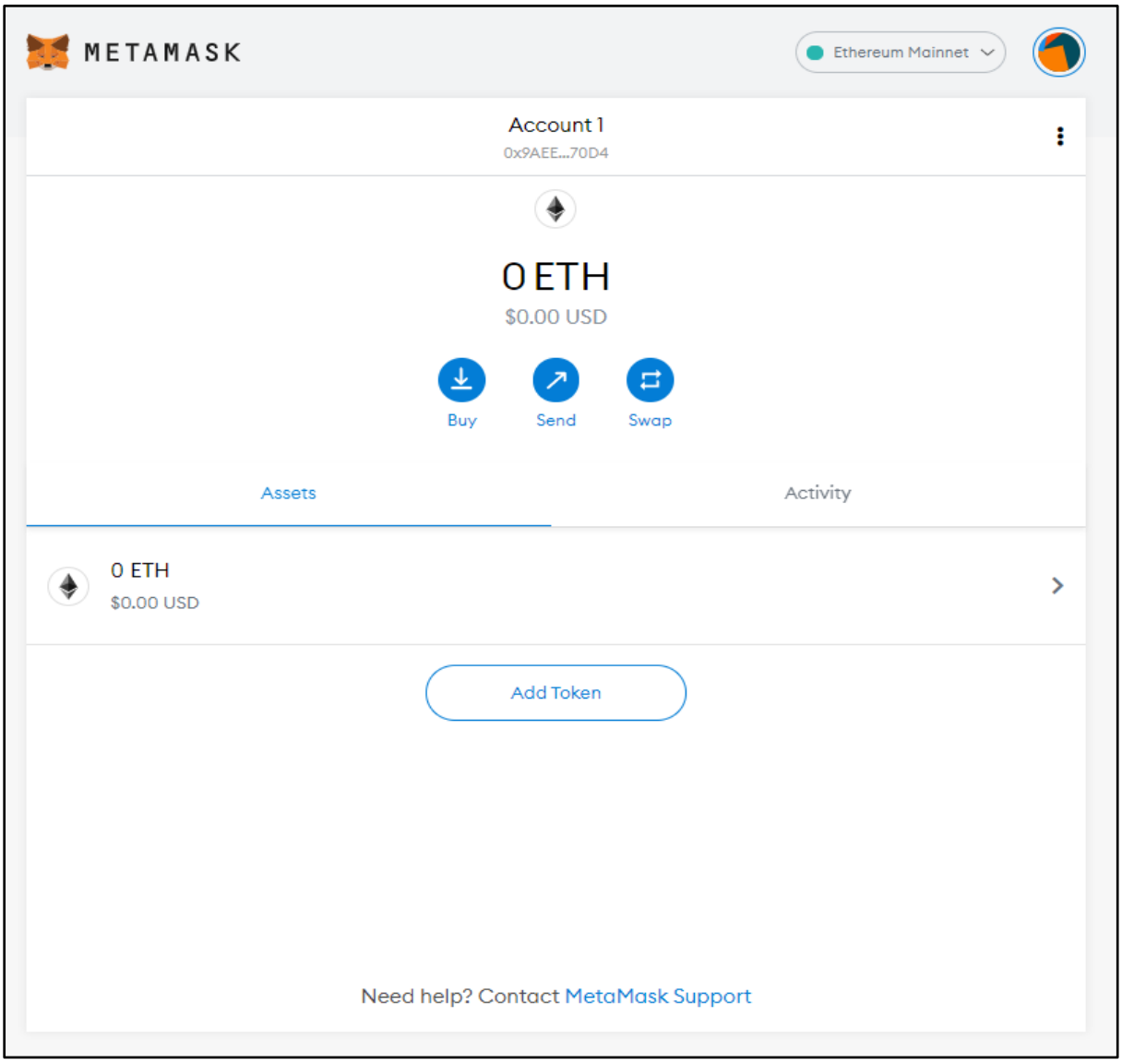
Step 11: Click the buttons respective to the order of the words in your seed phrase. In other words, type the seed phrase using the button on the screen. If done correctly the Confirm button should turn blue.



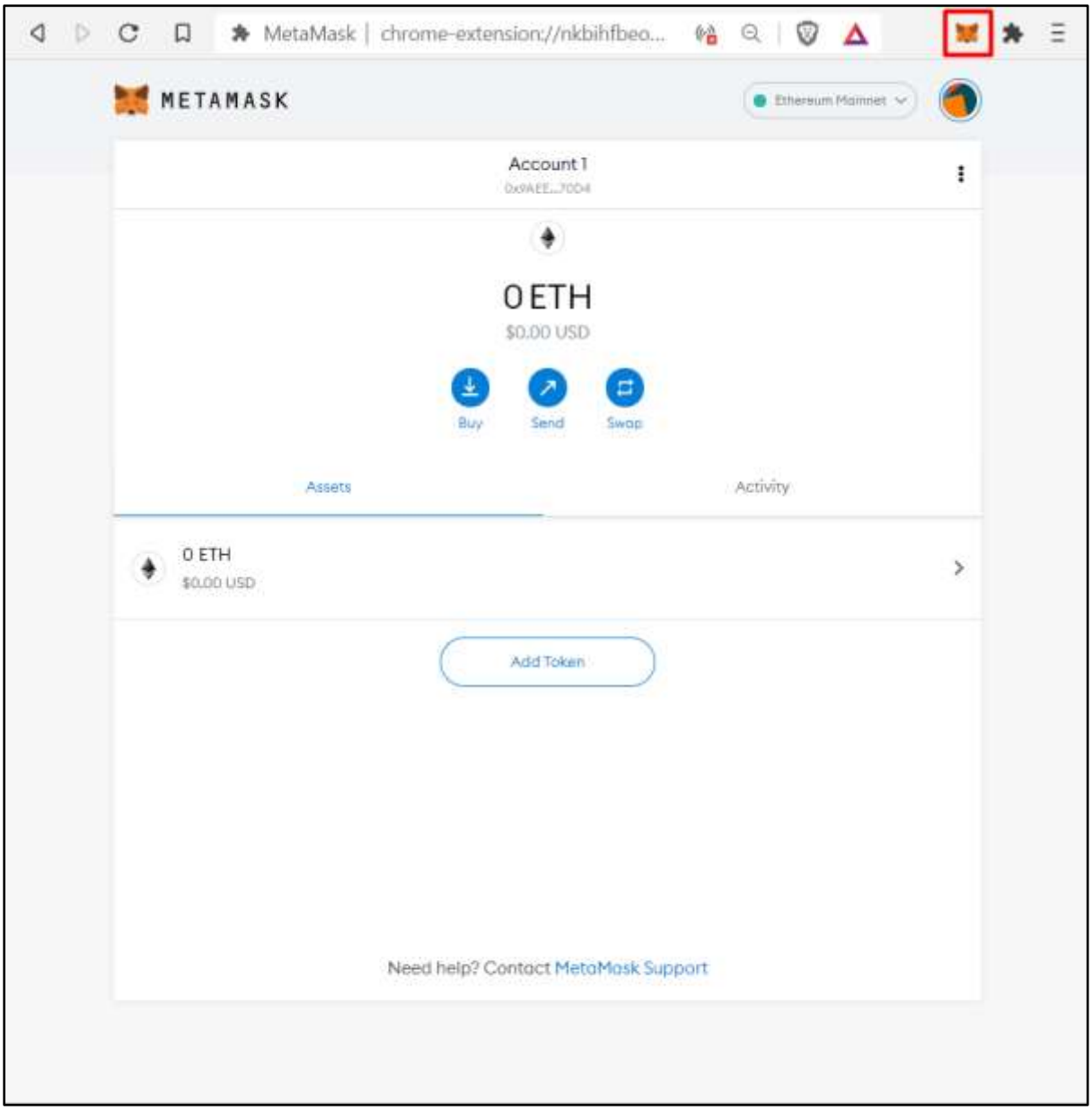
Step 12: Click the Confirm button. Please follow the tips mentioned.



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



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.

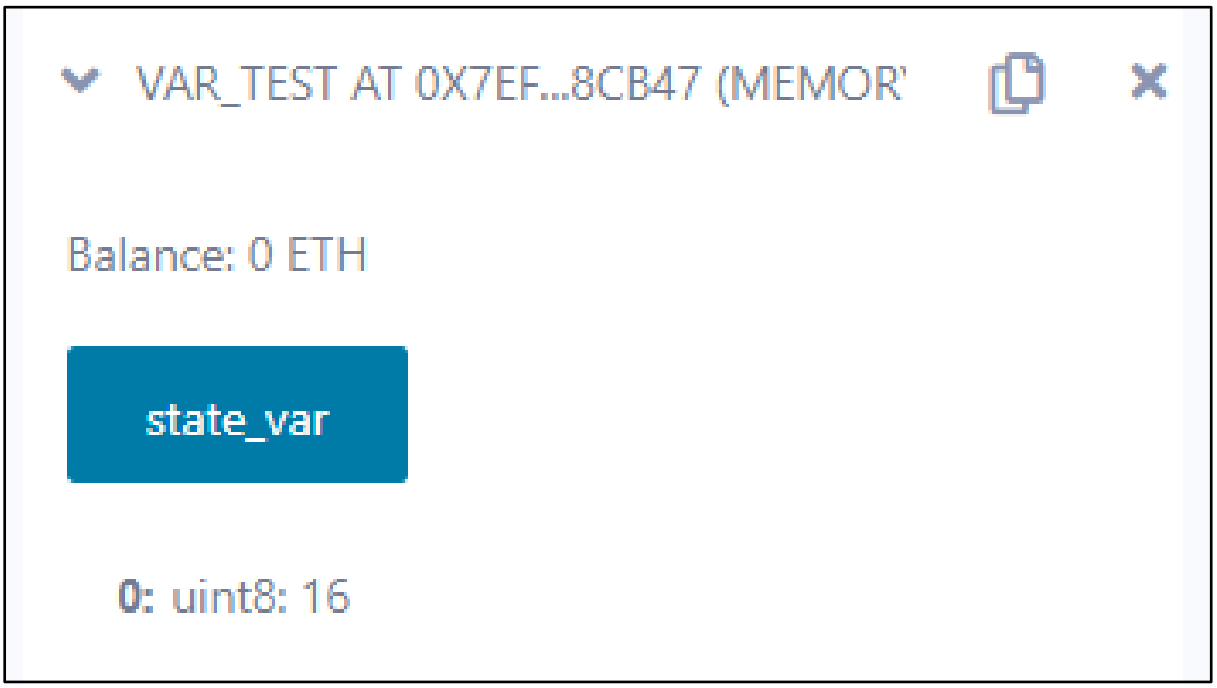
[I] Variable

[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:

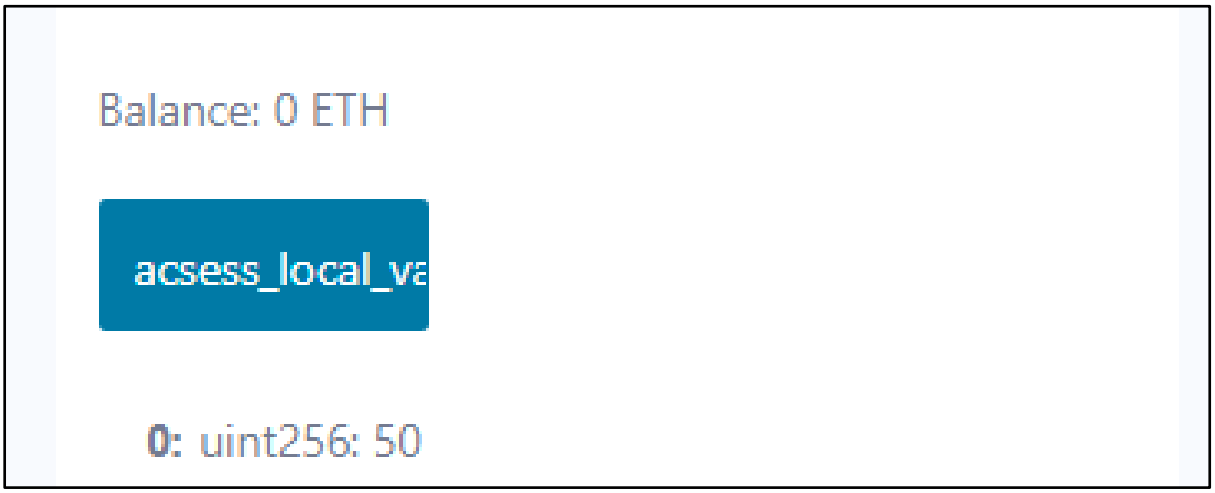

VAR_TEST AT 0X7EF...8CB47 (MEMOR'
Balance: 0 ETH
state_var
0: uint8: 16

[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:

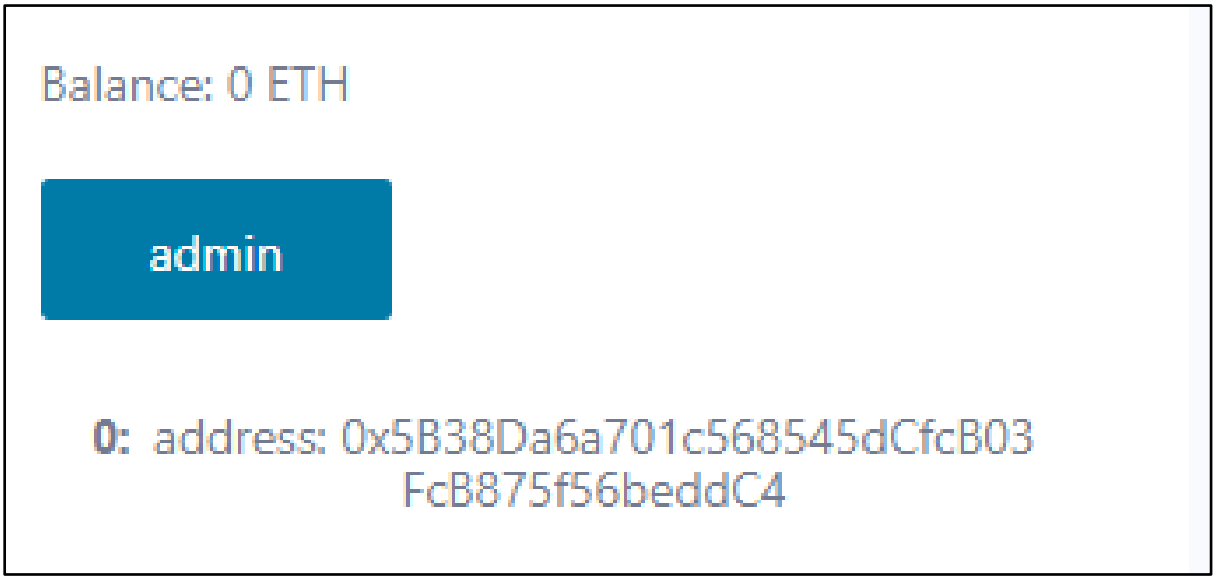


[iii] Global Variable

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:

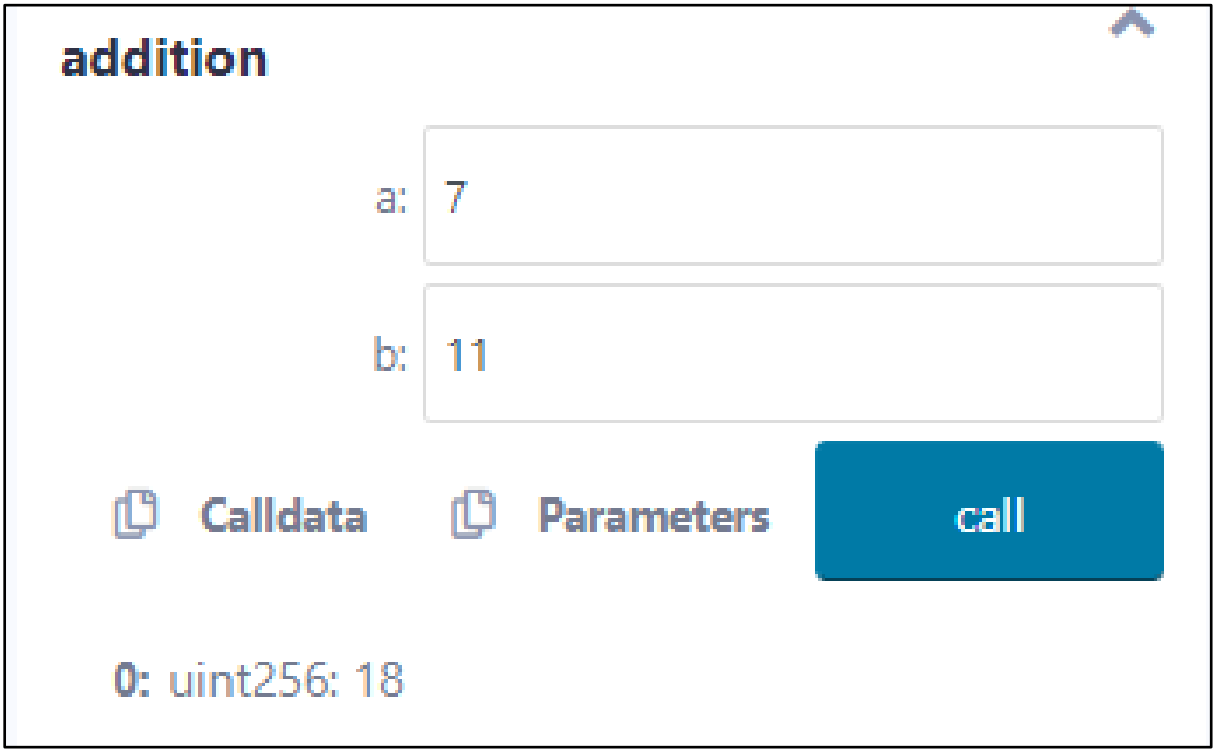


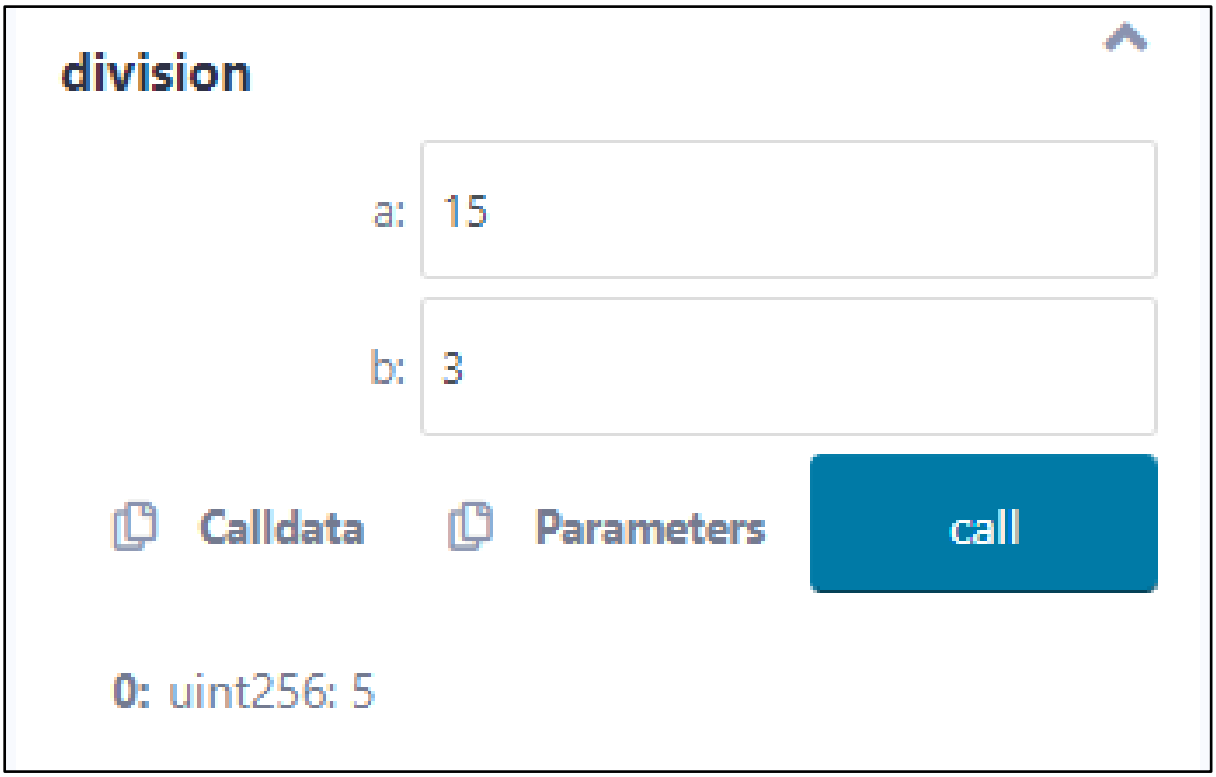
[II] Operators

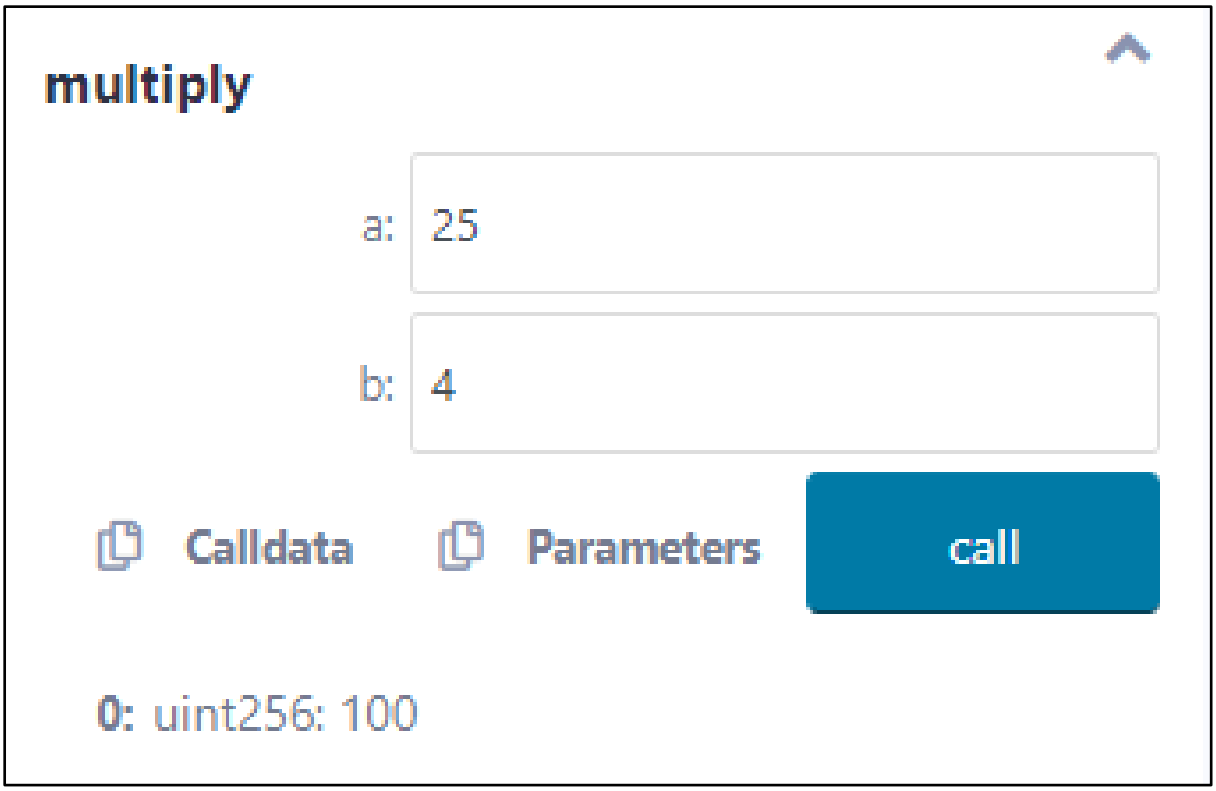
CODE:

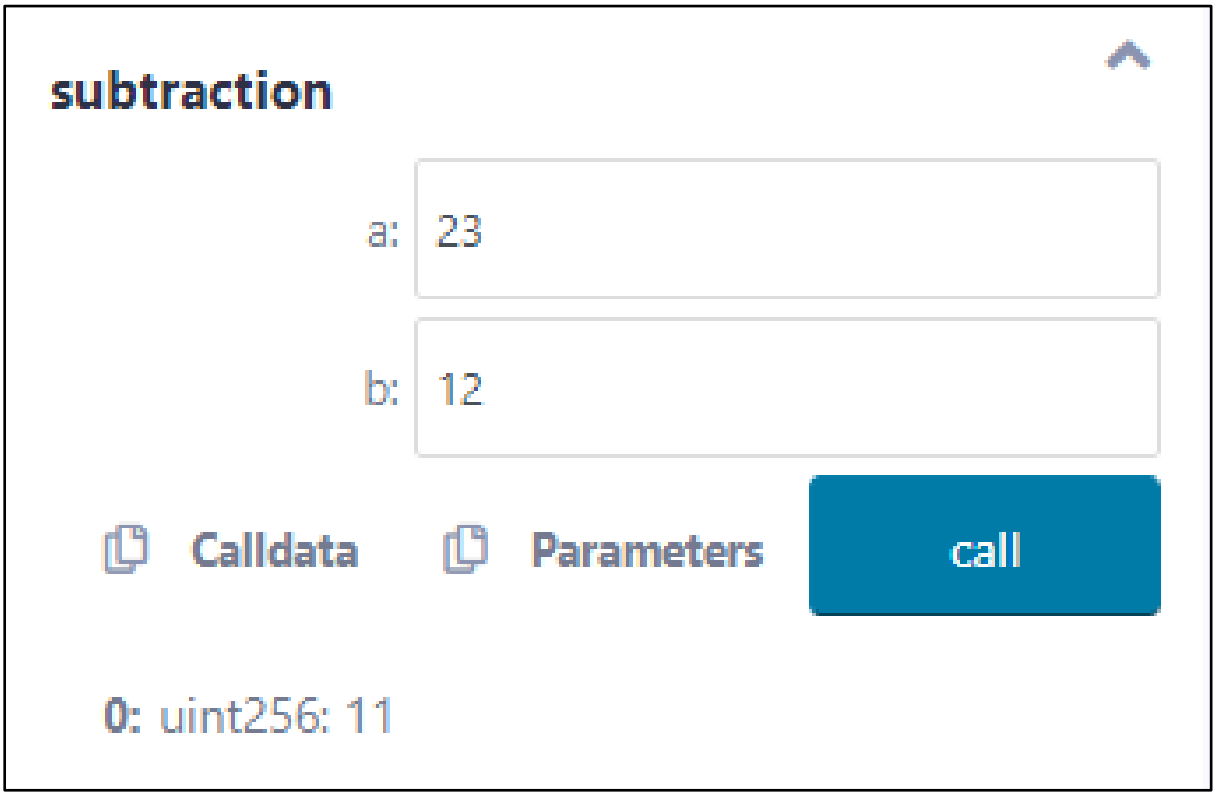
```
pragma solidity ^0.4.0;

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
Calldata
Parameters
call
0: uint256: 18


division
a: 15
b: 3
Calldata
Parameters
call
0: uint256: 5


multiply
a: 25
b: 4
Calldata
Parameters
call
0: uint256: 100


subtraction
a: 23
b: 12
Calldata
Parameters
call
0: uint256: 11

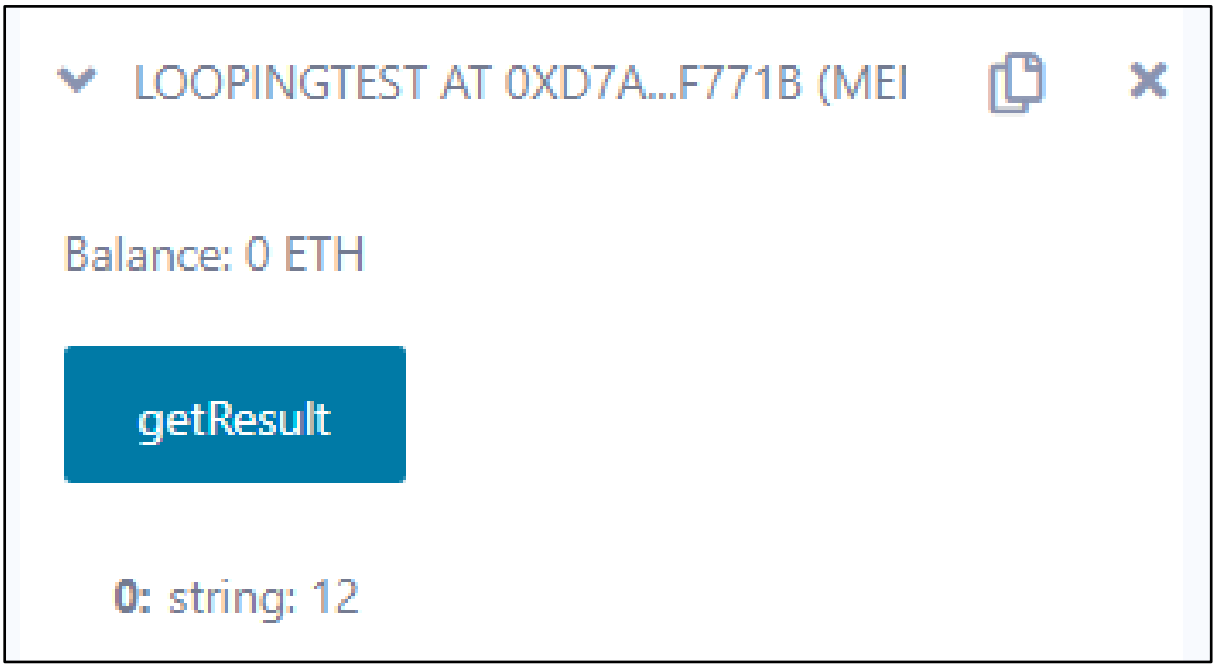
[III] Loops

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:

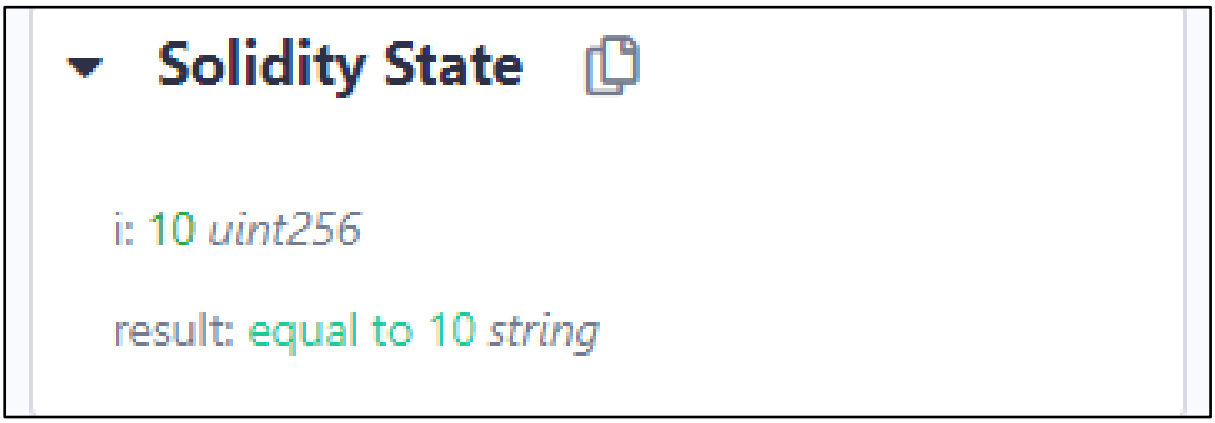

LOOPINGTEST AT 0XD7A...F771B (MEI
Balance: 0 ETH
getResult
0: string: 12

[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:



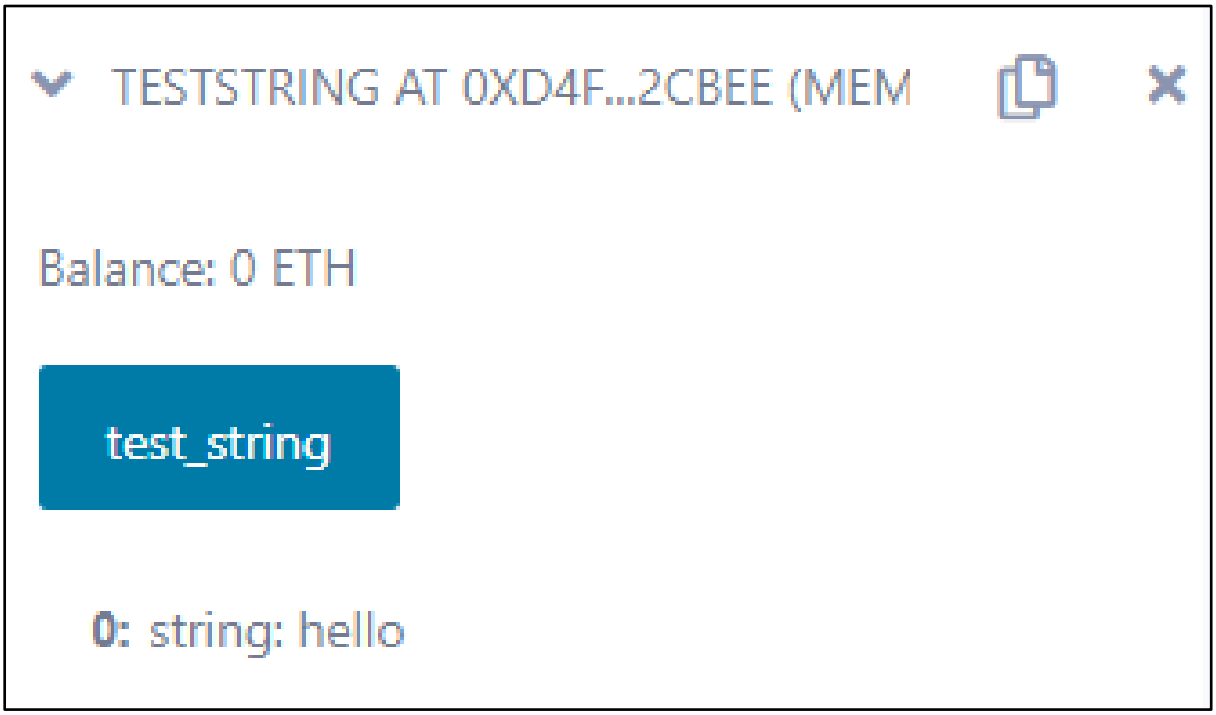
[V] Strings

[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:

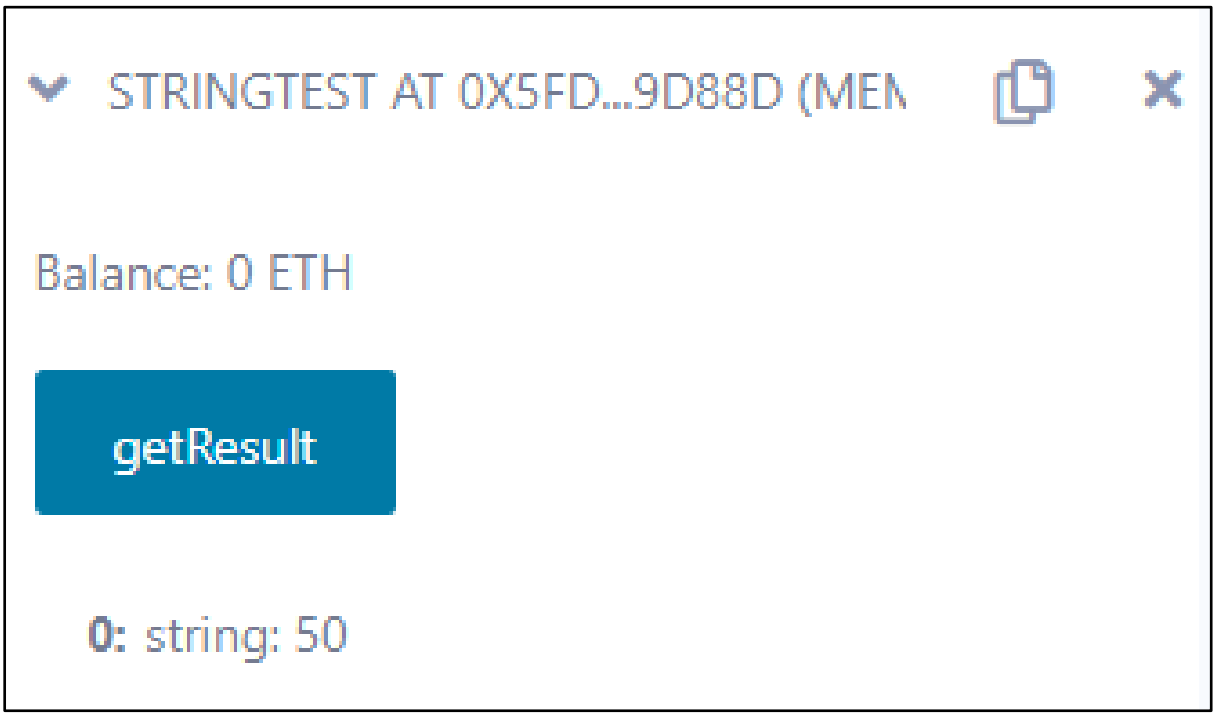


[ii] Single Quotes

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:

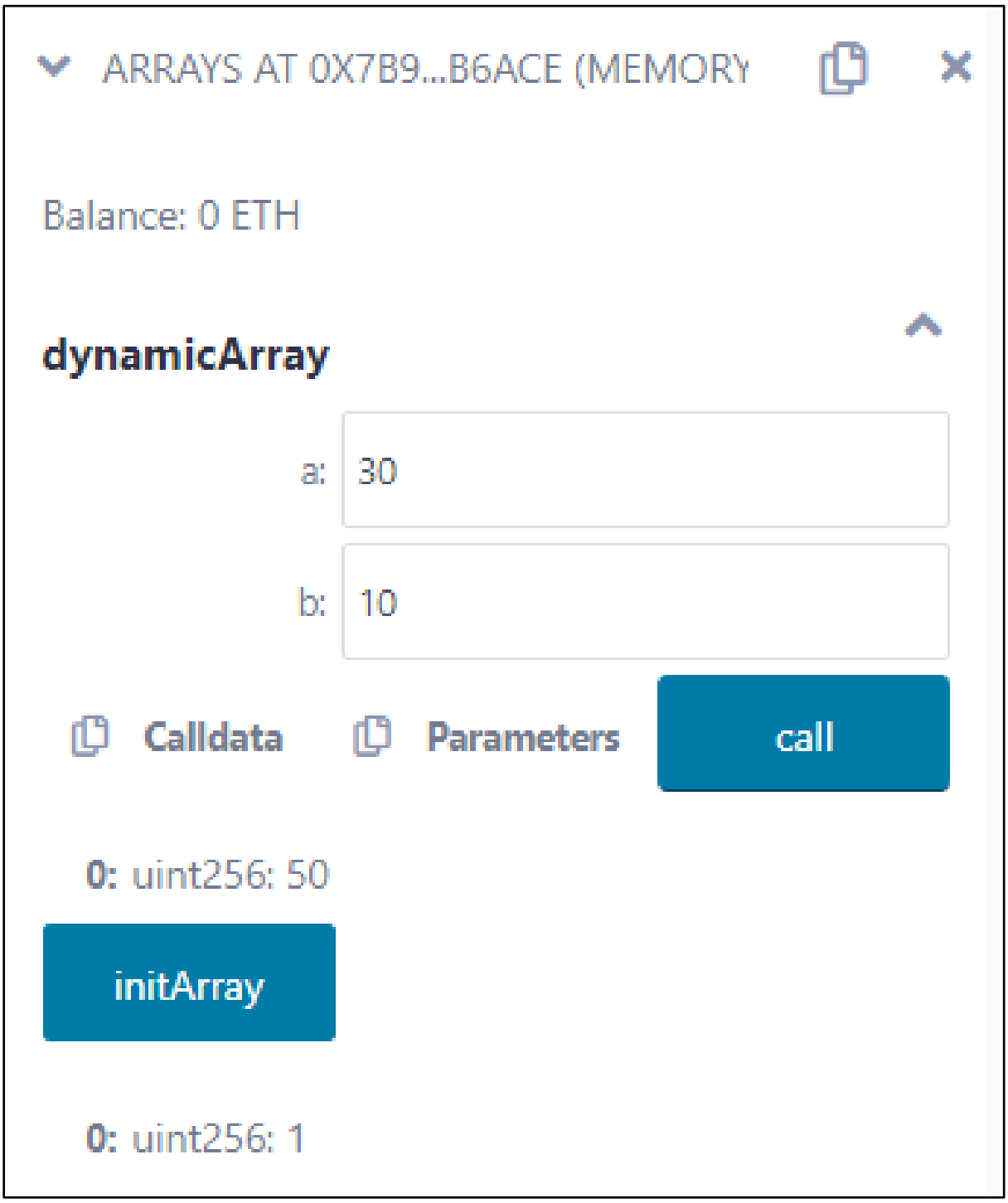

STRINGTEST AT 0X5FD...9D88D (MEN
Balance: 0 ETH
getResult
0: string: 50

[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:

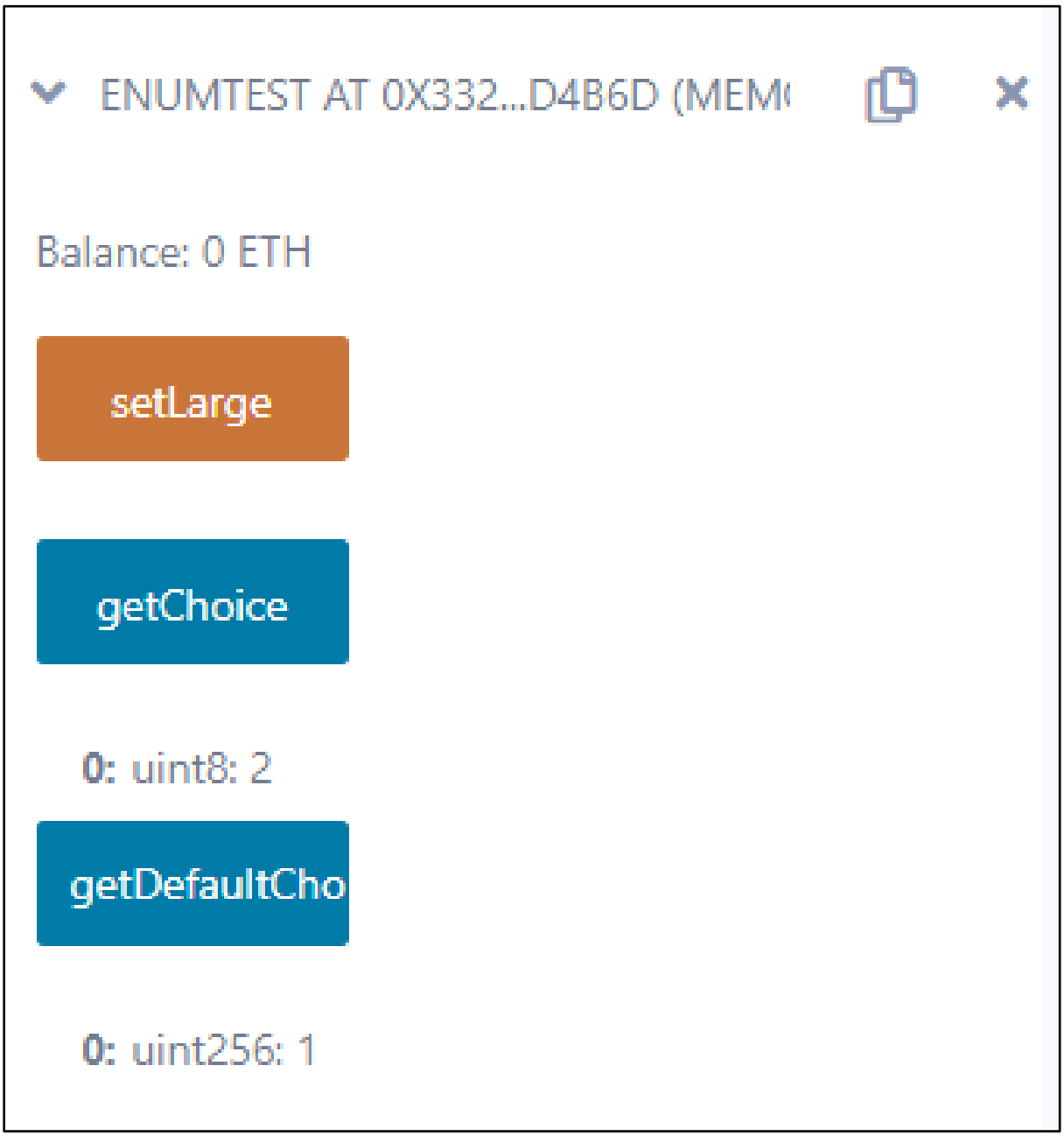

ARRAYS AT 0X7B9...B6ACE (MEMORY
Balance: 0 ETH
dynamicArray
a: 30
b: 10
Calldata
Parameters
call
0: uint256: 50
initArray
0: uint256: 1

[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:

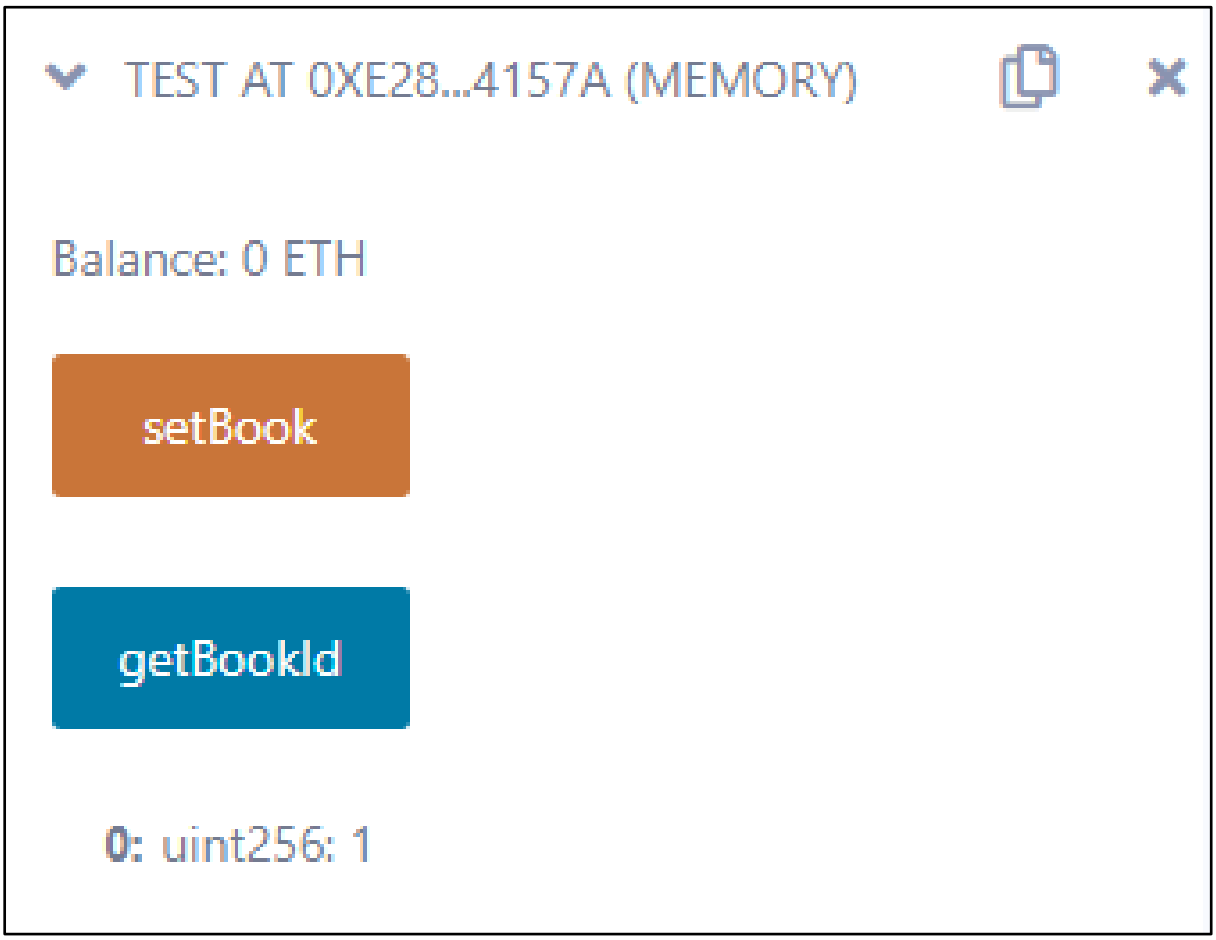

ENUMTEST AT 0X332...D4B6D (MEM
Balance: 0 ETH
setLarge
getChoice
0: uint8: 2
getDefaultCho
0: uint256: 1

[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:


TEST AT 0XE28...4157A (MEMORY)
Balance: 0 ETH
setBook
getBookId
0: uint256: 1

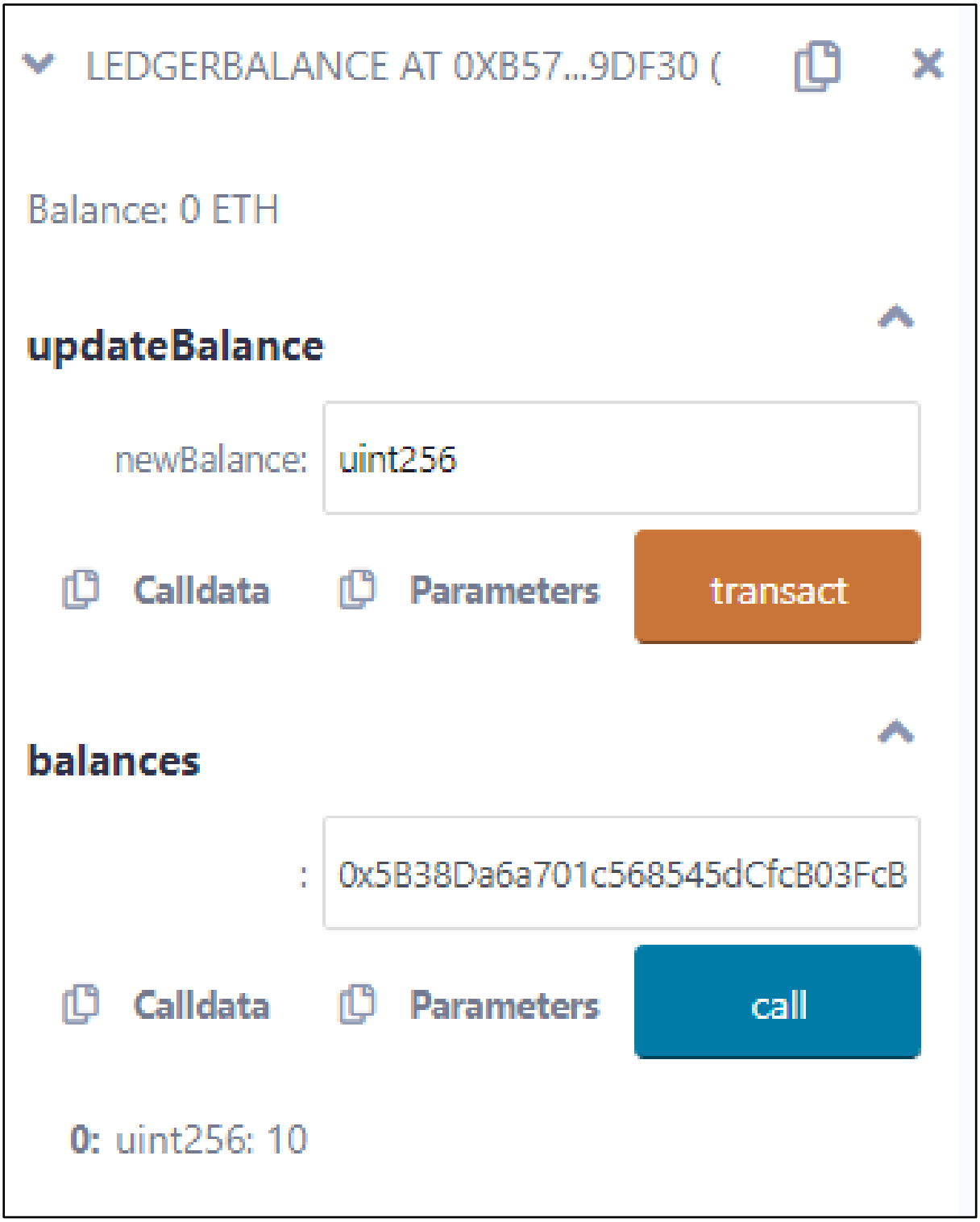
[IX] Mappings

CODE:

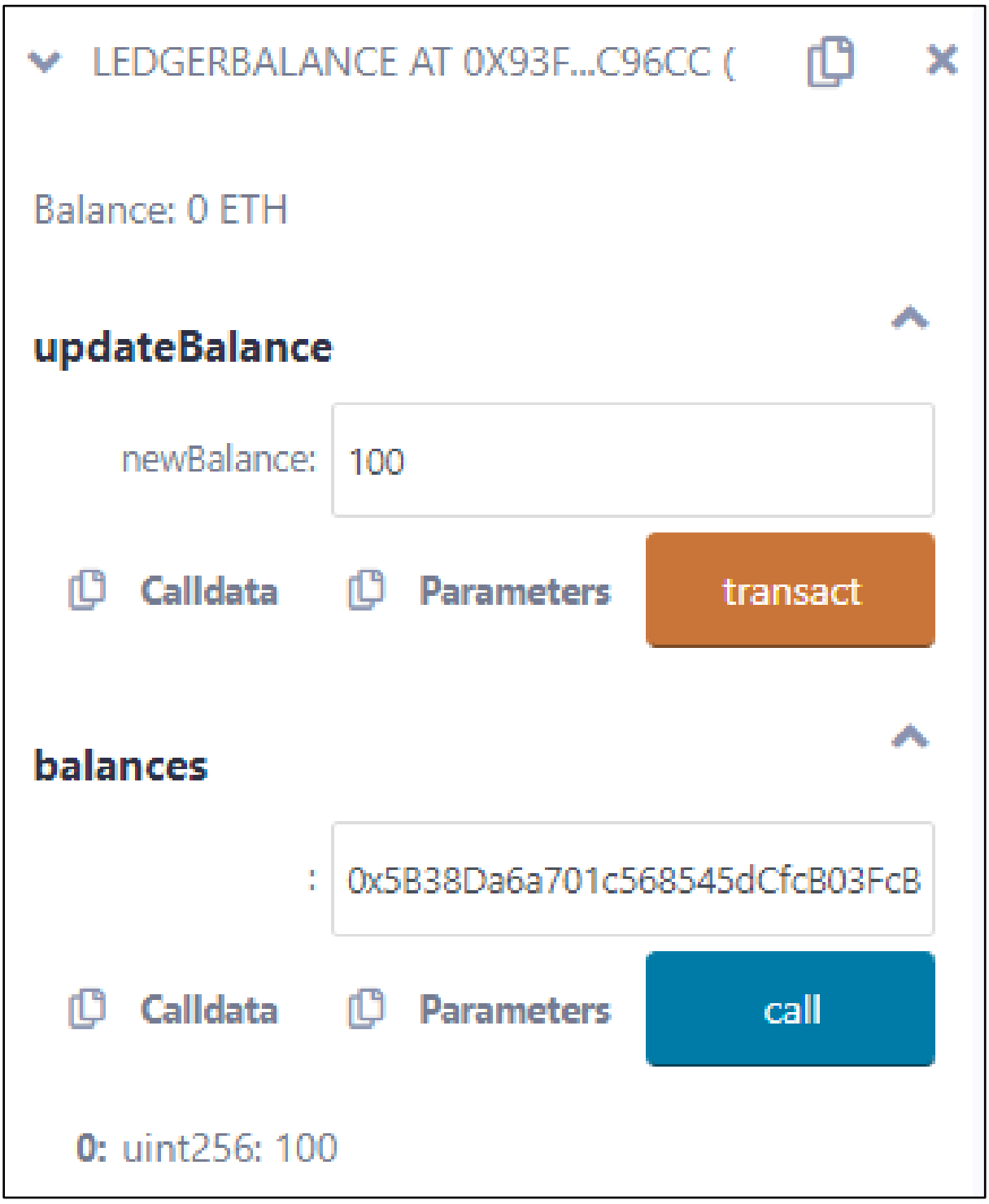
```
pragma solidity ^0.5.0;

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



With Updated Balance: 100

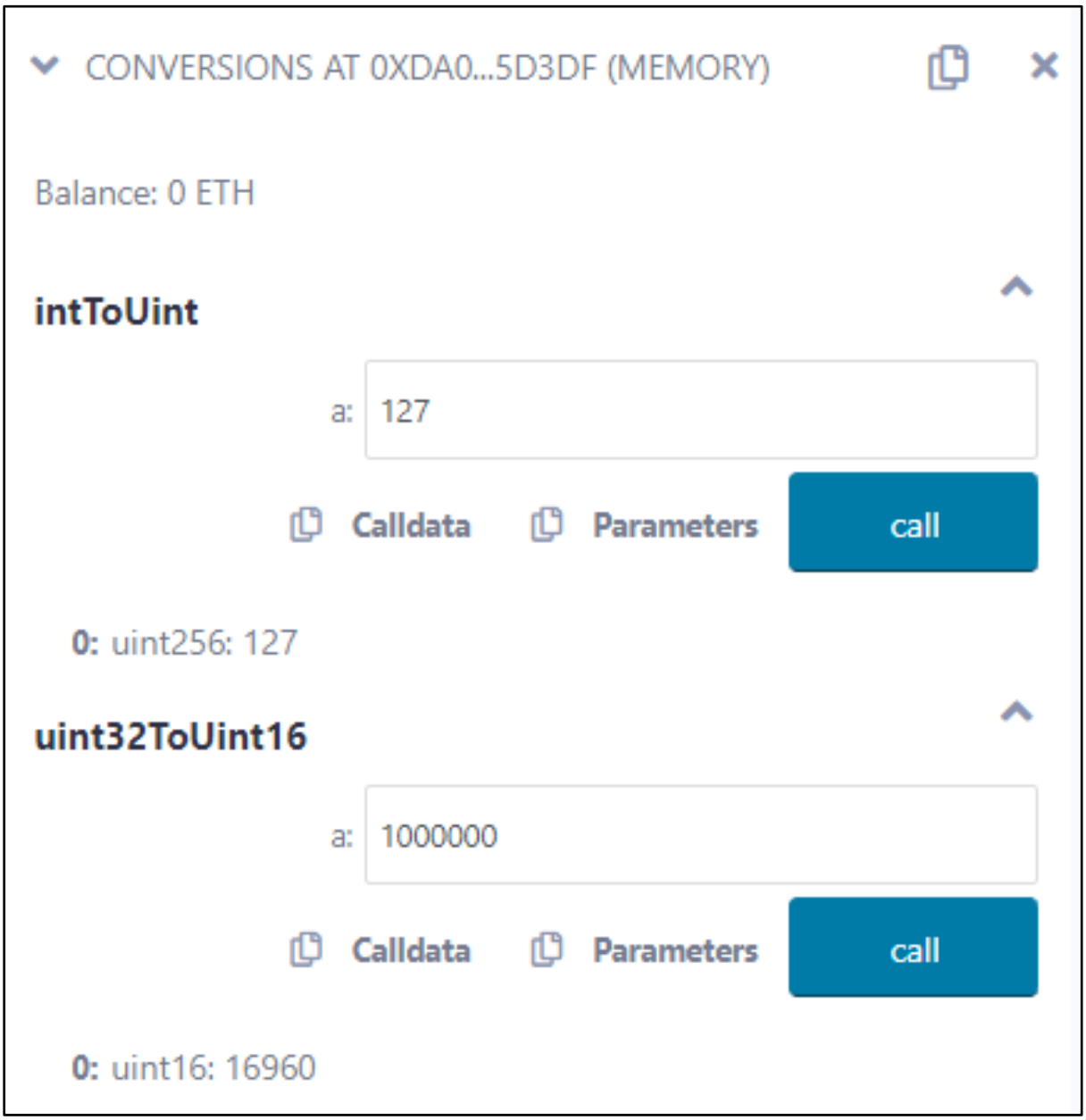


[X] Conversions

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:


CONVERSIONS AT 0XDA0...5D3DF (MEMORY)
Balance: 0 ETH
intToUint
a: 127
Calldata
Parameters
call
0: uint256: 127
uint32ToUint16
a: 1000000
Calldata
Parameters
call
0: uint16: 16960

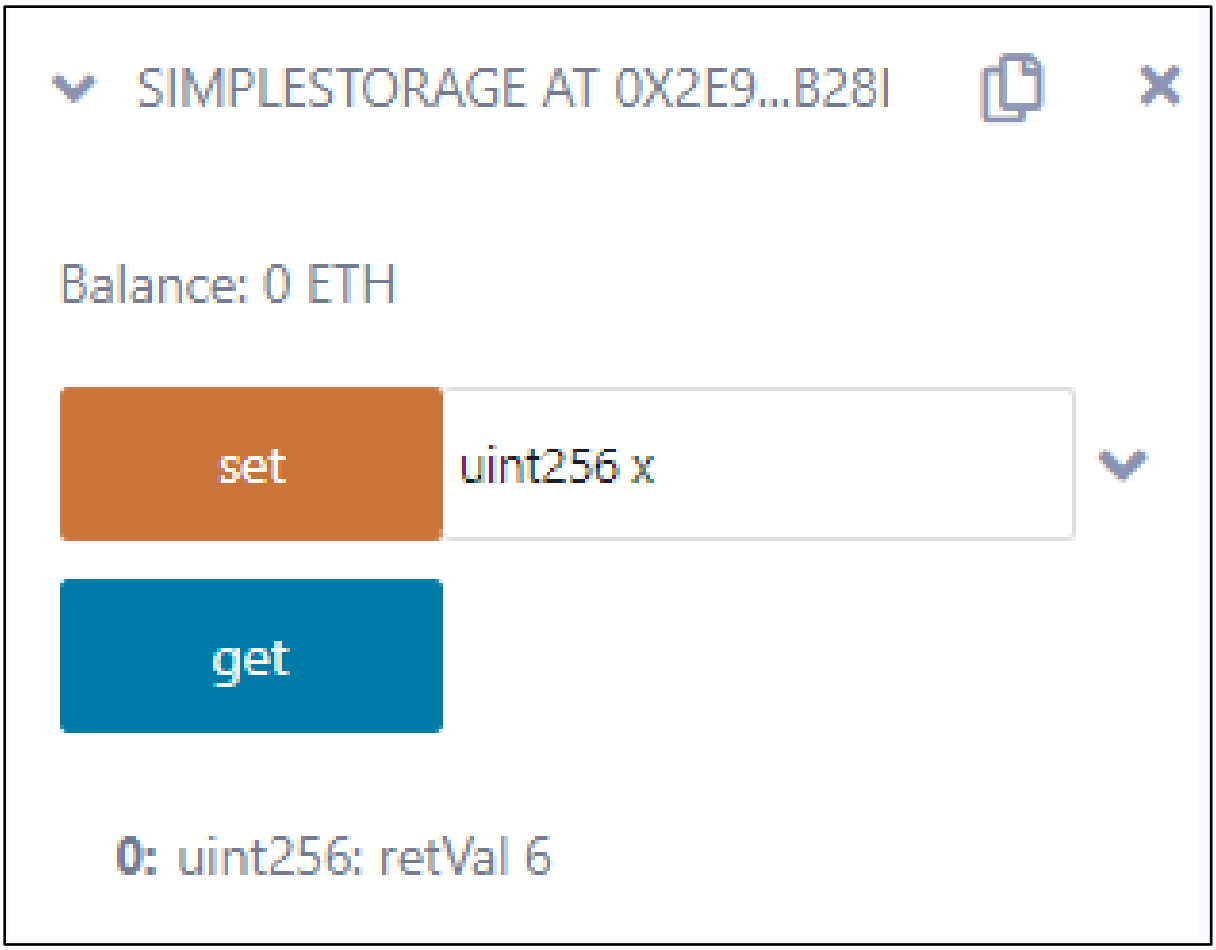
[XI] Ether Units

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:

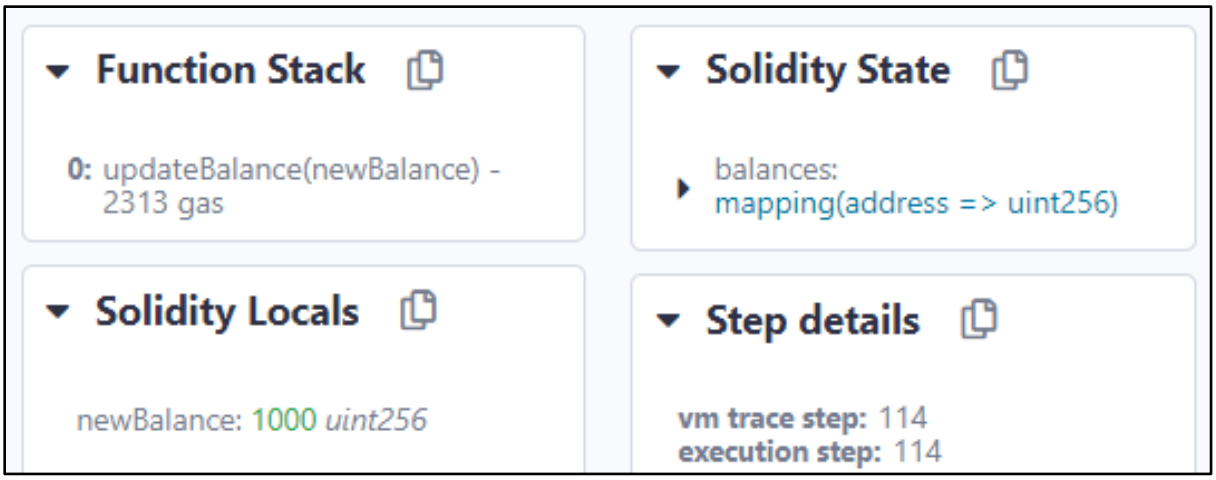


[XII] Special Variables

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:


Function Stack
0: updateBalance(newBalance) - 2313 gas
Solidity State
balances: mapping(address => uint256)
Solidity Locals
newBalance: 1000 uint256
Step details
vm trace step: 114
execution step: 114

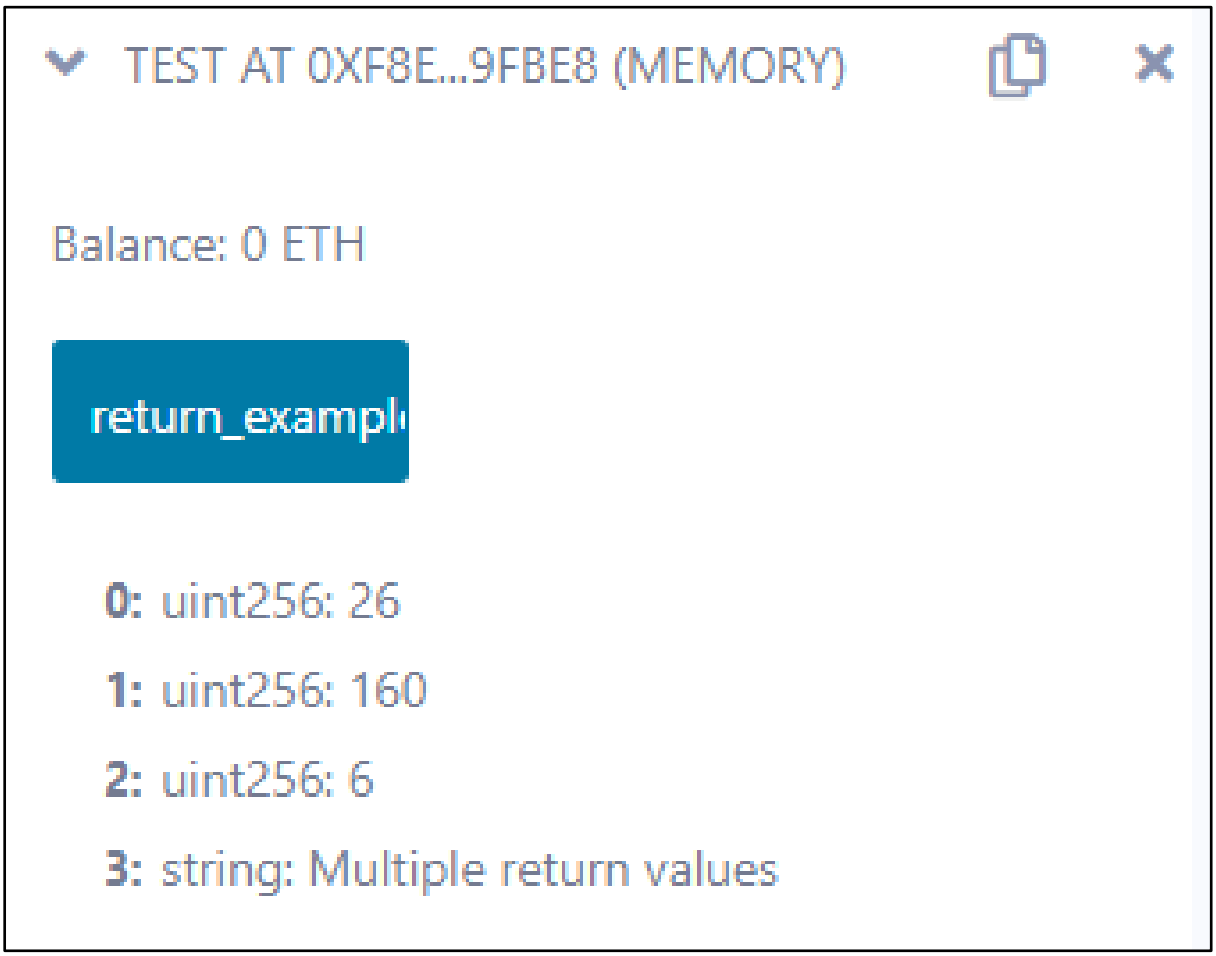
[B] Functions, Function Modifiers, View Functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

[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:

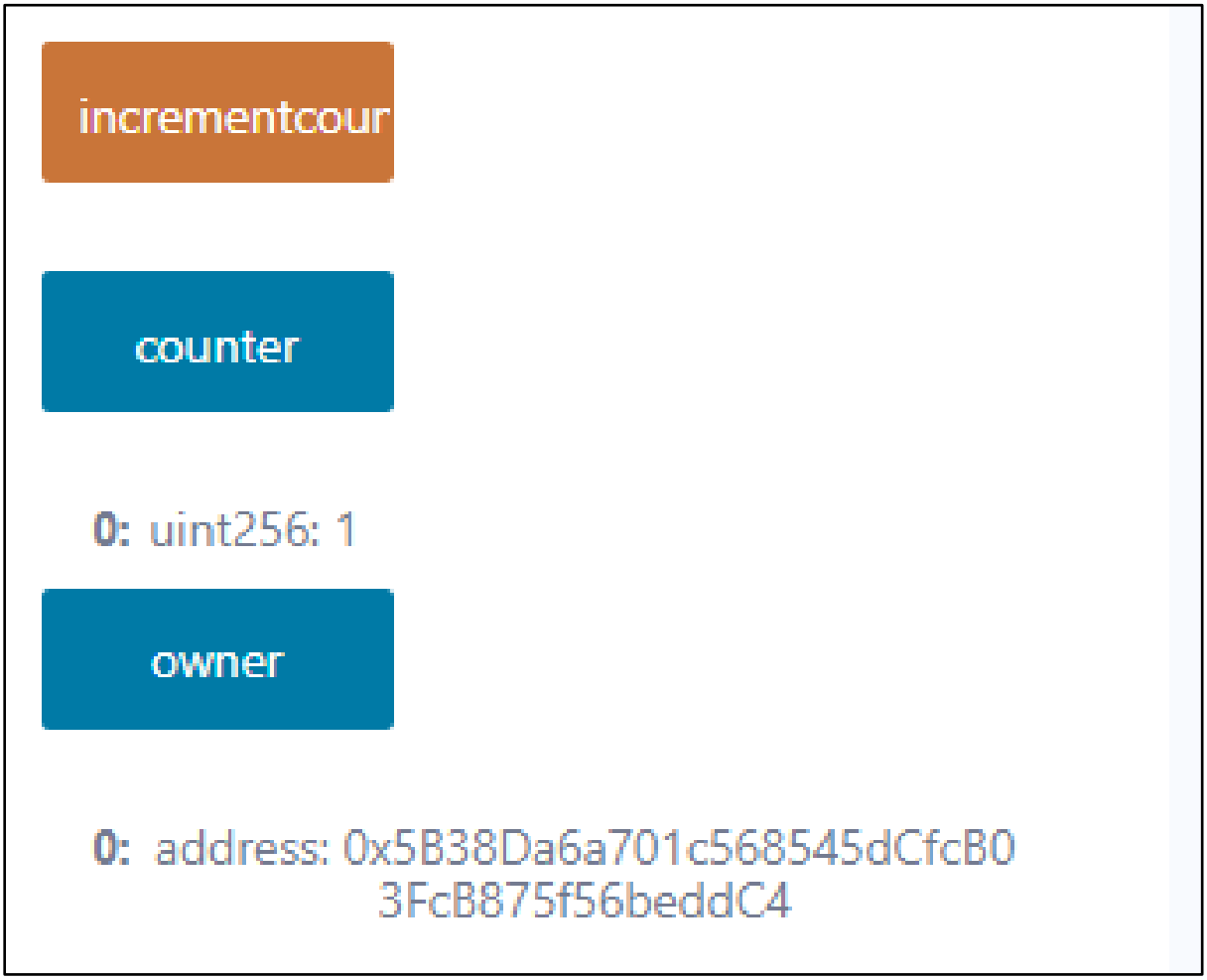

TEST AT 0XF8E...9FBE8 (MEMORY)
Balance: 0 ETH
return_exampl
0: uint256: 26
1: uint256: 160
2: uint256: 6
3: string: Multiple return values

[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:

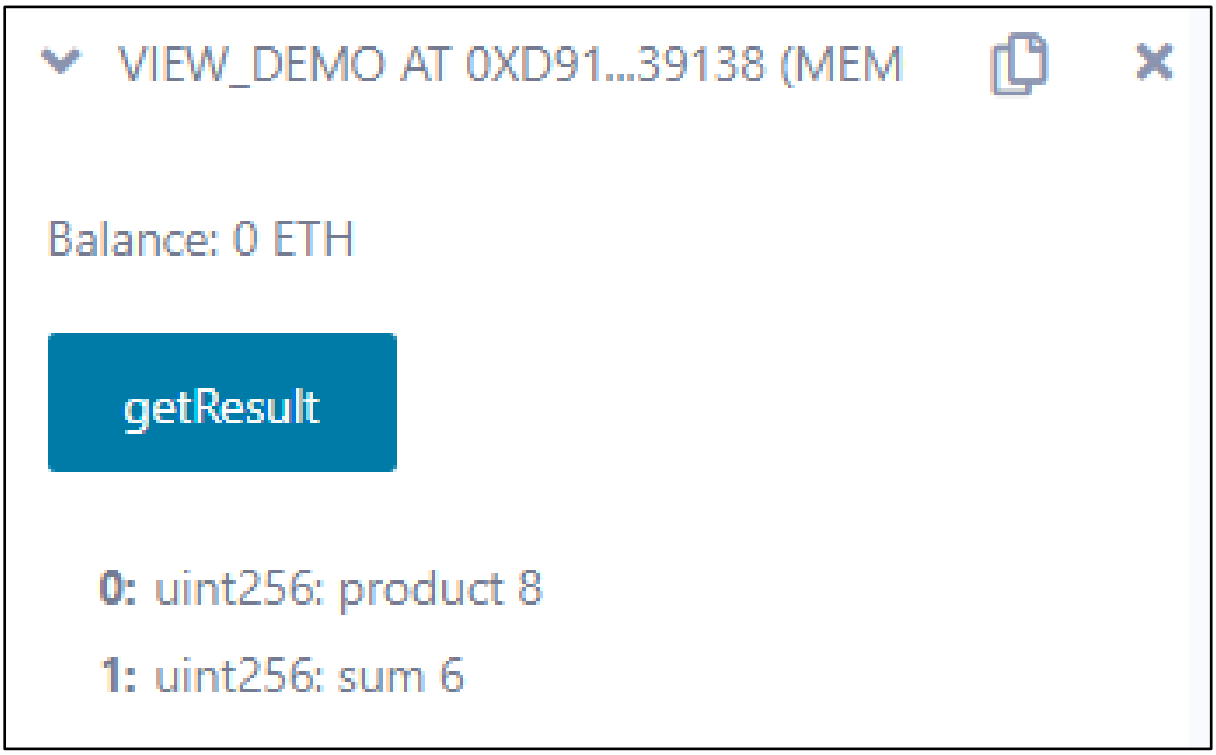

incrementcour
counter
0: uint256: 1
owner
0: address: 0x5B38Da6a701c568545dCfcB0
3FcB875f56beddC4

[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:

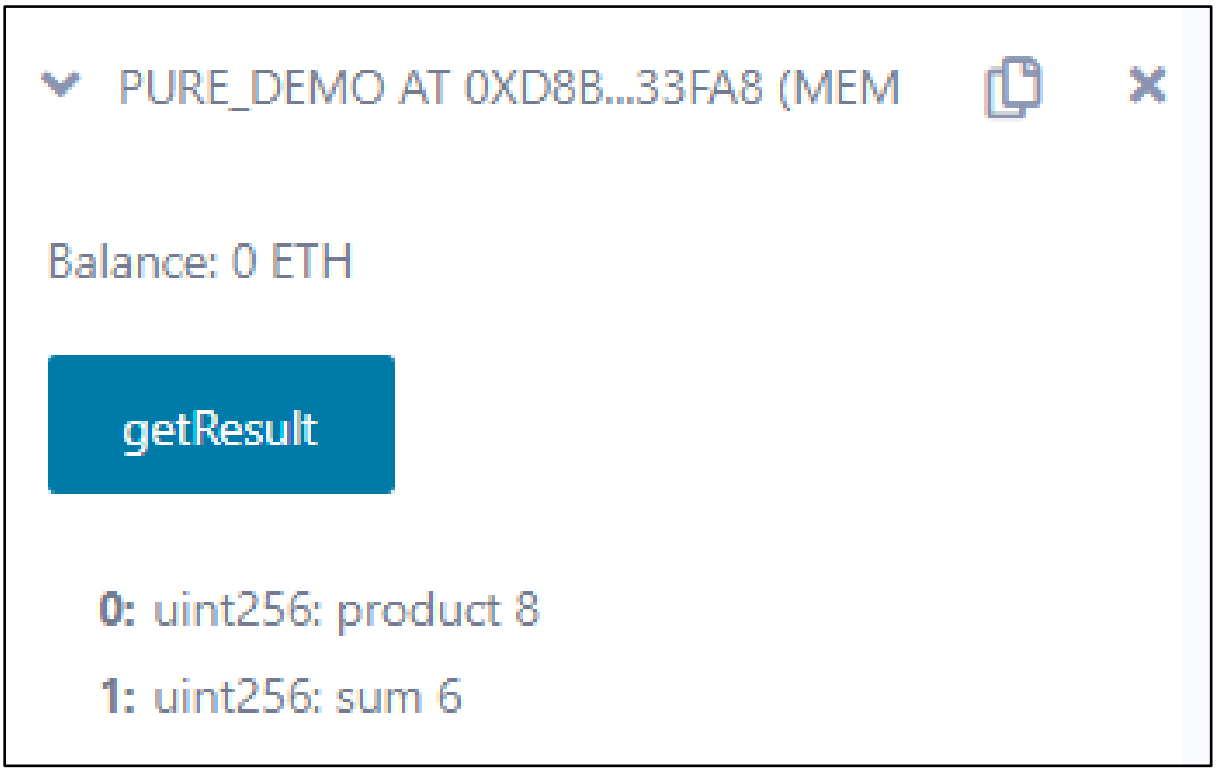


[IV] Pure Functions

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:

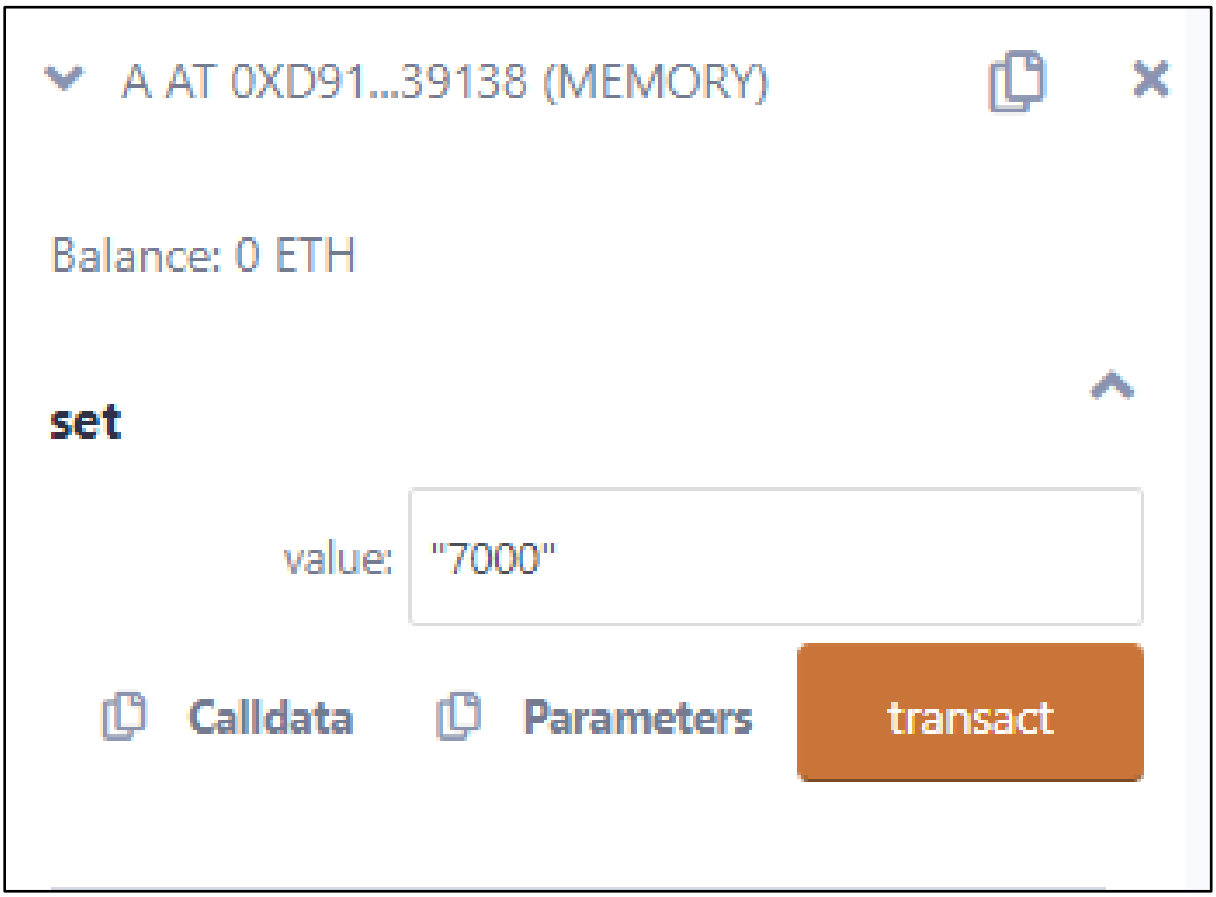


[V] Fallback Function

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"
Calldata
Parameters
transact

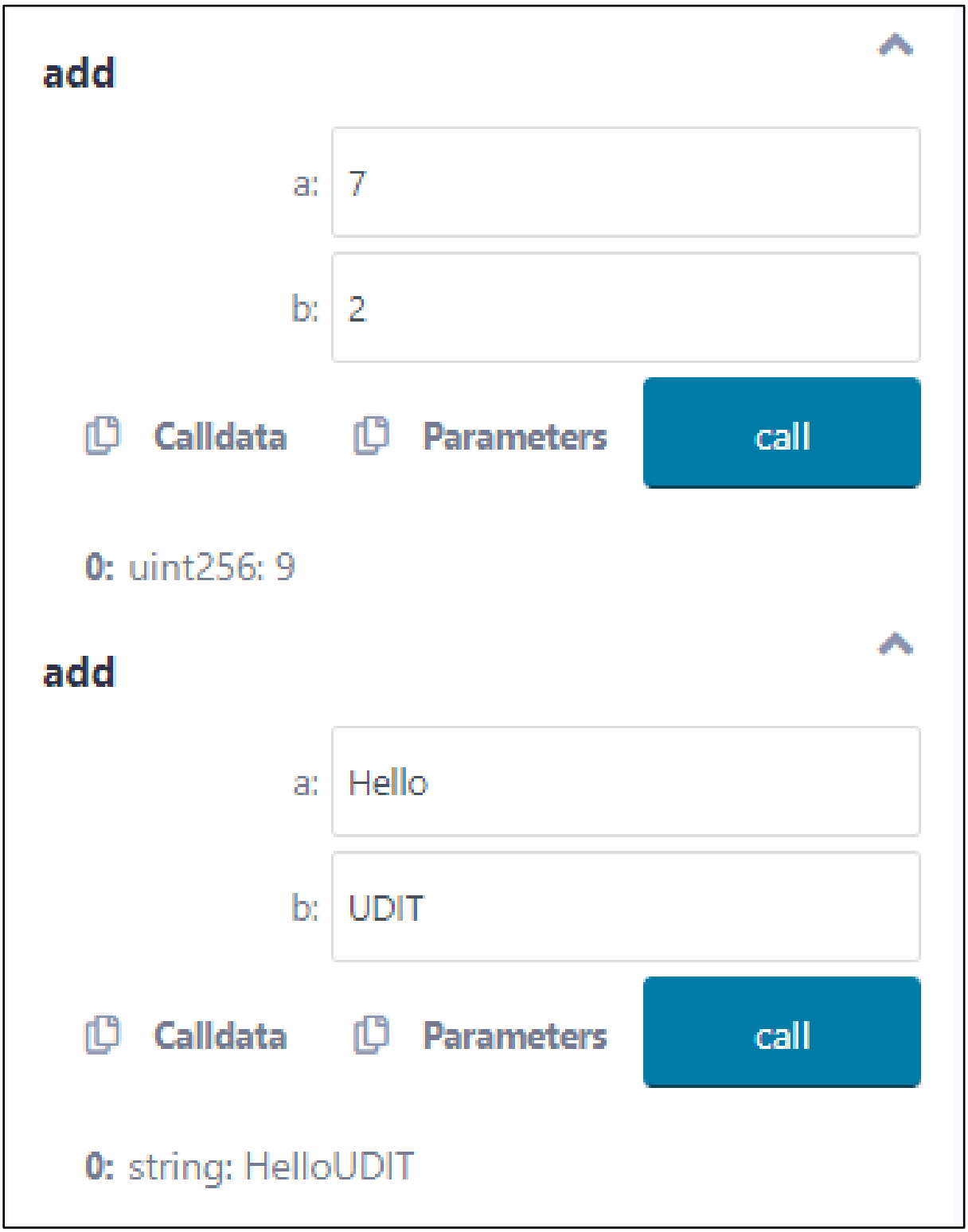

callA
a:
"0xd9145CCE52D386f254917e481eB
Calldata
Parameters
transact

[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:

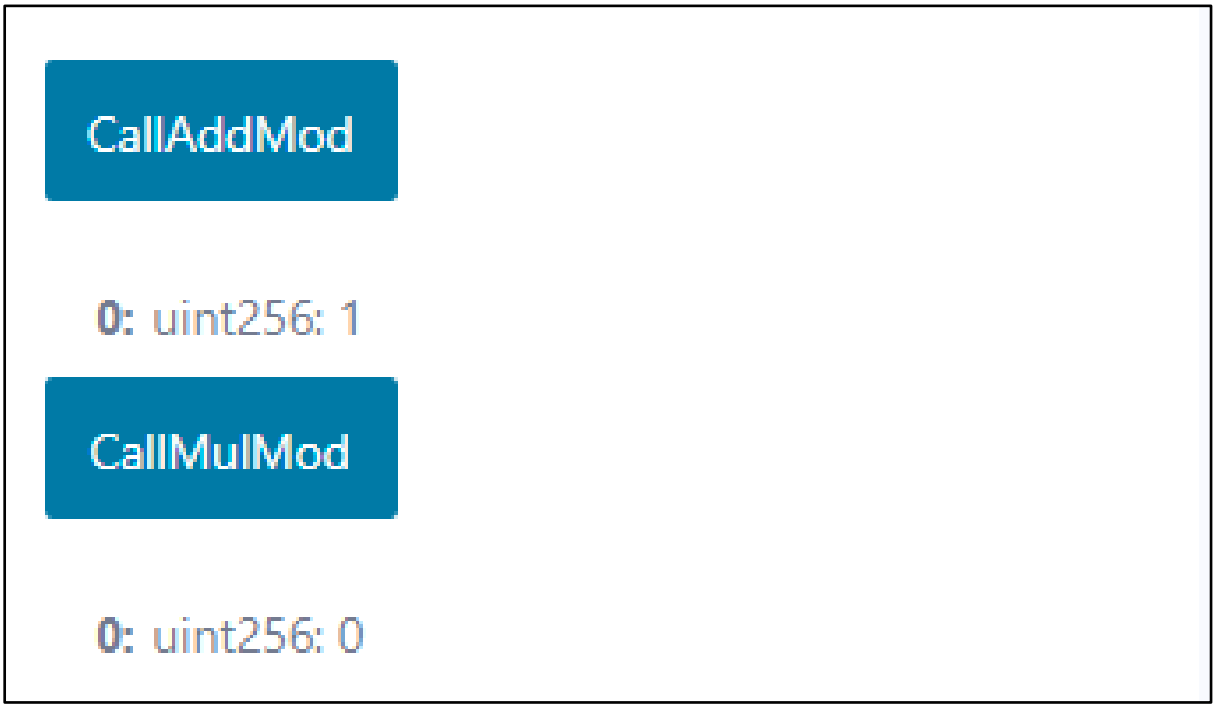

add
a: 7
b: 2
Calldata
Parameters
call
0: uint256: 9
add
a: Hello
b: UDIT
Calldata
Parameters
call
0: string: HelloUDIT

[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:

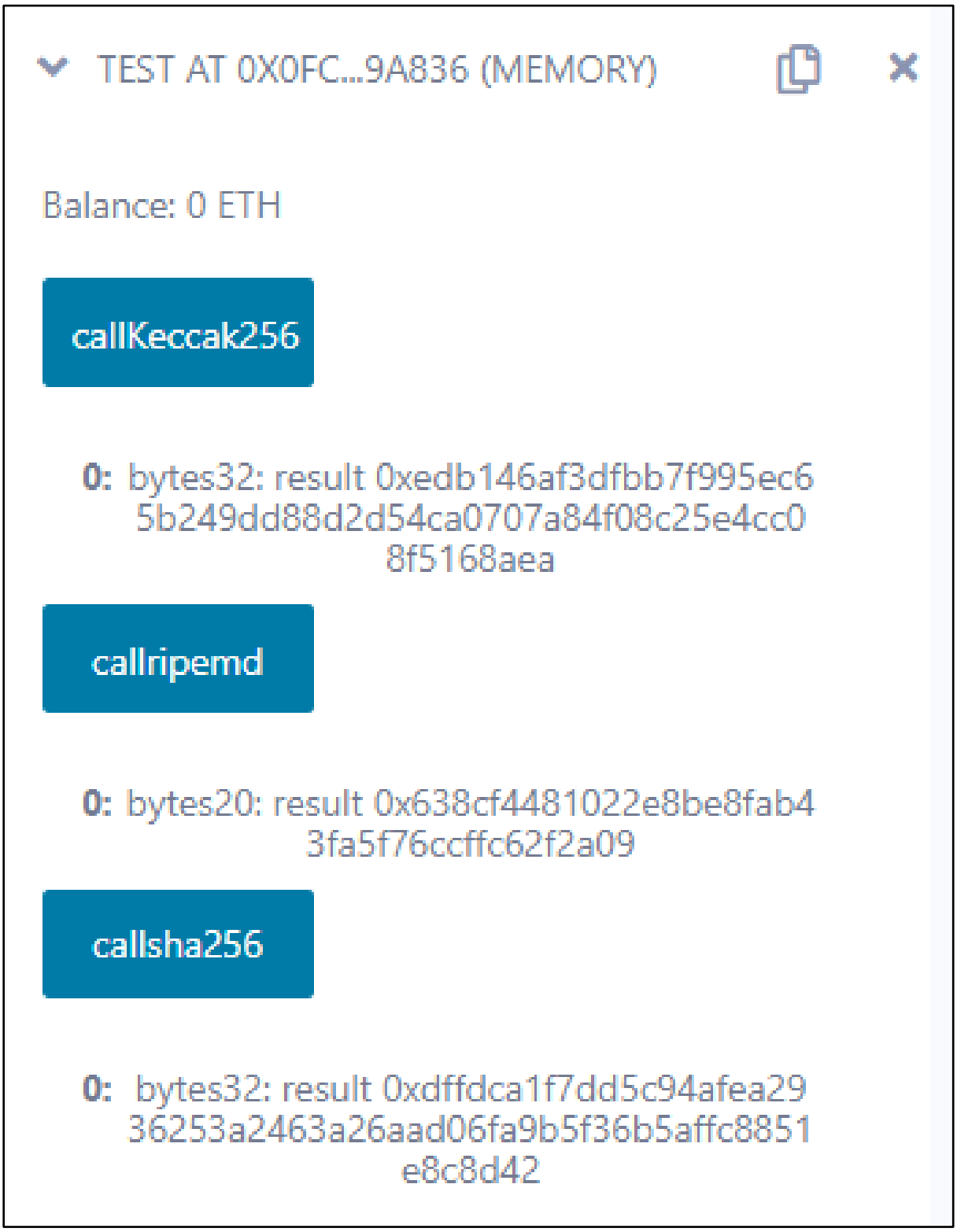


[VIII] Cryptographic Functions

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:


TEST AT 0X0FC...9A836 (MEMORY)
Balance: 0 ETH
callKeccak256
0: bytes32: result 0xedb146af3dfbb7f995ec6 5b249dd88d2d54ca0707a84f08c25e4cc0 8f5168aea
callripemd
0: bytes20: result 0x638cf4481022e8be8fab4 3fa5f76ccffc62f2a09
callsha256
0: bytes32: result 0xdffdca1f7dd5c94afea29 36253a2463a26aad06fa9b5f36b5affc8851 e8c8d42

PRACTICAL 4

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

[A] Withdrawal Pattern, Restricted Access.

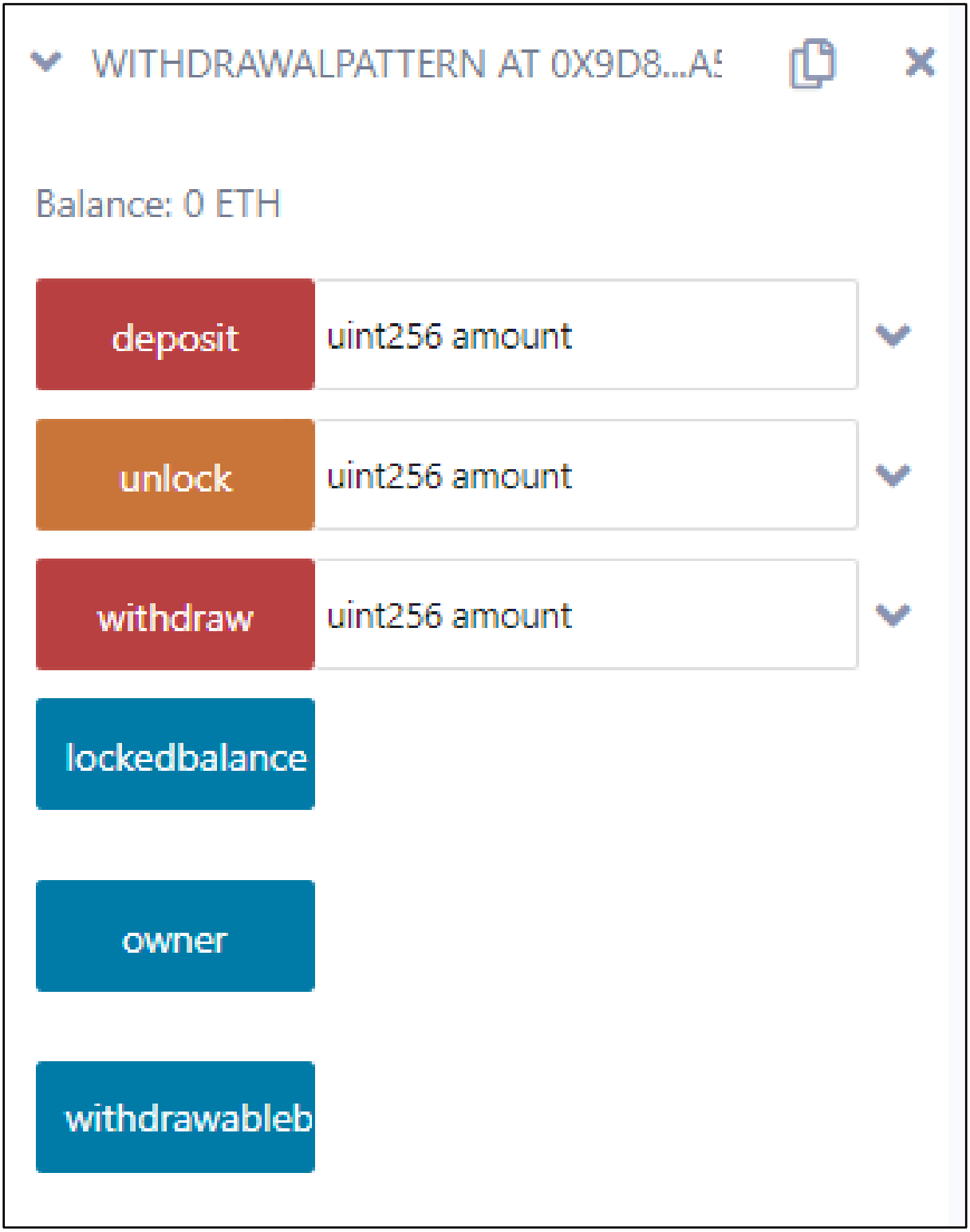
[I] Withdrawal Pattern

CODE:

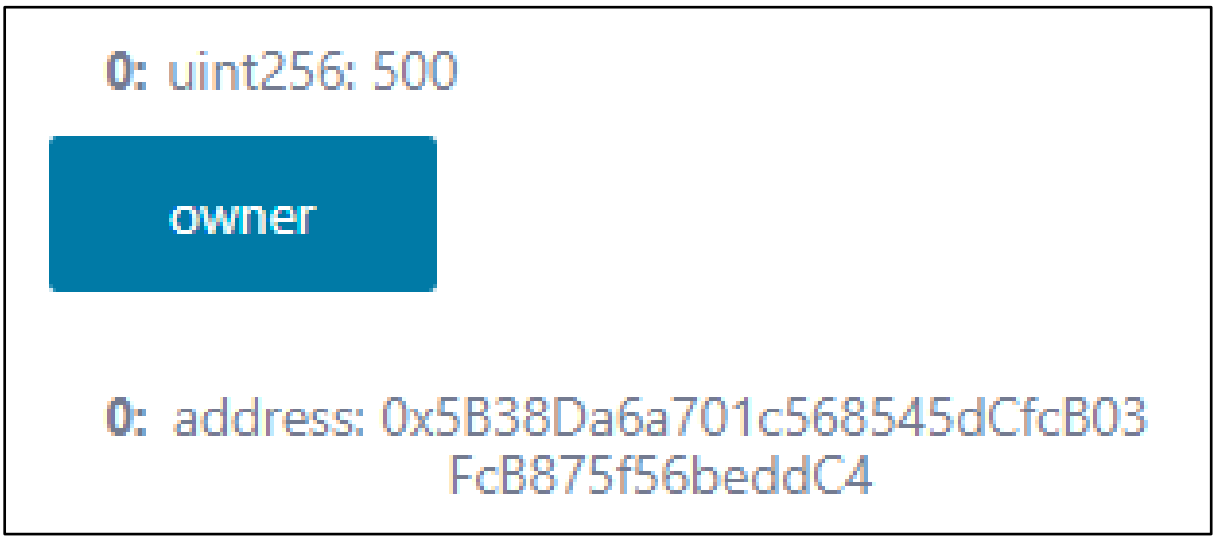
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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:

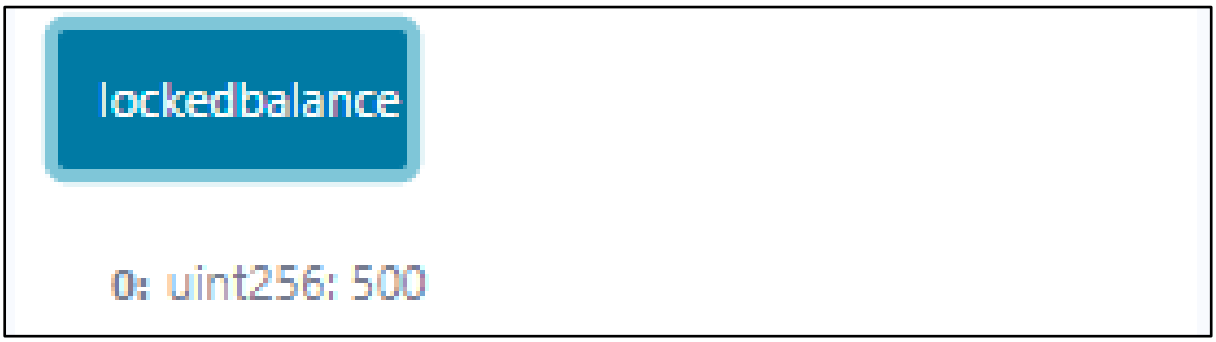

WITHDRAWALPATTERN AT 0X9D8...A
Balance: 0 ETH
deposit
uint256 amount
unlock
uint256 amount
withdraw
uint256 amount
lockedbalance
owner
withdrawableb

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

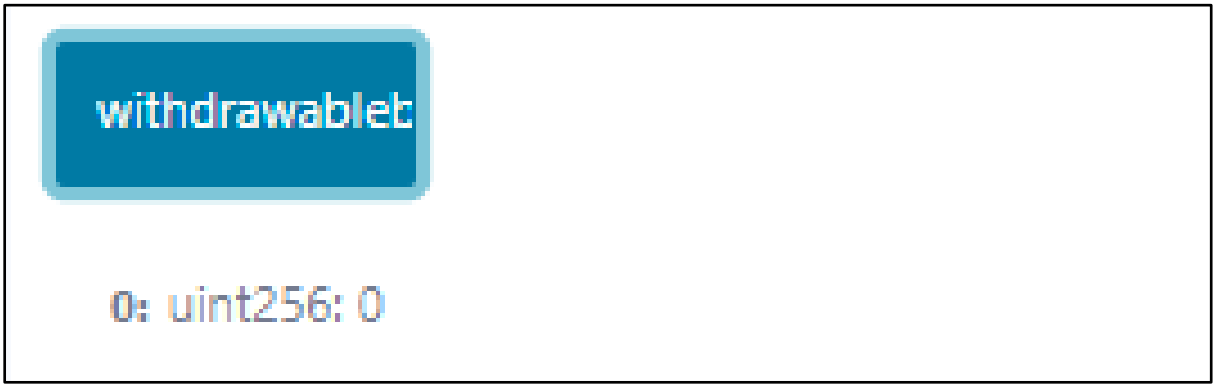


Step 2: Enter an amount and click on deposit.

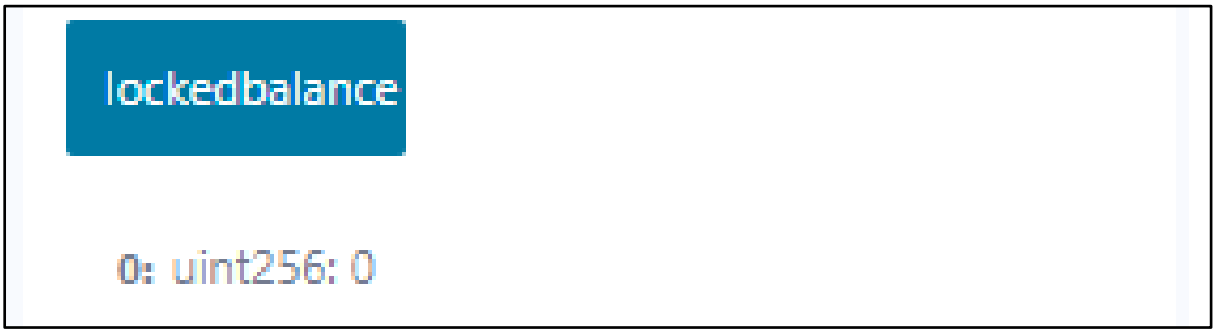
Step 3: Click on locked balance button to display the locked amount in the account.



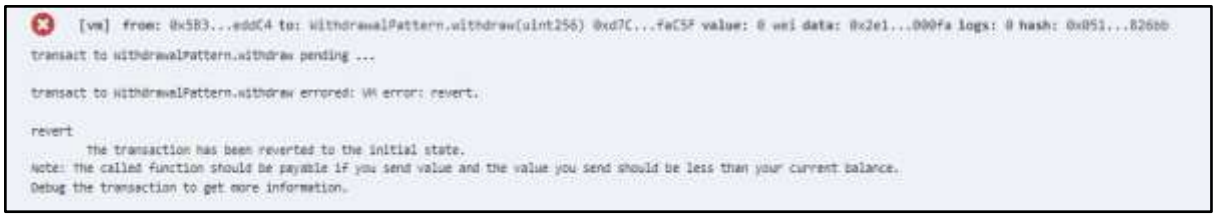
Step 4: Click on withdrawable balance button.



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



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.



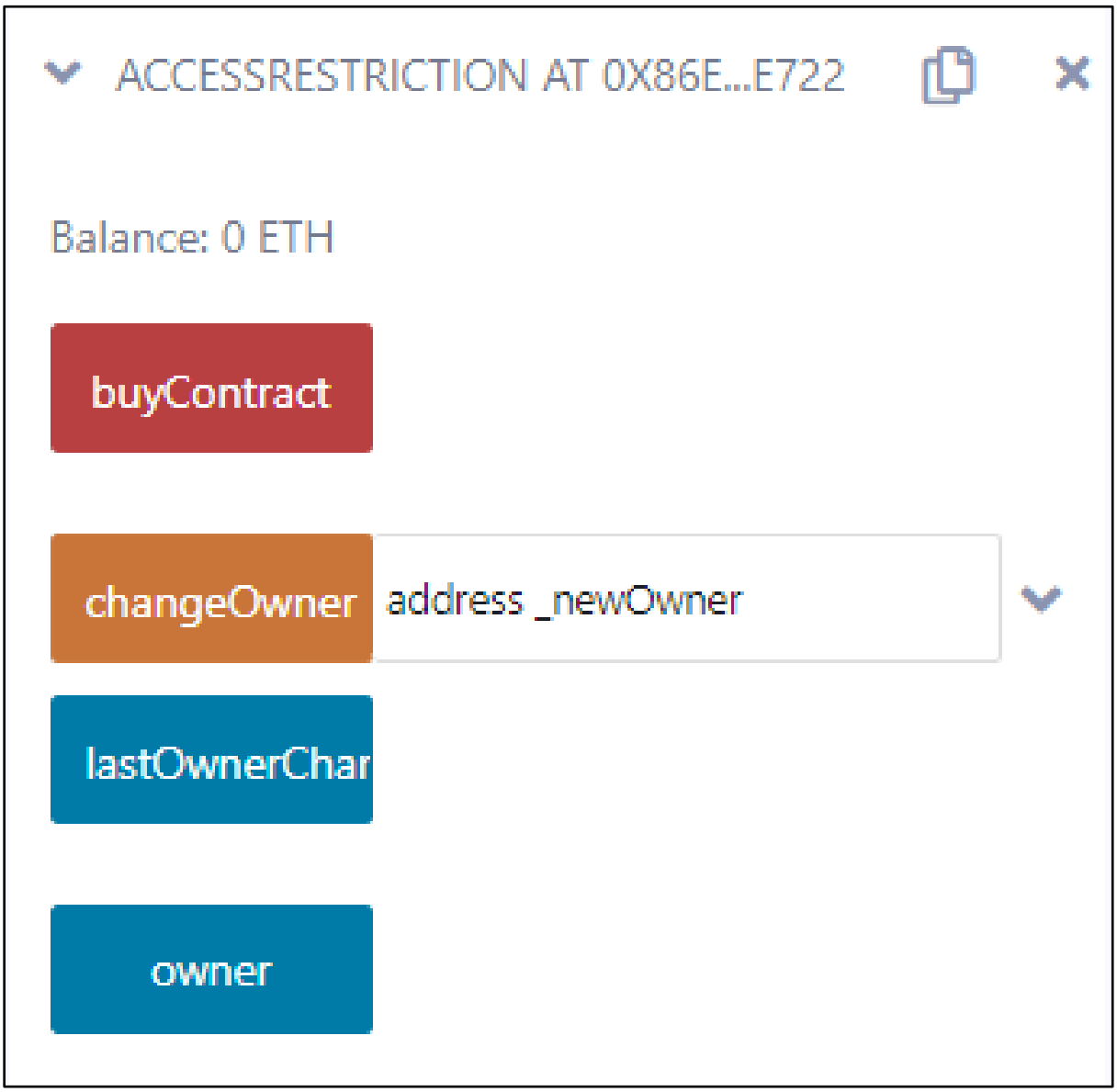
[II] Restricted Access

CODE:

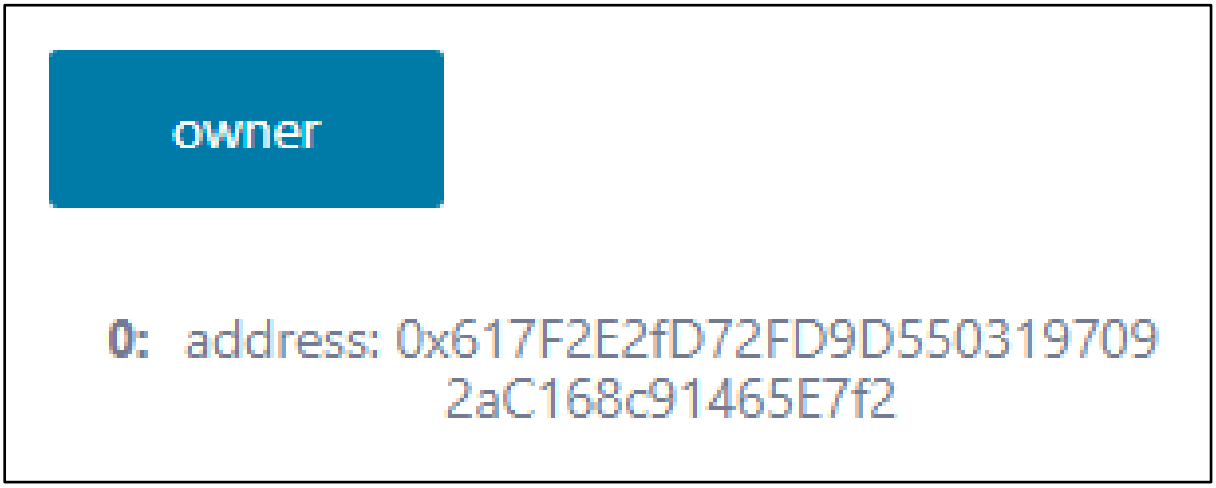
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

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:

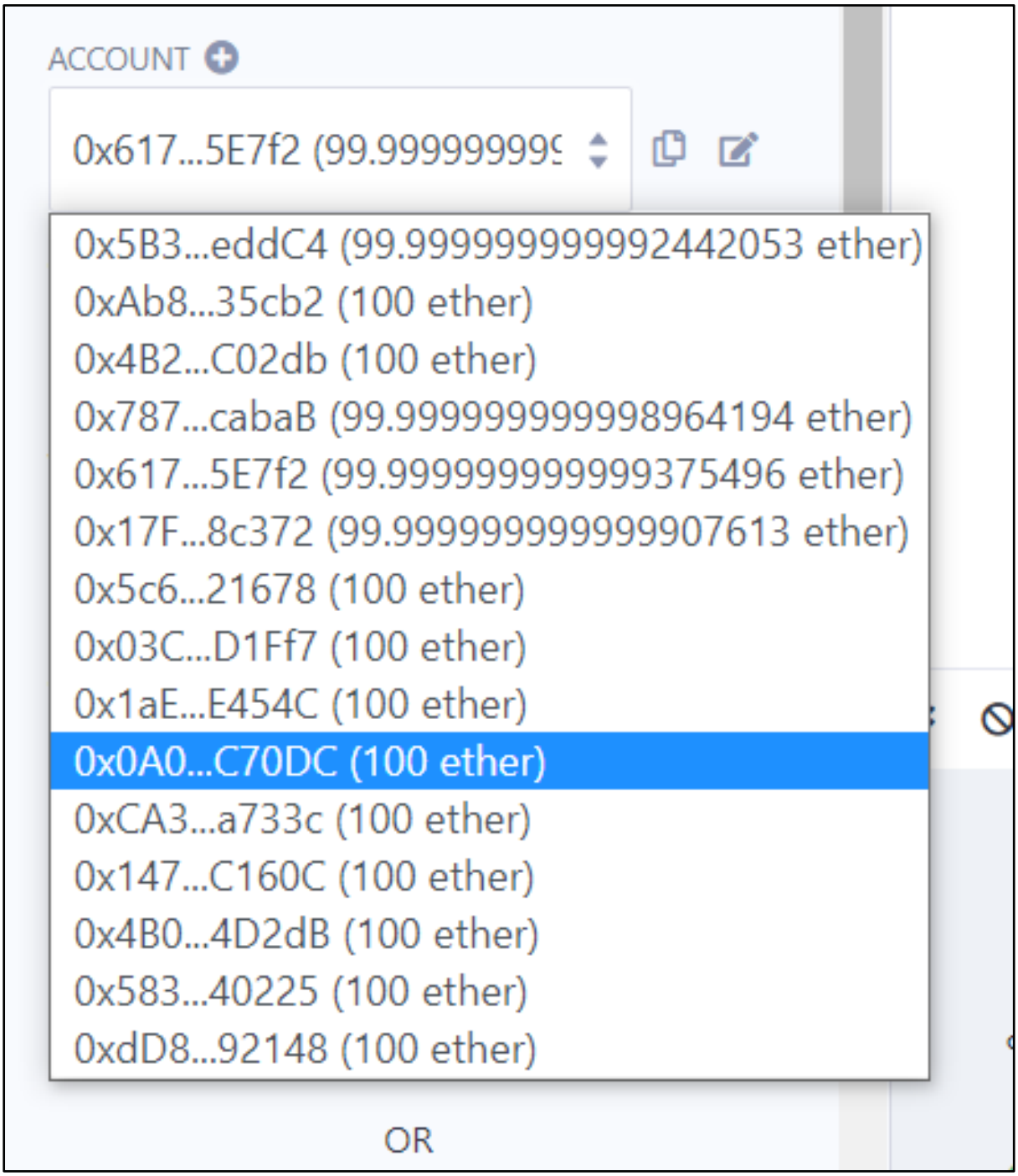

ACCESSRESTRICTION AT 0X86E...E722
Balance: 0 ETH
buyContract
changeOwner
address _newOwner
lastOwnerChar
owner

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

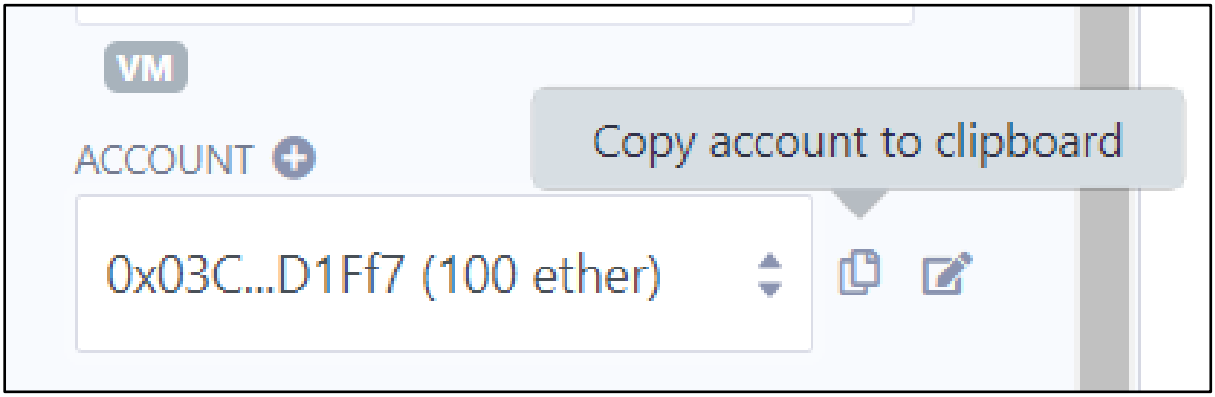


Step 2: Click on lastOwnerChange button.

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

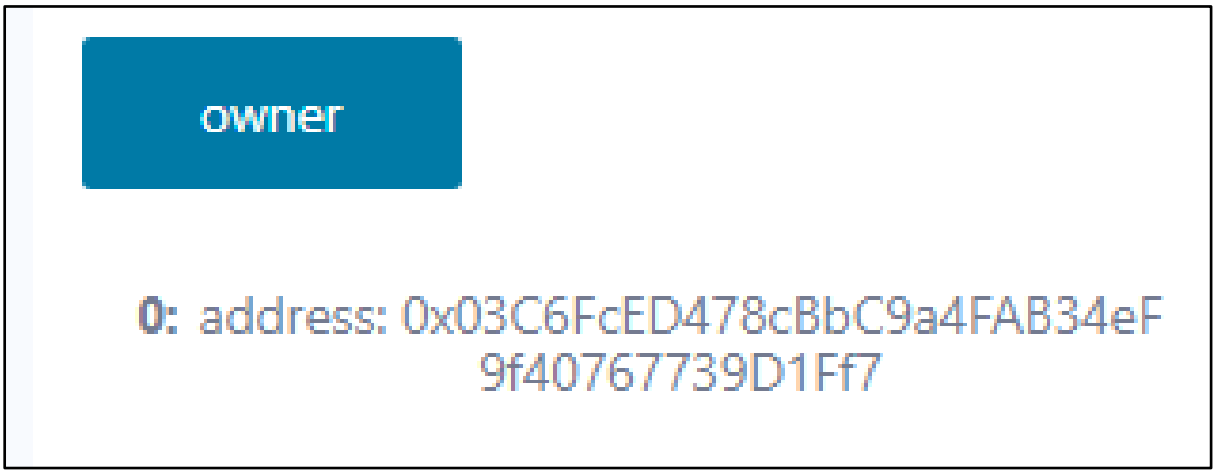


Step 4: Copy and paste the address in changeOwner input and click on changeOwner.



Step 5: You should get an error as following.

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



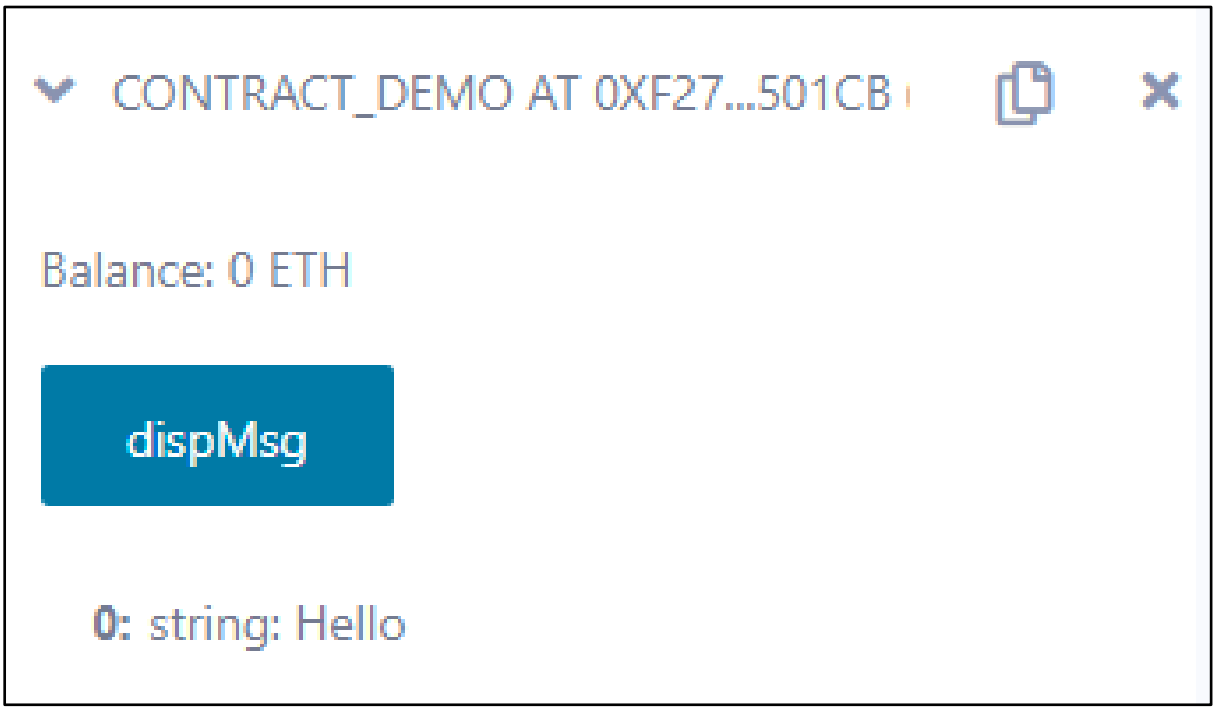
[B] Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.

[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:



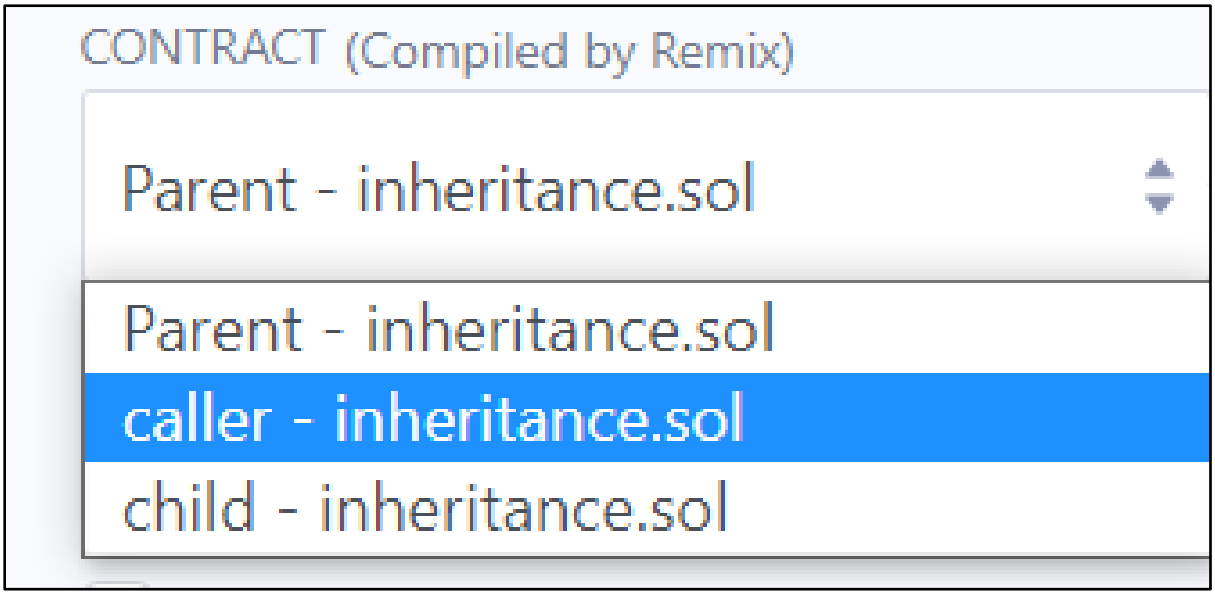
[II] Inheritance

CODE:

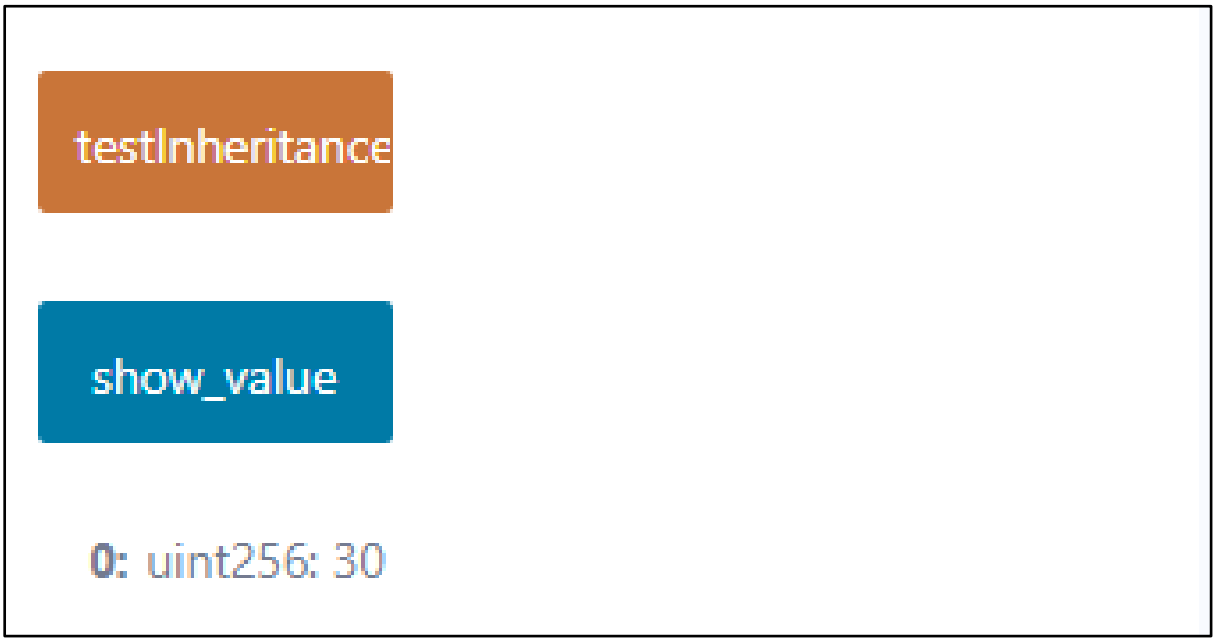
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

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.

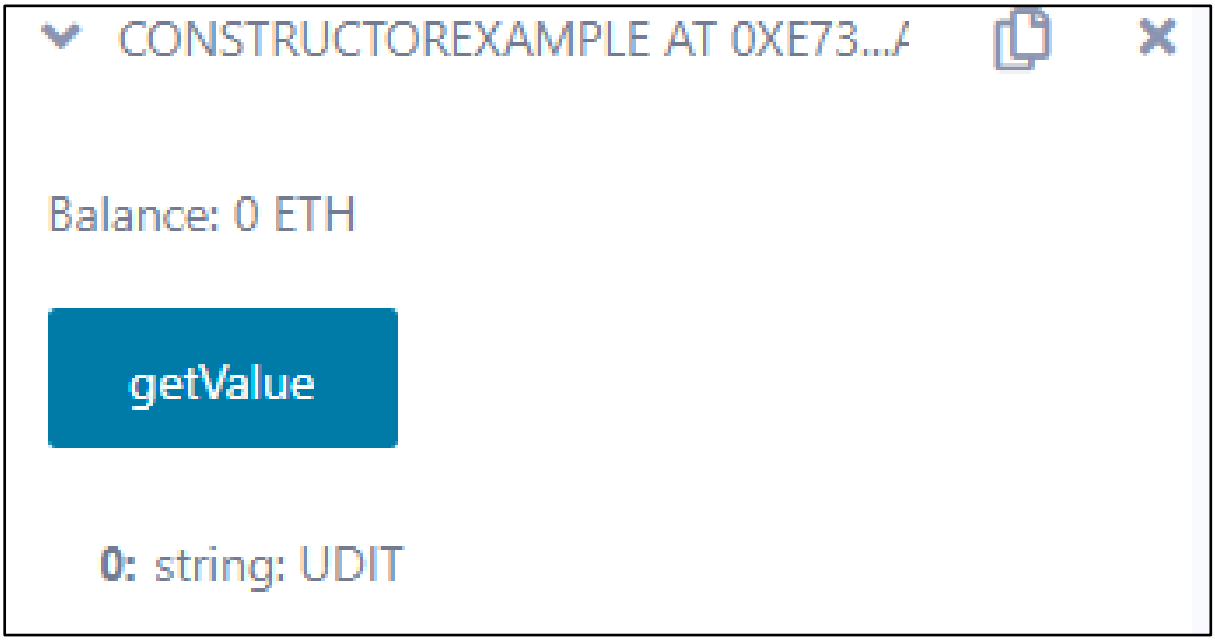


[III] Constructors

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:



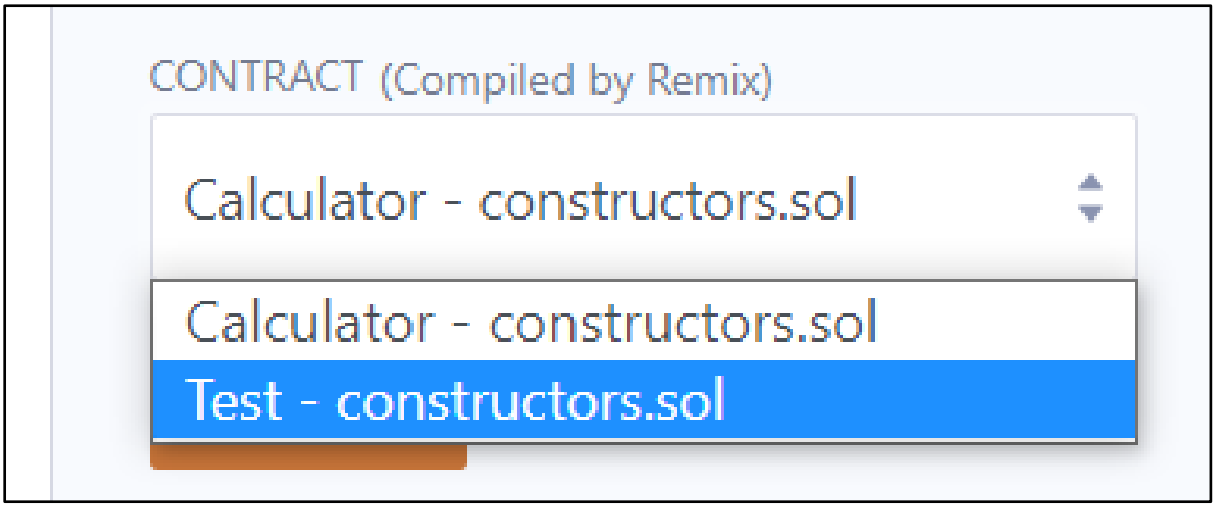
[IV] Abstract Contracts

CODE:

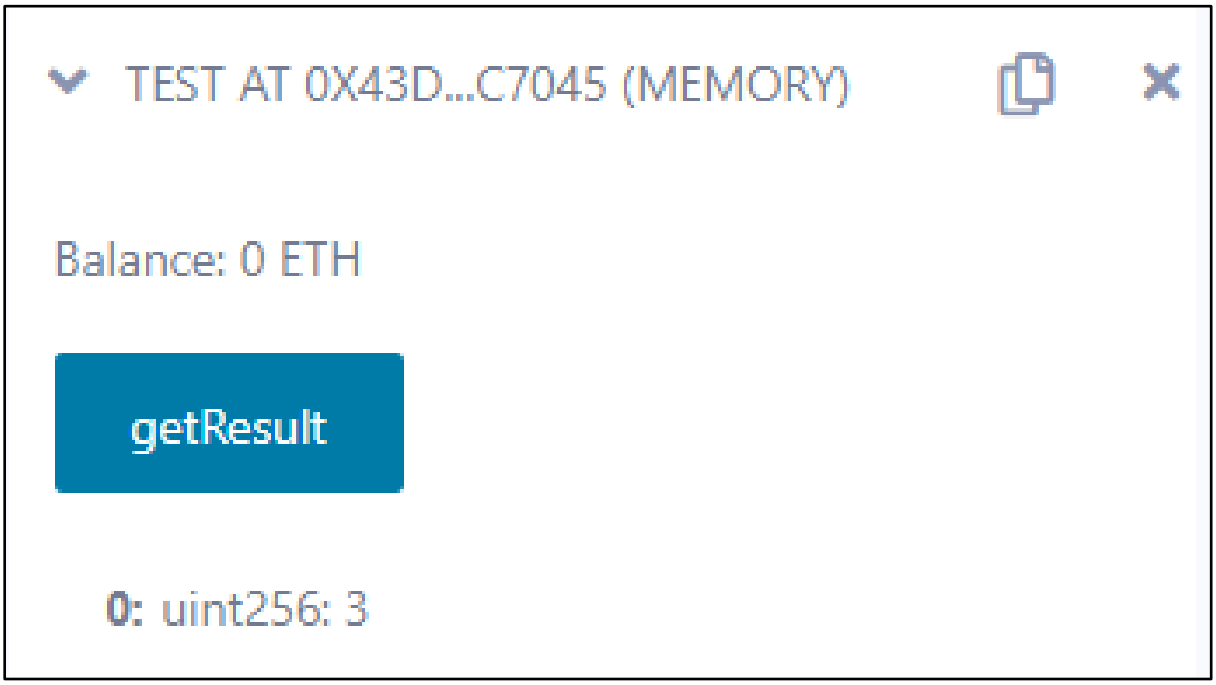
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

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.



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



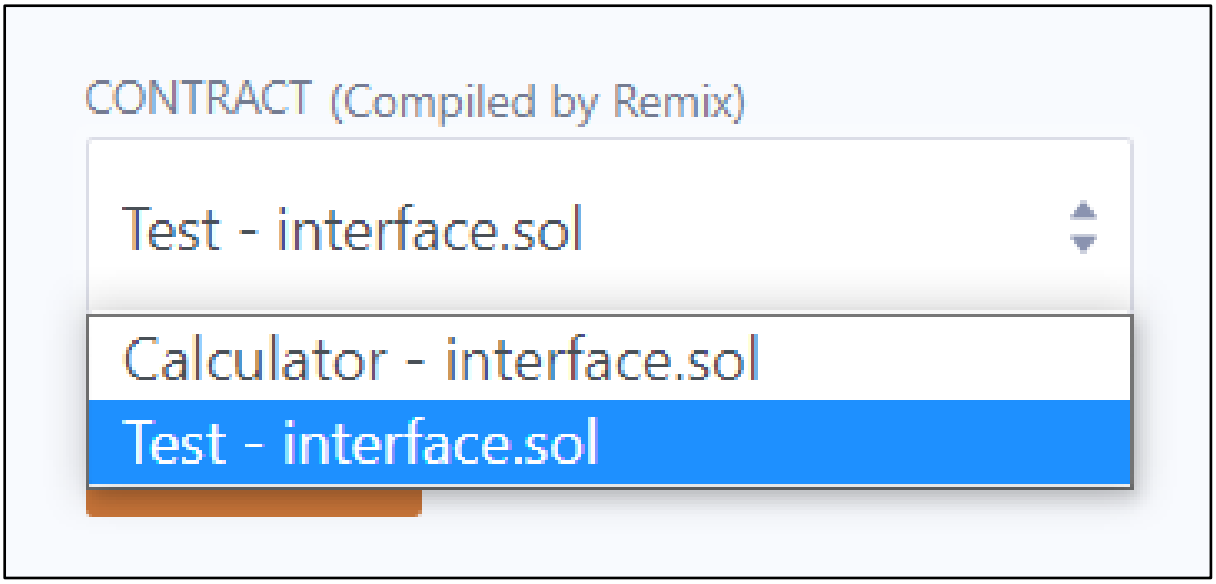
[V] Interfaces

CODE:

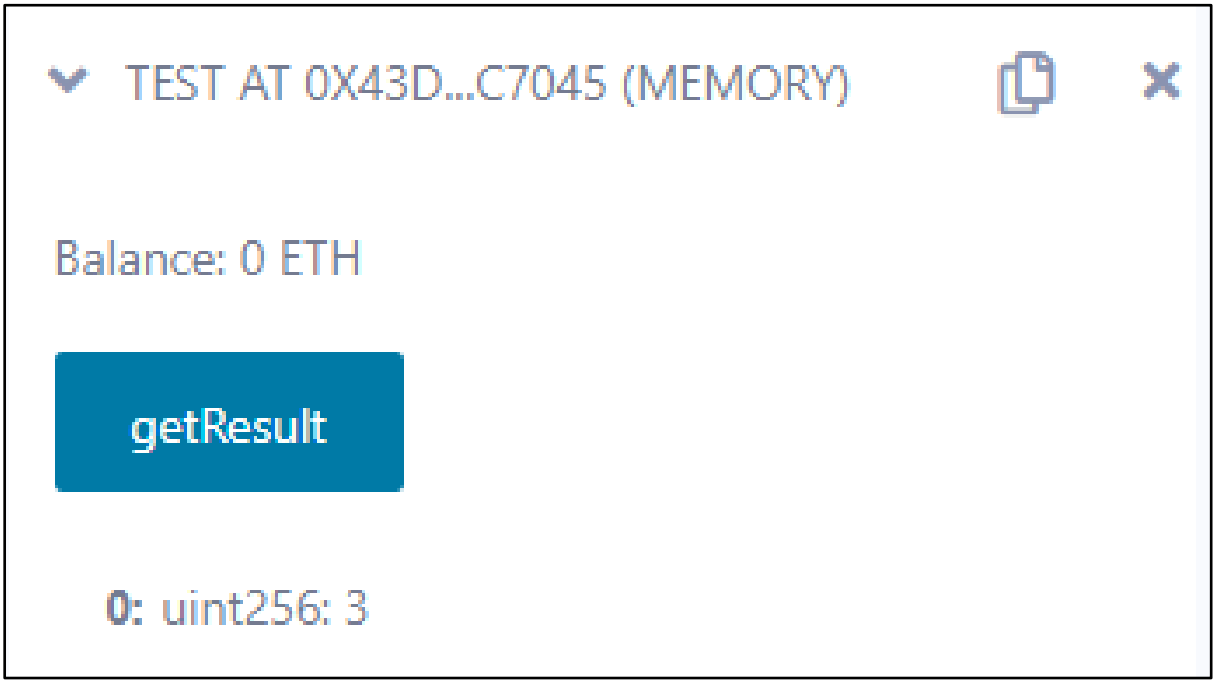
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

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.



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



[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;
  }
}
```

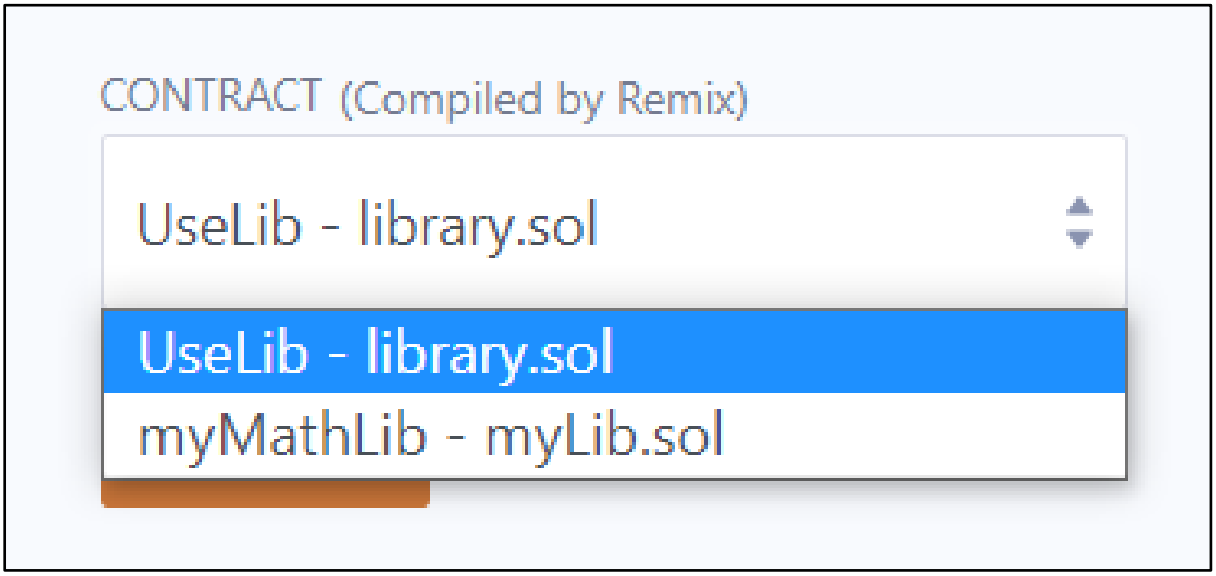
Using the library myLib.sol

```
// 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) {
```

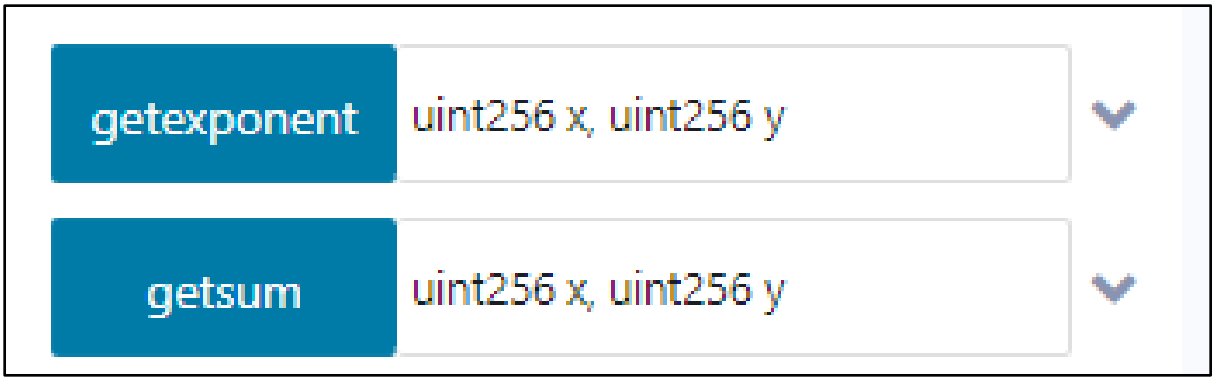
```
    return myMathLib.exponent(x, y);
  }
}
```

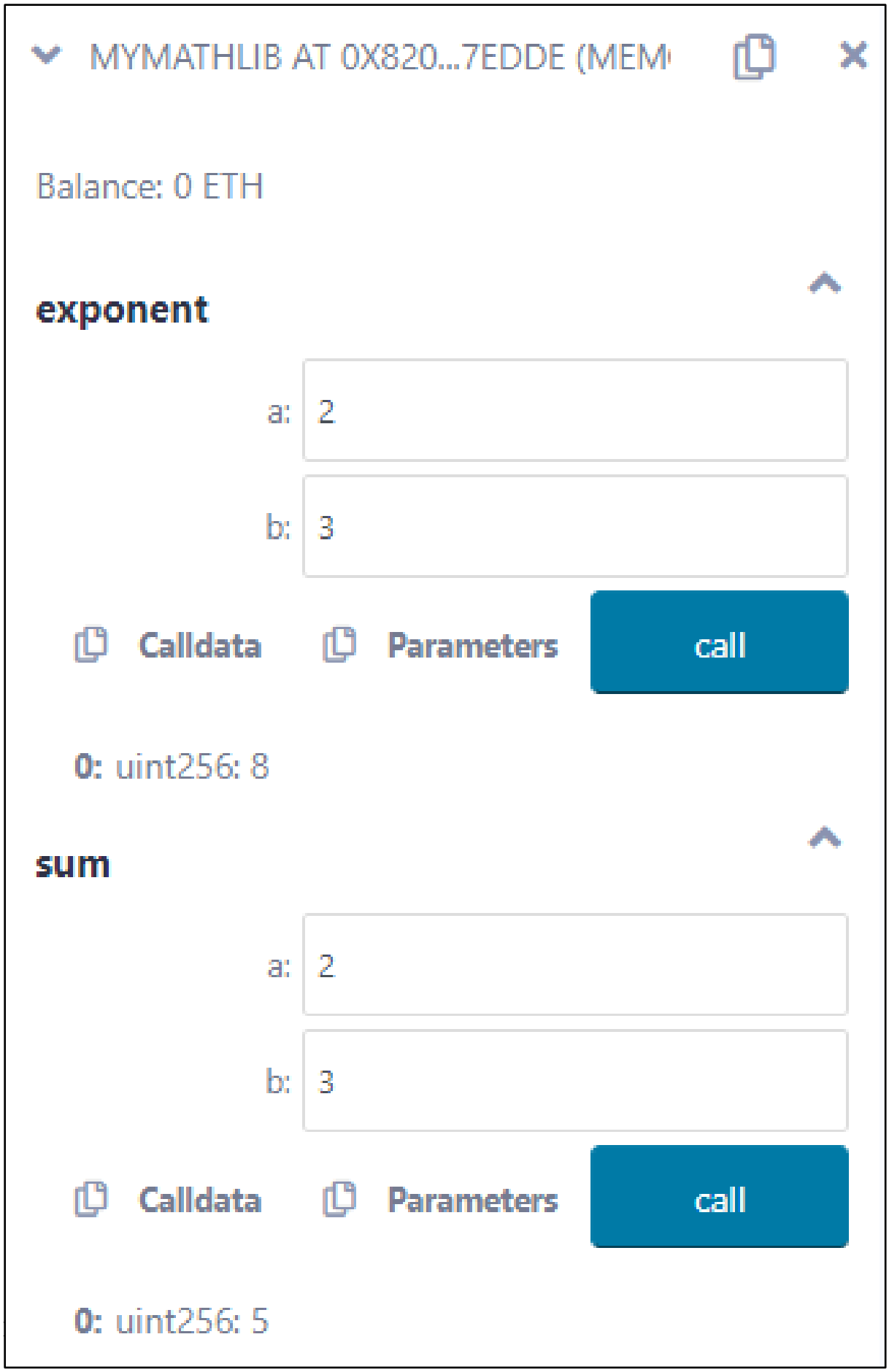
OUTPUT:

Step 1: Change contract to UseLib and deploy.



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



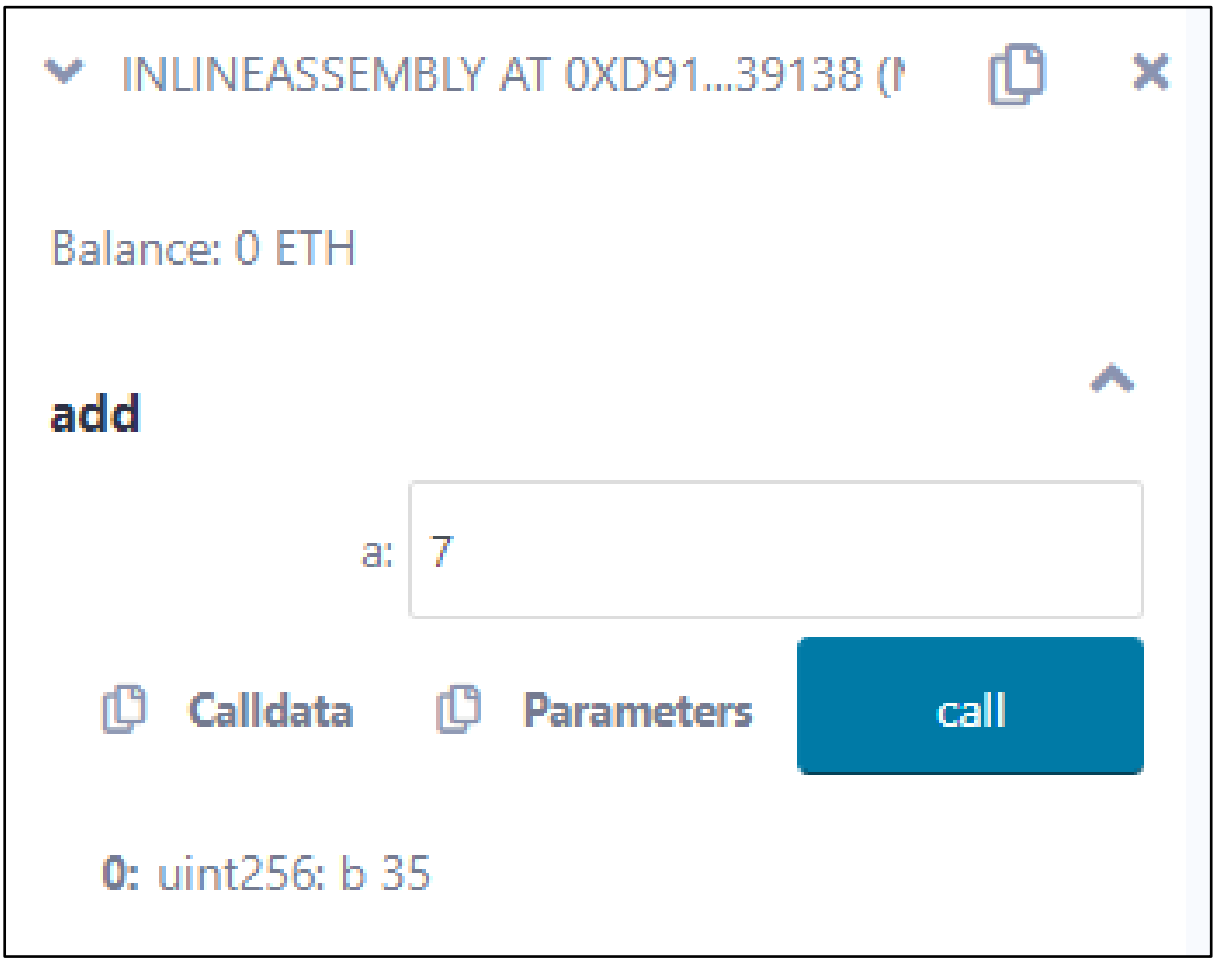

MYMATHLIB AT 0X820...7EDDE (MEM
Balance: 0 ETH
exponent
a: 2
b: 3
Calldata
Parameters
call
0: uint256: 8
sum
a: 2
b: 3
Calldata
Parameters
call
0: uint256: 5

[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:


INLINEASSEMBLY AT 0XD91...39138 (
Balance: 0 ETH
add
a: 7
Calldata
Parameters
call
0: uint256: b 35

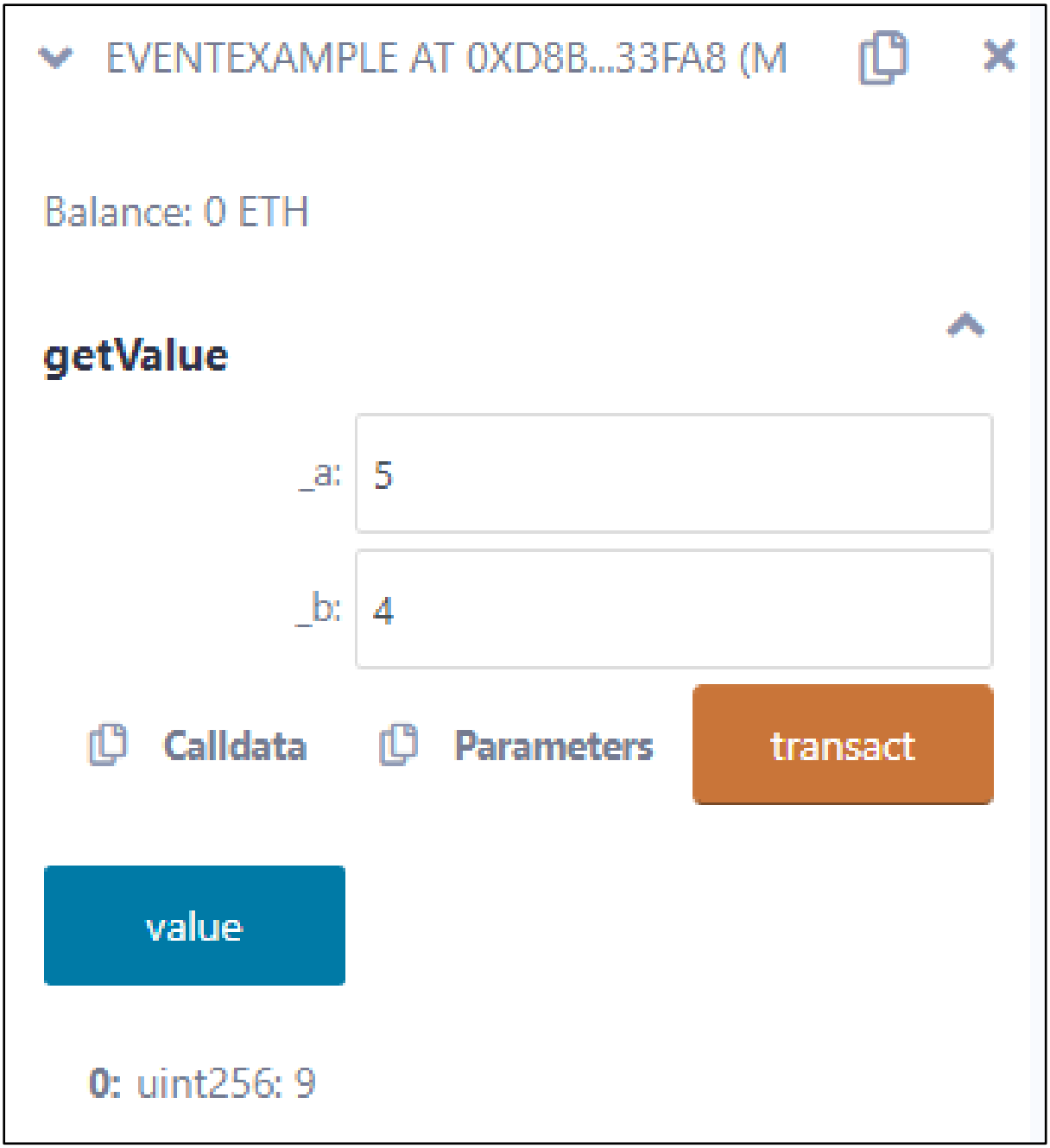
[III] Events

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.



Step 2: In the terminal, check for logs.

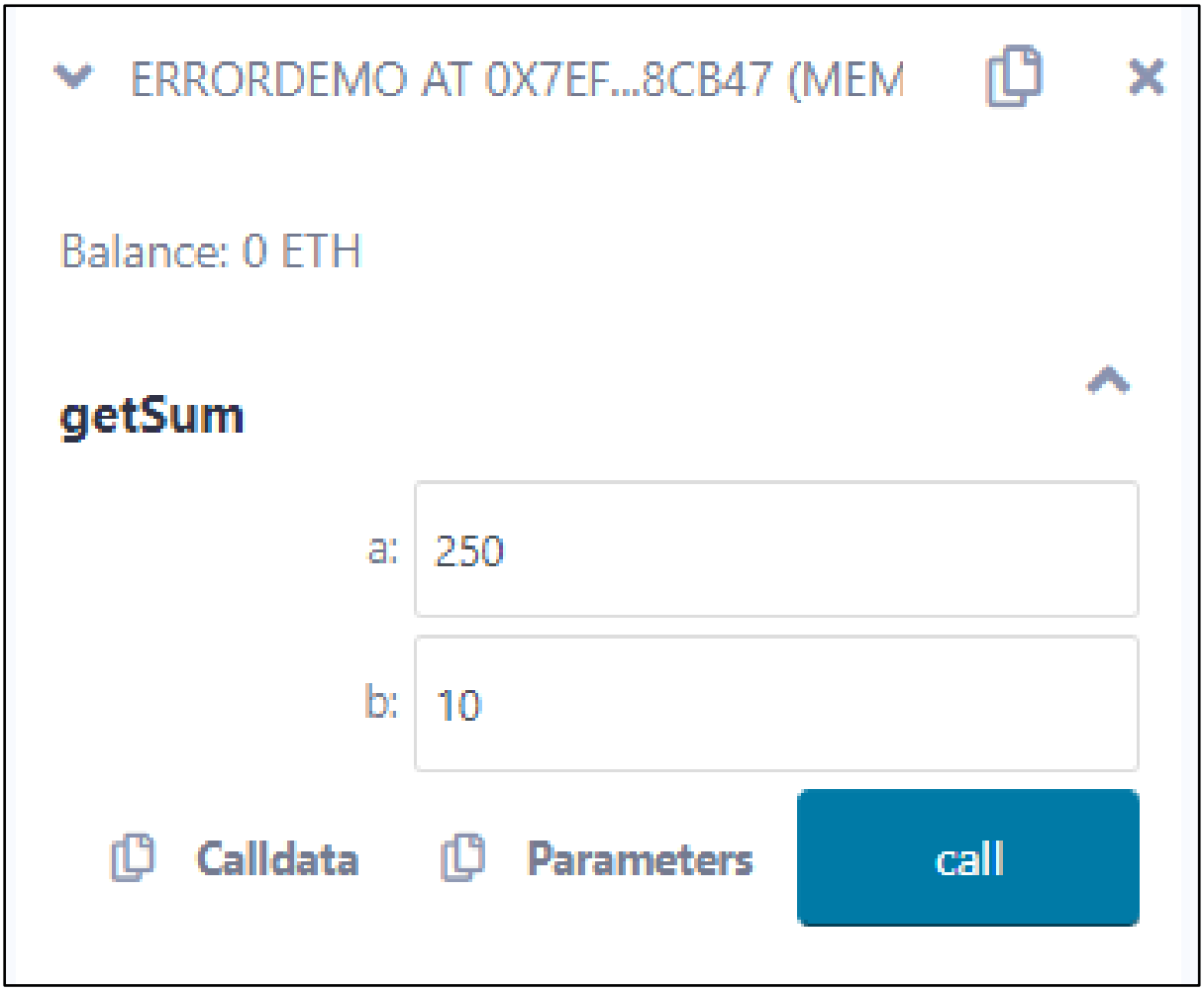
[IV] Error handling

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.



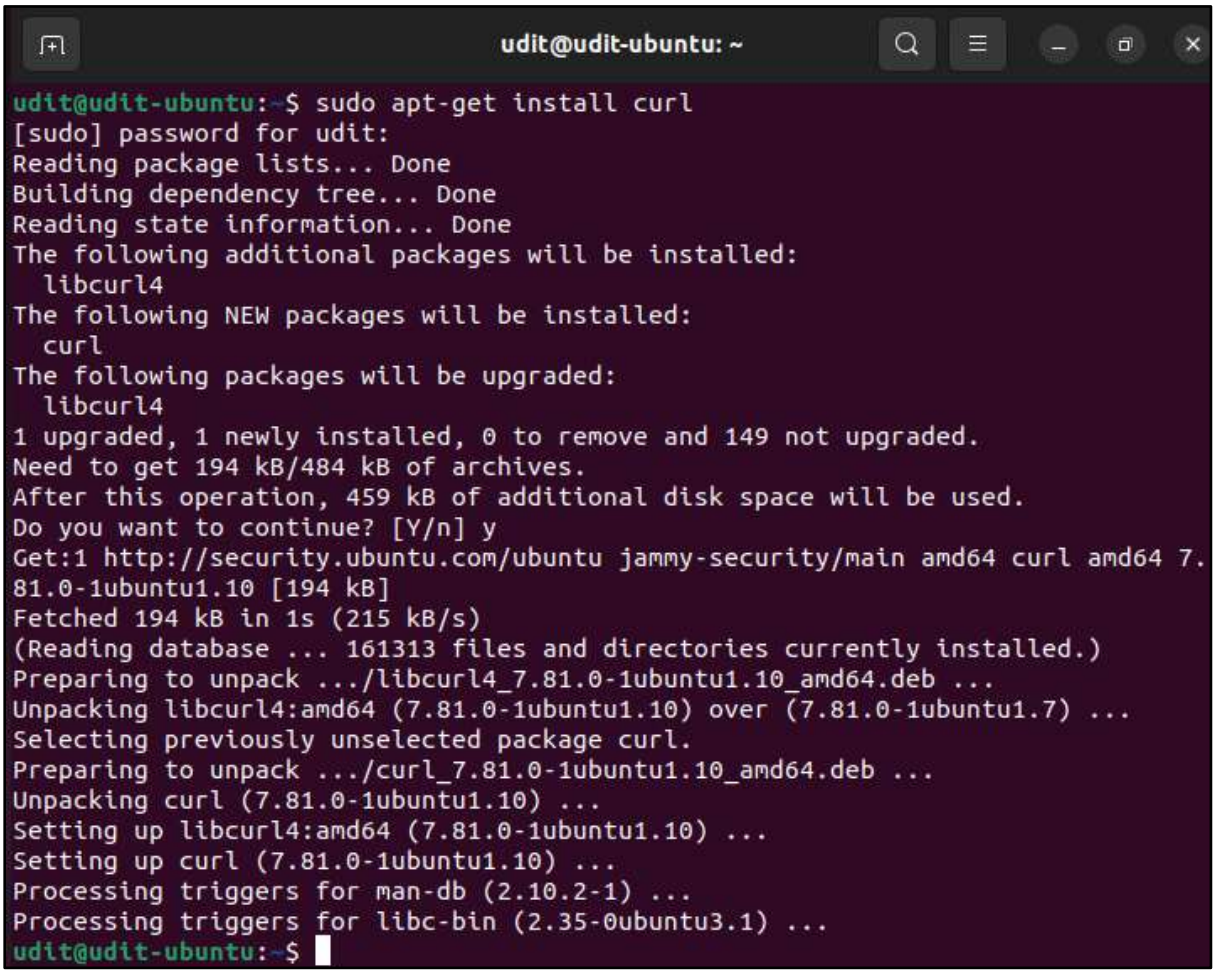
Step 2: Check terminal panel.

PRACTICAL 5

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

Step 1: Install Docker on Ubuntu

sudo apt-get install curl



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 \

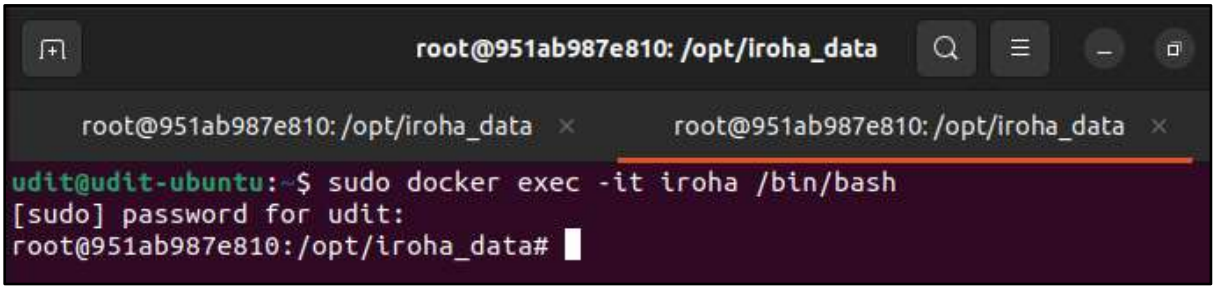
--genesis_block genesis.block \

--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



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'}
```

University of Mumbai

PRACTICAL JOURNAL – ELECTIVE I

PSIT4P2a

Natural Language Processing

SUBMITTED BY

Khan Hajra Mohammed Rashid

SEAT NO 40150

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 Ms. **Khan Hajra Mohammed Rashid Seat No. 40150** studying in **Master of Science** in **Information Technology Part II Semester IV** has satisfactorily completed the Practical of **PSIT4P2a Natural Language Processing** 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

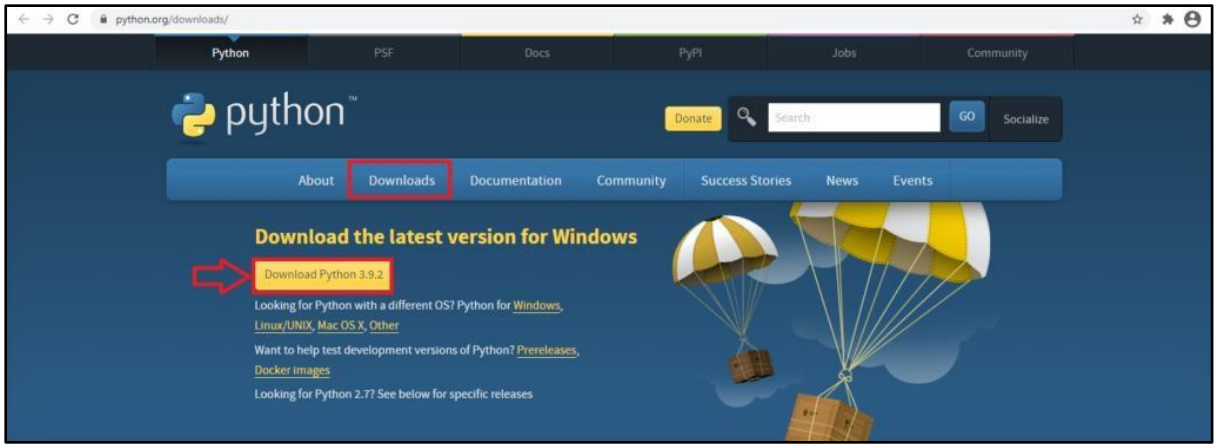
4	07/04/2023	A] Text Tokenization: Tokenization using Python's split() function. B] Tokenization using Regular Expressions (RegEx). C] Tokenization using NLTK. D] Tokenization using the spaCy library. E] Tokenization using Keras. F] Tokenization using Gensim.	39	
5	14/04/2023	A] Illustrate part of speech tagging. B] Named Entity recognition of user defined text. C] Named Entity recognition with diagram using NLTK corpus – treebank POS Tagging, chunking and NER.	45	
6	21/04/2023	A] Finite State Automata: Define grammar using nltk. Analyze a sentence using the same. B] Accept the input string with Regular expression of Finite Automaton: 101+. C] Accept the input string with Regular expression of FA: (a+b)*bba. D] Implementation of Deductive Chart Parsing using context free grammar and a given sentence.	49	
7	28/04/2023	A] Study PorterStemmer. B] Study LancasterStemmer C] Study RegexpStemmer. D] Study SnowballStemmer. E] Study WordNetLemmatizer.	57	
8	05/05/2023	Implement Naive Bayes classifier.	62	
9	12/05/2023	A] Speech Tagging. B] Statistical Parsing	64	
10	19/05/2023	A] Multiword Expressions in NLP. B] Normalized Web Distance and Word Similarity. C] Word Sense Disambiguation.	71	

PRACTICAL 1

[A] AIM: Installation of NLTK.

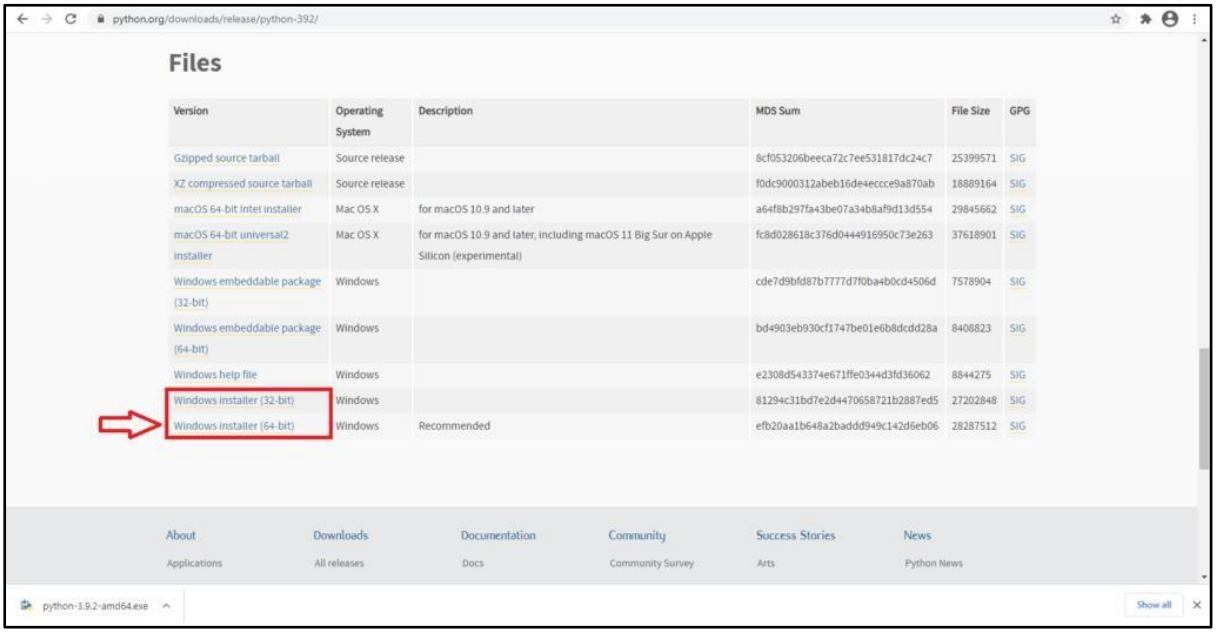
Step 1: Python Installation.

Go to link https://www.python.org/downloads/, and select the latest version for windows.

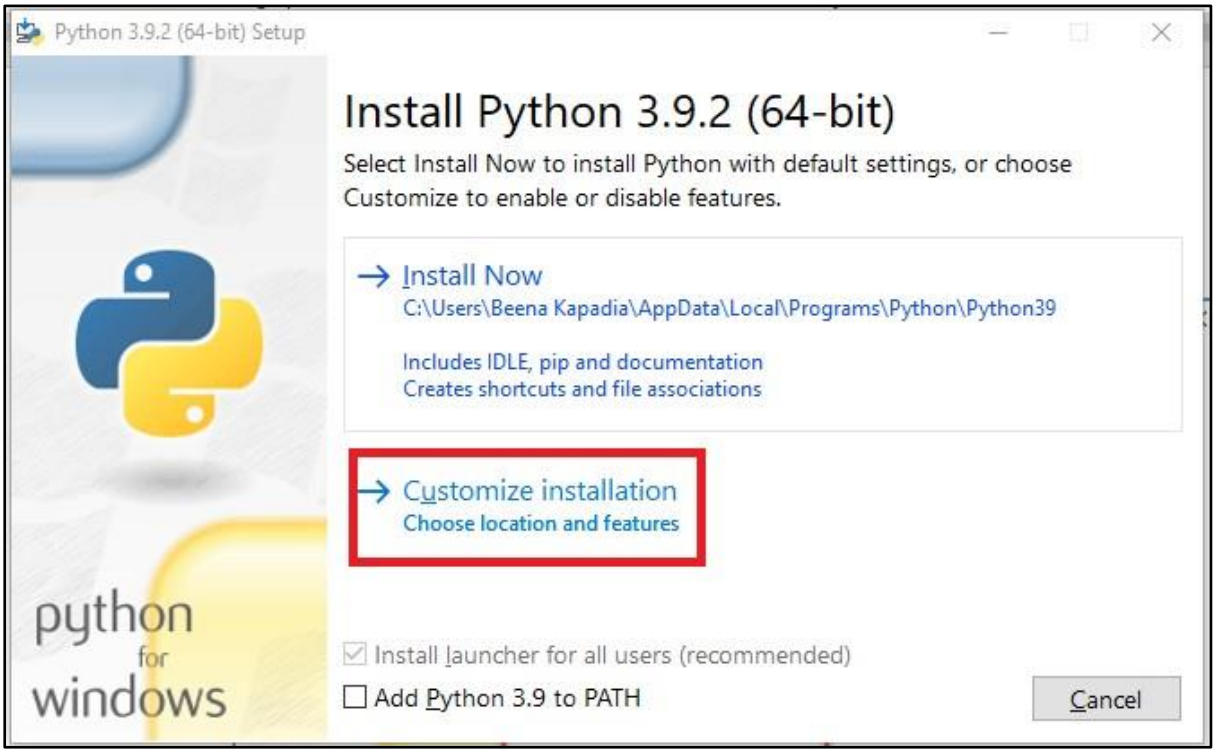


Note: If you don't want to download the latest version, you can visit the download tab and see all releases.

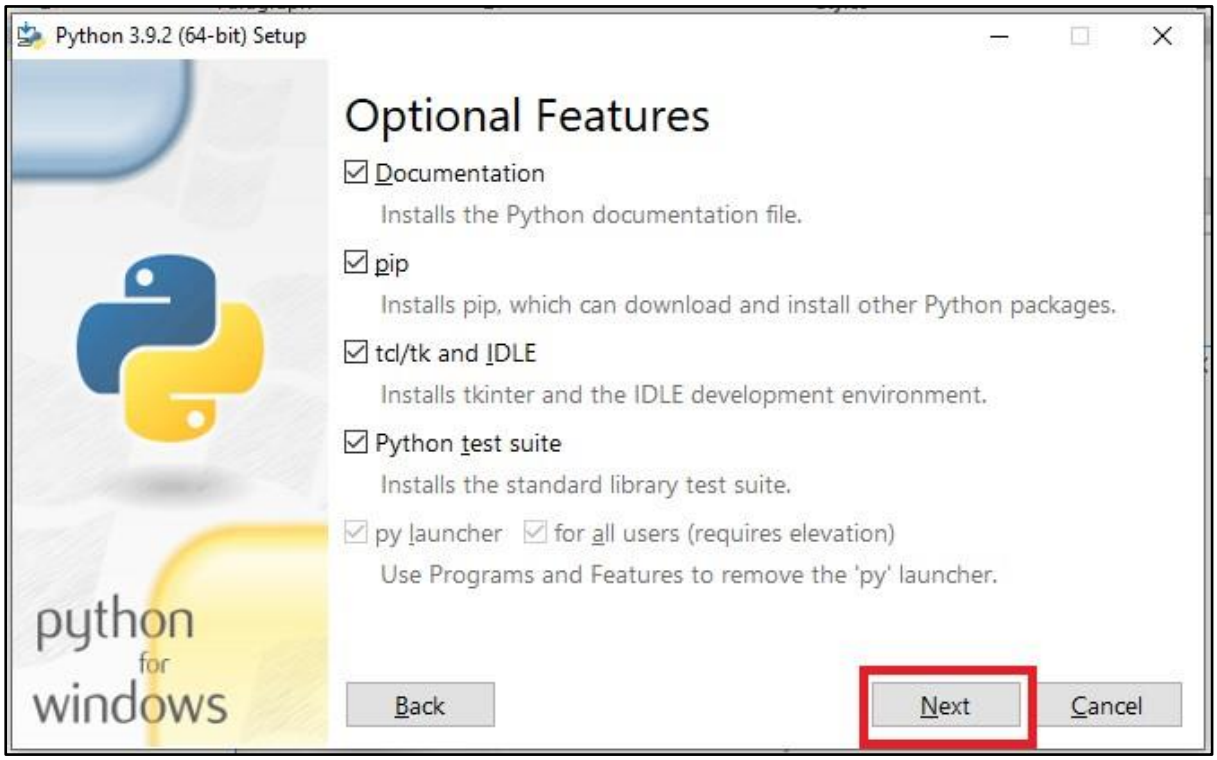
Step 2: Click on the Windows installer (64 bit).



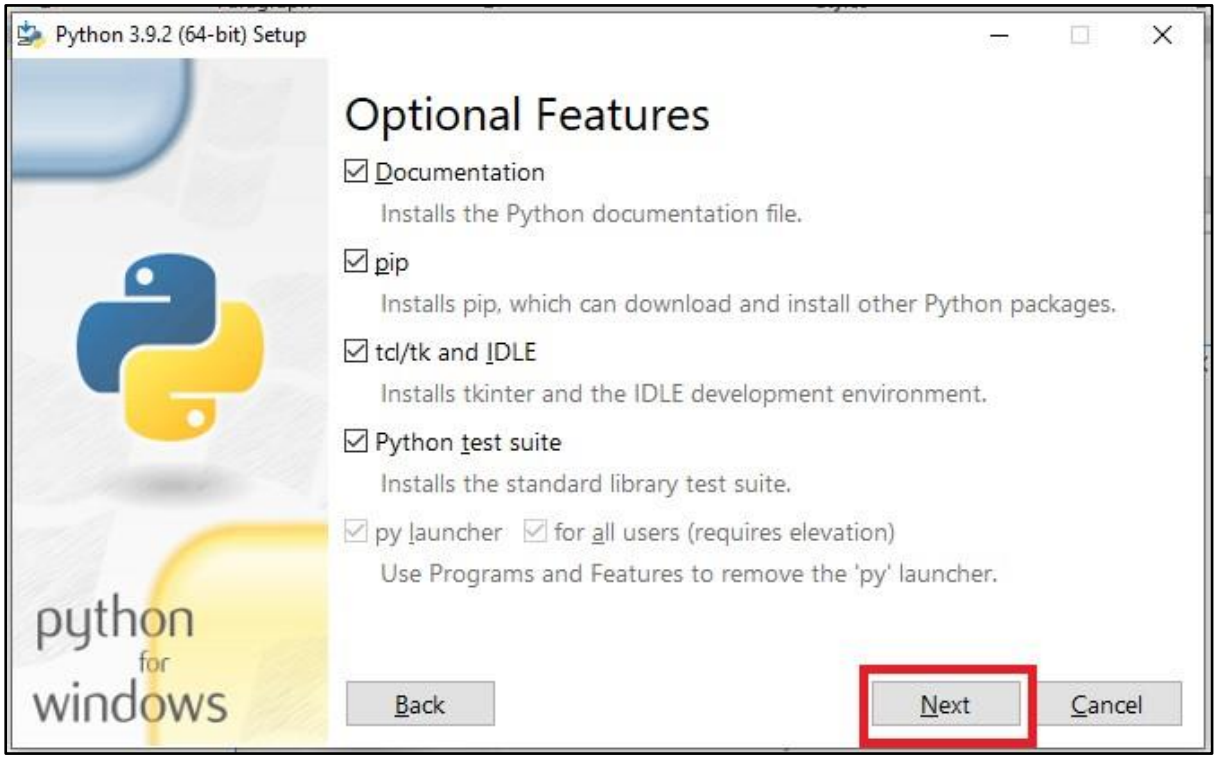
Step 3: Select Customize Installation.



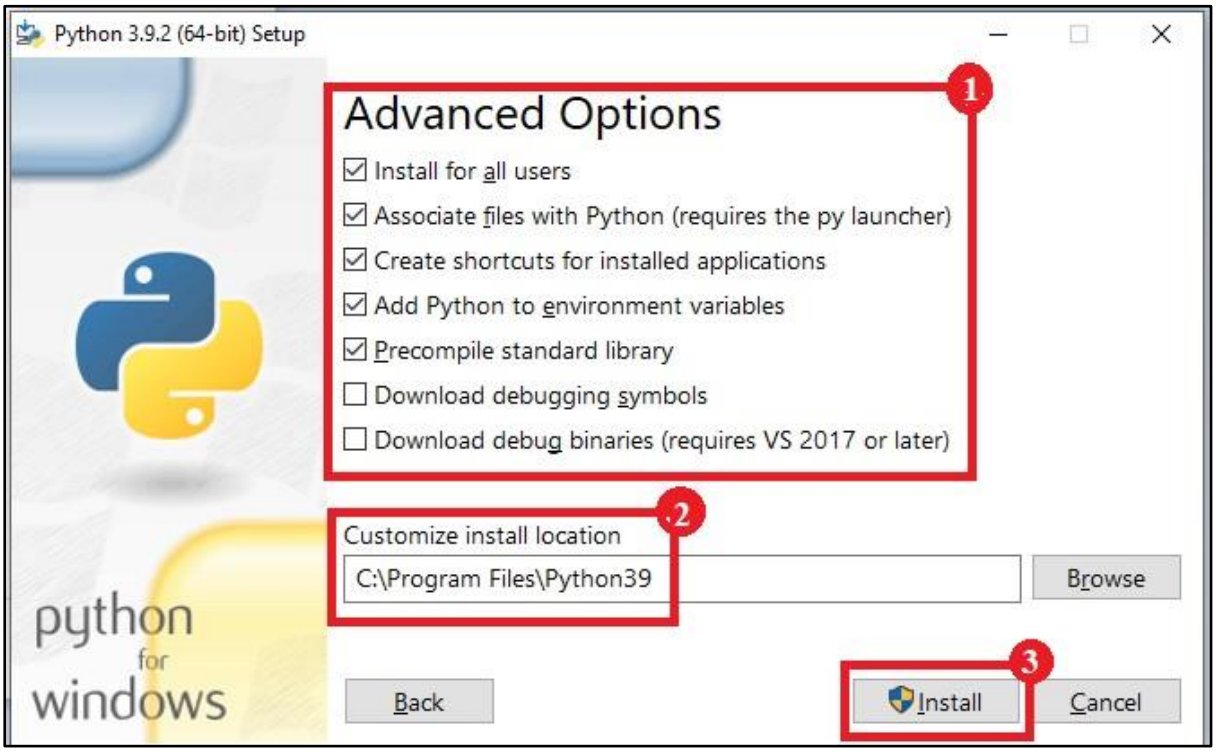
Step 4: Click "Next".



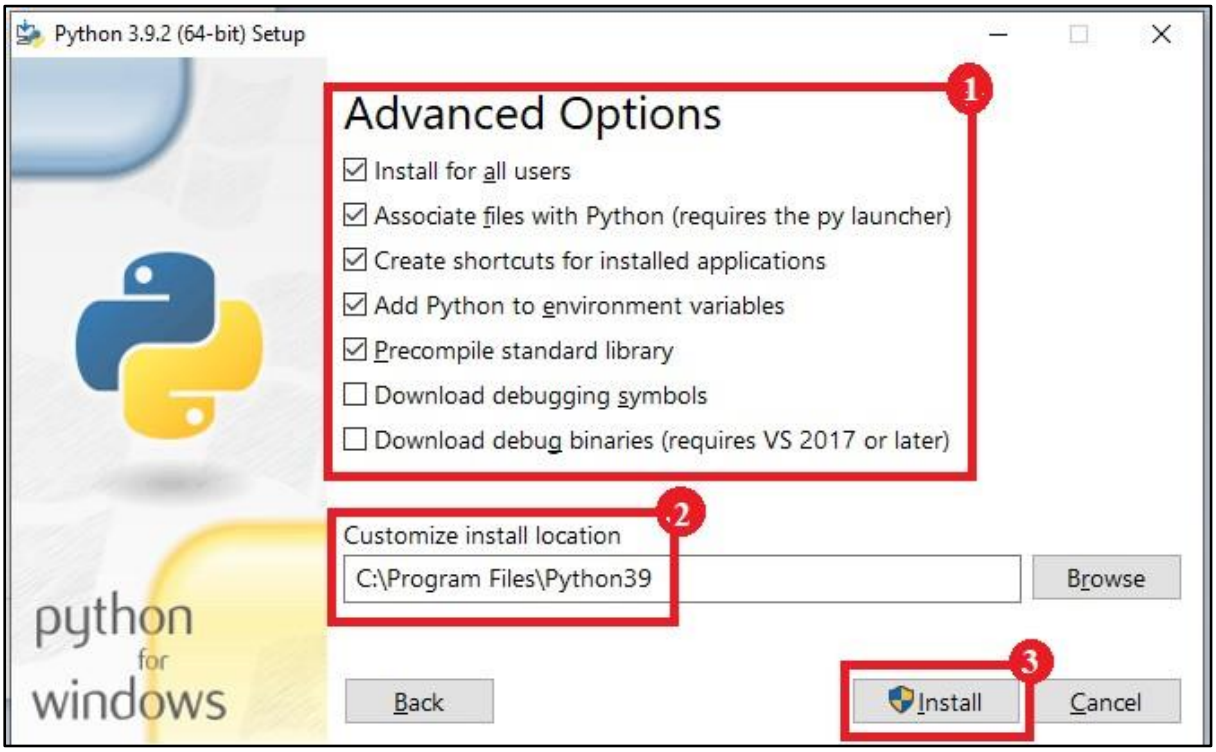
Step 5: Click "Next".



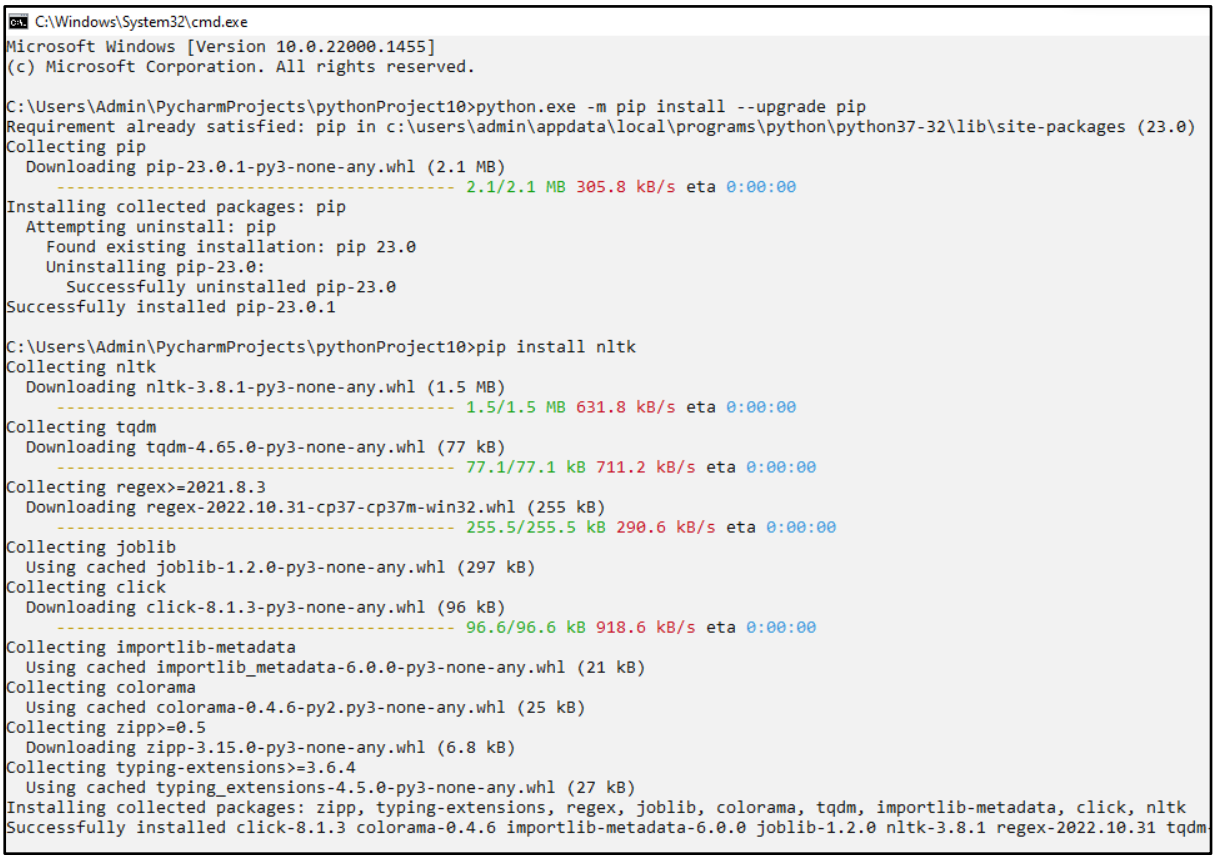
Step 6: In next screen, Select the advanced options, Give a Custom install location. Keep the default folder as c:\Program files\Python39, Click Install to complete Python Installation.



Step 7: In next screen, Select the advanced options, Give a Custom install location. Keep the default folder as c:\Program files\Python39, Click Install to complete Python Installation.



Step 8: Open command prompt window and run the following commands:
python.exe –m pip install --upgrade pip
pip install nltk



Browse https://www.nltk.org/install.html for more details

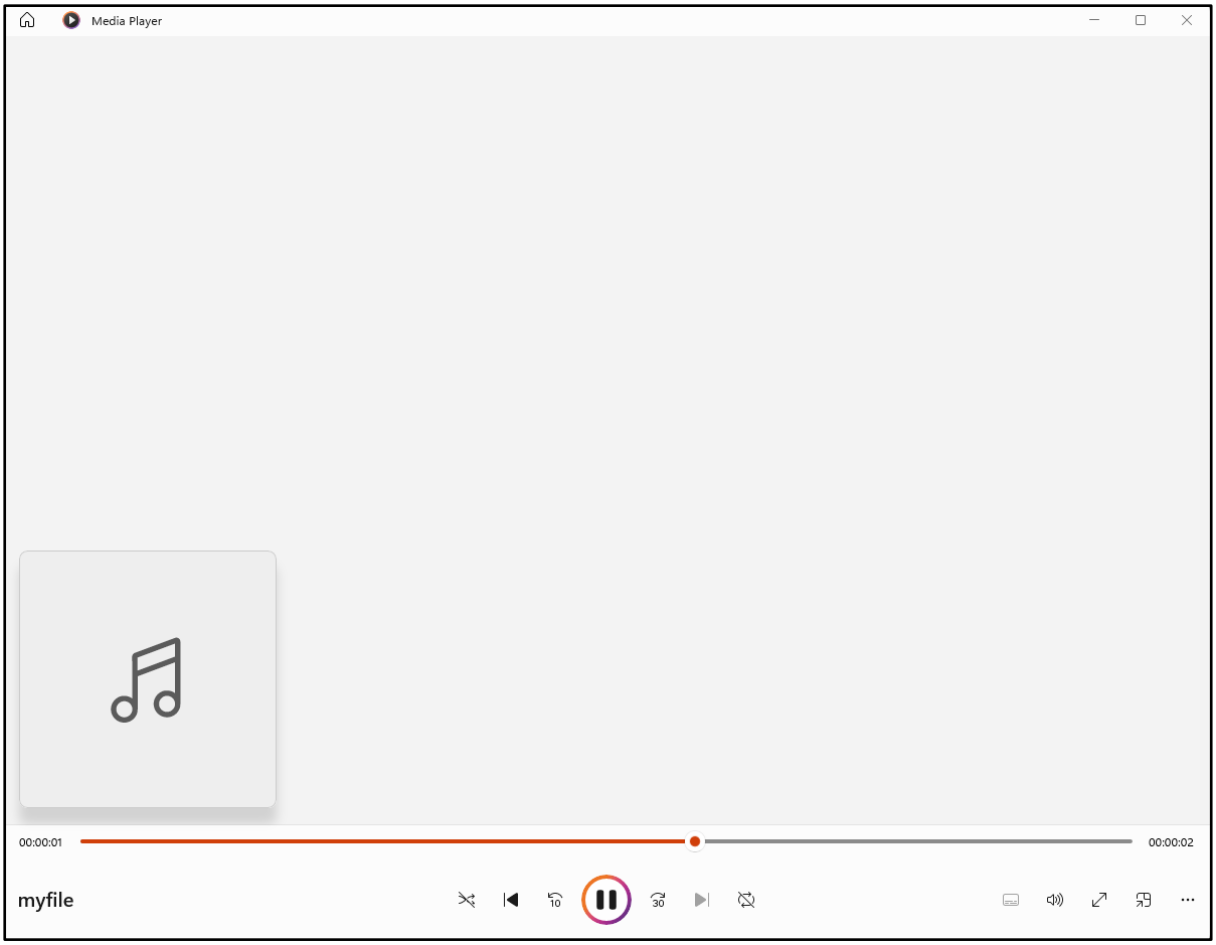
[B] AIM: Convert the given text to speech.

CODE:

```
# text to speech
# pip install gtts
# pip install playsound
# import required for text to speech conversion
from gtts import gTTS
import os
mytext = "Welcome to Natural Language programming"
language = "en"
myobj = gTTS(text=mytext, lang=language, slow=False)
myobj.save("myfile.mp3")
os.system("myfile.mp3")
print("myfile.mp3 has been saved.")
```

OUTPUT:

```
C:\Users\Admin\PycharmProjects\NLP
myfile.mp3 has been saved.

Process finished with exit code 0
```


Media Player
00:00:01
00:00:02
myfile

[C] AIM: Convert the given speech to text.

[I] Audio file to text.

CODE:

```
import speech_recognition as sr
filename = "male.wav"
# initialize the recognizer
r = sr.Recognizer()
# open the file
with sr.AudioFile(filename) as source:
# listen for the data (load audio to memory)
    audio_data = r.record(source)
    # recognize (convert from speech to text)
    text = r.recognize_google(audio_data)
    print(text)
```

OUTPUT:

```
result2:
{   'alternative': [   {   'confidence': 0.87246531,
                           'transcript': 'what is summary decides to break it '
                                         'be careful that you keep coverage '
                                         'but look for places to save money '
                                         "maybe it's taking longer to get "
                                         'things then the bank is expected '
                                         "hiring the life for one's company "
                                         'system house is it'},
                       {   'transcript': 'what is summary decides to break it '
                                         'be careful that you keep coverage '
                                         'but look for places to save money '
                                         "maybe it's taking longer to get "
                                         'things then the bank is expected '
                                         "hiring the life for one's company "
                                         'system houses its'},
                       {   'transcript': 'what is summary decides to break it '
                                         'be careful that you keep coverage '
                                         'but look for places to save money '
                                         "maybe it's taking longer to get "
                                         'things then the bank is expected '
                                         "hiring the life for one's company"},
                       {   'transcript': 'what is summary decides to break it '
                                         'be careful that you keep coverage '
                                         'but look for places to save money '
                                         "maybe it's taking longer to get "
                                         'things then the bank is expected '
                                         "hiring the life for one's company "
                                         'system houses'},
                       {   'transcript': 'what is summary decides to break it '
                                         'be careful that you keep coverage '
                                         'but look for places to save money '
                                         "maybe it's taking longer to get "
                                         'things then the bank is expected '
                                         "hiring the life for one's company "
                                         'system house is'}],
```

```
    'final': True}
what is summary decides to break it be careful that you keep coverage but look for
 places to save money maybe it's taking longer to get things then the bank is expected
 hiring the life for one's company system house is it

Process finished with exit code 0
```

[II] Audio input to text.

CODE:

```
import speech_recognition as sr

# initialize the recognizer
r = sr.Recognizer()

with sr.Microphone() as source:

    print("Listening...")
    r.pause_threshold = 1
    r.adjust_for_ambient_noise(source)

# listen for the data (load audio to memory)
    audio = r.listen(source)
    # recognize (convert from speech to text)
    print("Recognizing")

    request = r.recognize_google(audio, language="en-in")
    print(request)
```

OUTPUT:

```
Listening...
Recognizing
result2:
{   'alternative': [   {   'confidence': 0.88687533,
                           'transcript': 'welcome to natural language '
                                         'processing'}],
    'final': True}
welcome to natural language processing
```

PRACTICAL 2

[A] AIM: Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like filelds, raw, words, sents, categories.

CODE:

```
'''NLTK includes a small selection of texts from the Project brown electronic
text
archive, which contains some 25,000 free electronic books, hosted at
http://www.brown.org/. We begin by getting the Python interpreter to load
the NLTK
package, then ask to see nltk.corpus.brown.fileids(), the file identifiers
in this corpus:'''
import nltk

from nltk.corpus import brown
nltk.download('brown')

print ('File ids of brown corpus\n',brown.fileids())

'''Let's pick out the first of these texts — Emma by Jane Austen — and give
it a short
name, emma, then find out how many words it contains:'''

ca01 = brown.words('ca01')

# display first few words
print('\nca01 has following words:\n',ca01)

# total number of words in ca01
print('\nca01 has',len(ca01),'words')

#categories or files
print ('\n\nCategories or file in brown corpus:\n')

print (brown.categories())
```

```
'''display other information about each text, by looping over all the values
of fileid
corresponding to the brown file identifiers listed earlier and then computing
statistics
for each text.'''

print ('\n\nStatistics for each text:\n')

print
('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFileName')
for fileid in brown.fileids():
 num_chars = len(brown.raw(fileid))
 num_words = len(brown.words(fileid))
 num_sents = len(brown.sents(fileid))
 num_vocab = len(set([w.lower() for w in brown.words(fileid)]))
 print(int(num_chars / num_words), '\t\t\t',
       int(num_words / num_sents), '\t\t\t',
       int(num_words / num_vocab), '\t\t\t', fileid)
```

OUTPUT:

```
File ids of brown corpus
 ['ca01', 'ca02', 'ca03', 'ca04', 'ca05', 'ca06', 'ca07', 'ca08', 'ca09', 'ca10', 'ca11', 'ca12', 'ca13', 'ca14', 'ca15', 'ca16', 'ca17', 'ca18', 'ca19', 'ca20', 'ca21', 'ca22', 'ca23',

ca01 has following words:
 ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

ca01 has 2242 words


Categories or file in brown corpus:

['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']


Statistics for each text:

9           22          2           ca01
8           23          2           ca02
8           20          2           ca03
9           25          2           ca04
8           26          3           ca05
8           22          2           ca06
9           18          2           ca07
8           21          2           ca08
9           19          2           ca09
8           21          2           ca10
8           22          2           ca11
8           22          2           ca12
8           20          2           ca13
```

[B] AIM: Create and use your own corpora (plaintext, categorical).

CODE:

```
'''NLTK includes a small selection of texts from the Project filelist
electronic text
archive, which contains some 25,000 free electronic books, hosted at
http://www.filelist.org/. We begin by getting the Python interpreter to load
the NLTK
package, then ask to see nltk.corpus.filelist.fileids(), the file identifiers
in this corpus:'''
import nltk
nltk.download('punkt')
from nltk.corpus import PlaintextCorpusReader
corpus_root = 'test'
filelist = PlaintextCorpusReader(corpus_root, '.*')
print ('\n File list: \n')
print (filelist.fileids())
print (filelist.root)

'''display other information about each text, by looping over all the values
of fileid
corresponding to the filelist file identifiers listed earlier and then
computing statistics
for each text.'''
print ('\n\nStatistics for each text:\n')
print
('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\tFileName')
for fileid in filelist.fileids():
 num_chars = len(filelist.raw(fileid))
 num_words = len(filelist.words(fileid))
 num_sents = len(filelist.sents(fileid))
 num_vocab = len(set([w.lower() for w in filelist.words(fileid)]))
 print                                    (int(num_chars/num_words),'\t\t\t',
int(num_words/num_sents),'\t\t\t',
int(num_words/num_vocab),'\t\t', fileid)
```

OUTPUT:

```
 File list:

['test.txt']
D:\sample


Statistics for each text:

4            25              2       test.txt
```

[C] AIM: Study Conditional frequency distributions.

CODE:

```
#process a sequence of pairs
text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
pairs = [('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ...]
import nltk
nltk.download('brown')
nltk.download('inaugural')
nltk.download('udhr')
from nltk.corpus import brown
fd = nltk.ConditionalFreqDist(
 (genre, word)
 for genre in brown.categories()
 for word in brown.words(categories=genre))
genre_word = [(genre, word)
 for genre in ['news', 'romance']
 for word in brown.words(categories=genre)]
print(len(genre_word))
print(genre_word[:4])
print(genre_word[-4:])
cfd = nltk.ConditionalFreqDist(genre_word)
print(cfd)
print(cfd.conditions())
print(cfd['news'])
print(cfd['romance'])
print(list(cfd['romance']))
from nltk.corpus import inaugural
cfd = nltk.ConditionalFreqDist(
 (target, fileid[:4])
 for fileid in inaugural.fileids()
 for w in inaugural.words(fileid)
 for target in ['america', 'citizen']
 if w.lower().startswith(target))
from nltk.corpus import udhr
languages = ['Chickasaw', 'English', 'German_Deutsch',
 'Greenlandic_Inuktikut', 'Hungarian_Magyar', 'Ibibio_Efik']
cfd = nltk.ConditionalFreqDist(
```

```
 (lang, len(word))
 for lang in languages
 for word in udhr.words(lang + '-Latin1'))
cfd.tabulate(conditions=['English', 'German_Deutsch'],
 samples=range(10), cumulative=True)
```

OUTPUT:

```
170576
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]
[('romance', 'afraid'), ('romance', 'not'), ('romance', "''"), ('romance', '.')]
<ConditionalFreqDist with 2 conditions>
['news', 'romance']
<FreqDist with 14394 samples and 100554 outcomes>
<FreqDist with 8452 samples and 70022 outcomes>
[',', '.', 'the', 'and', 'to', 'a', 'of', '``', "''", 'was', 'I', 'in', 'he', 'had', '?', 'her', 'that', 'it', 'his', 'she', 'with', 'you',
                  0    1    2    3    4    5    6    7    8    9
       English    0  185  525  883  997 1166 1283 1440 1558 1638
German_Deutsch    0  171  263  614  717  894 1013 1110 1213 1275
```

[D] AIM: Study of tagged corpora with methods like tagged_sents, tagged_words.

CODE:

```
import nltk
from nltk import tokenize
nltk.download('punkt')
nltk.download('words')
para = "Hello! My name is UDIT. Today you'll be learning NLTK."
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n==================\n",sents)
# word tokenization
print("\nword tokenization\n==================\n")
for index in range(len(sents)):
 words = tokenize.word_tokenize(sents[index])
 print(words)
```

OUTPUT:

```
sentence tokenization
====================
 ['Hello!', 'My name is UDIT.', "Today you'll be learning NLTK."]

word tokenization
====================

['Hello', '!']
['My', 'name', 'is', 'UDIT', '.']
['Today', 'you', "'ll", 'be', 'learning', 'NLTK', '.']
```

[E] AIM: Write a program to find the most frequent noun tags.

CODE:

```
import nltk
nltk.download('averaged_perceptron_tagger')
from collections import defaultdict
text = nltk.word_tokenize("Nick likes to play football. Nick does not like
to play cricket.")
tagged = nltk.pos_tag(text)
print(tagged)
# checking if it is a noun or not
addNounWords = []
count=0
for words in tagged:
 val = tagged[count][1]
 if(val == 'NN' or val == 'NNS' or val == 'NNPS' or val == 'NNP'):
     addNounWords.append(tagged[count][0])
     count+=1
print (addNounWords)
temp = defaultdict(int)
# memoizing count
for sub in addNounWords:
 for wrd in sub.split():
    temp[wrd] += 1
# getting max frequency
res = max(temp, key=temp.get)
# printing result
print("Word with maximum frequency : " + str(res))
```

OUTPUT:

```
[('Nick', 'NNP'), ('likes', 'VBZ'), ('to', 'TO'), ('play', 'VB'),
 ('football', 'NN'), ('.', '.'), ('Nick', 'NNP'), ('does', 'VBZ'), ('not',
 'RB'), ('like', 'VB'), ('to', 'TO'), ('play', 'VB'), ('cricket', 'NN'),
 ('.', '.')]
['Nick', 'football', 'Nick', 'cricket']
Word with maximum frequency : Nick
```

[F] AIM: Map Words to Properties Using Python Dictionaries.

CODE:

```
#creating and printing a dictionay by mapping word with its properties
thisdict = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```

OUTPUT:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Ford
3
<class 'dict'>
```

[G] AIM: Study i) DefaultTagger, ii) Regular expression tagger, iii) UnigramTagger.

I) Default Tagger

CODE:

```
import nltk
from nltk.tag import DefaultTagger

exptagger = DefaultTagger('NN')
nltk.download('treebank')
from nltk.corpus import treebank

testsentences = treebank.tagged_sents()[1000:]
print(exptagger.accuracy(testsentences))
# Tagging a list of sentences
import nltk
from nltk.tag import DefaultTagger

exptagger = DefaultTagger('NN')
print(exptagger.tag_sents([['Hi', ','], ['How', 'are', 'you', '?']]))
```

OUTPUT:

```
0.13198749536374715
[[('Hi', 'NN'), (',', 'NN')], [('How', 'NN'), ('are', 'NN'), ('you', 'NN'), ('?', 'NN')]]
```

II) Regular Expression Tagger

CODE:

```
from nltk.corpus import brown
from nltk.tag import RegexpTagger
test_sent = brown.sents(categories='news')[0]
regexp_tagger = RegexpTagger(
 [(r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers
 (r'(The|the|A|a|An|an)$', 'AT'), # articles
 (r'.*able$', 'JJ'), # adjectives
 (r'.*ness$', 'NN'), # nouns formed from adjectives
 (r'.*ly$', 'RB'), # adverbs
 (r'.*s$', 'NNS'), # plural nouns
 (r'.*ing$', 'VBG'), # gerunds
 (r'.*ed$', 'VBD'), # past tense verbs
 (r'.*', 'NN') # nouns (default)
])
print(regexp_tagger)
print(regexp_tagger.tag(test_sent))
```

OUTPUT:

```
<Regexp Tagger: size=9>
[('The', 'AT'), ('Fulton', 'NN'), ('County', 'NN'), ('Grand', 'NN'),
 ('Jury', 'NN'), ('said', 'NN'), ('Friday', 'NN'), ('an', 'AT'),
 ('investigation', 'NN'), ('of', 'NN'), ("Atlanta's", 'NNS'), ('recent',
 'NN'), ('primary', 'NN'), ('election', 'NN'), ('produced', 'VBD'), ('``',
 'NN'), ('no', 'NN'), ('evidence', 'NN'), ("''", 'NN'), ('that', 'NN'),
 ('any', 'NN'), ('irregularities', 'NNS'), ('took', 'NN'), ('place', 'NN'),
 ('.', 'NN')]
```

III) Unigram Tagger

CODE:

```
# Loading Libraries
from nltk.tag import UnigramTagger
from nltk.corpus import treebank
# Training using first 10 tagged sentences of the treebank corpus as data.
# Using data
train_sents = treebank.tagged_sents()[:10]

# Initializing
tagger = UnigramTagger(train_sents)

# Lets see the first sentence
# (of the treebank corpus) as list
print(treebank.sents()[0])
print('\n',tagger.tag(treebank.sents()[0]))
#Finding the tagged results after training.
tagger.tag(treebank.sents()[0])
#Overriding the context model
tagger = UnigramTagger(model ={'Pierre': 'NN'})
print('\n',tagger.tag(treebank.sents()[0]))
```

OUTPUT:

```
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the',
 'board', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']

 [('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years',
  'NNS'), ('old', 'JJ'), (',', ','), ('will', 'MD'), ('join', 'VB'), ('the',
   'DT'), ('board', 'NN'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive',
  'JJ'), ('director', 'NN'), ('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')]

 [('Pierre', 'NN'), ('Vinken', None), (',', None), ('61', None), ('years',
  None), ('old', None), (',', None), ('will', None), ('join', None), ('the',
   None), ('board', None), ('as', None), ('a', None), ('nonexecutive',
  None), ('director', None), ('Nov.', None), ('29', None), ('.', None)]
```

[H] AIM: Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find the score of words.

CODE:

```
from __future__ import with_statement  # with statement for reading file
import re  # Regular expression

words = []  # corpus file words
testword = []  # test words
ans = []  # words matches with corpus
print("MENU")
print("-----------")
print(" 1 . Hash tag segmentation ")
print(" 2 . URL segmentation ")
print("enter the input choice for performing word segmentation")
choice = int(input())
if choice == 1:
    text = "#whatismyname"  # hash tag test data to segment
    print("input with HashTag", text)
    pattern = re.compile("[^\w']")
    a = pattern.sub('', text)
elif choice == 2:
    text = "www.whatismyname.com"  # url test data to segment
    print("input with URL", text)
    a = re.split('\s|(?<!\d)[,.](?!\d)', text)
    splitwords = ["www", "com", "in"]  # remove the words which is containg
in the list
    a = "".join([each for each in a if each not in splitwords])

else:
    print("wrong choice...try again")

print(a)
for each in a:
    testword.append(each)  # test word
test_lenth = len(testword)  # lenth of the test data

# Reading the corpus
```

```
with open('words.txt', 'r') as f:
    lines = f.readlines()
    words = [(e.strip()) for e in lines]


def Seg(a, lenth):
    ans = []
    for k in range(0, lenth + 1):  # this loop checks char by char in the
corpus

        if a[0:k] in words:
            print(a[0:k], "-appears in the corpus")
            ans.append(a[0:k])
            break
    if ans != []:
        g = max(ans, key=len)
        return g


test_tot_itr = 0  # each iteration value
answer = []  # Store the each word contains the corpus
Score = 0  # initial value for score
N = 37  # total no of corpus
M = 0
C = 0
while test_tot_itr < test_lenth:
    ans_words = Seg(a, test_lenth)
    if ans_words != 0:
        test_itr = len(ans_words)
        answer.append(ans_words)
        a = a[test_itr:test_lenth]
        test_tot_itr += test_itr

Aft_Seg = " ".join([each for each in answer])
# print segmented words in the list
print("output")
print("---------")
print(Aft_Seg)  # print After segmentation the input
# Calculating Score
```

```
C = len(answer)
score = C * N / N  # Calculate the score
print("Score", score)
```

OUTPUT:

```
MENU
-----------
 1 . Hash tag segmentation
 2 . URL segmentation
enter the input choice for performing word segmentation
1
input with HashTag #whatismyname
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
---------
what is my name
Score 4.0
```

```
MENU
-----------
 1 . Hash tag segmentation
 2 . URL segmentation
enter the input choice for performing word segmentation
2
input with URL www.whatismyname.com
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
---------
what is my name
Score 4.0
```

PRACTICAL 3

[A] AIM: Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms.

CODE:

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
# definition and example of the word 'computer'
print(wordnet.synset("computer.n.01").definition())
#examples
print("Examples:", wordnet.synset("computer.n.01").examples())
#get Antonyms
print(wordnet.lemma('buy.v.01.buy').antonyms())
```

OUTPUT:

```
[Synset('computer.n.01'), Synset('calculator.n.01')]
a machine for performing calculations automatically
Examples: []
[Lemma('sell.v.01.sell')]
```

[B] AIM: Study lemmas, hyponyms, hypernyms.

CODE:

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
print(wordnet.synset("computer.n.01").lemma_names())
#all lemmas for each synset.
for e in wordnet.synsets("computer"):
 print(f'{e} --> {e.lemma_names()}')
#print all lemmas for a given synset
print(wordnet.synset('computer.n.01').lemmas())
#get the synset corresponding to lemma
print(wordnet.lemma('computer.n.01.computing_device').synset())
#Get the name of the lemma
print(wordnet.lemma('computer.n.01.computing_device').name())
#Hyponyms give abstract concepts of the word that are much more specific
#the list of hyponyms words of the computer
syn = wordnet.synset('computer.n.01')
print(syn.hyponyms)
print([lemma.name()   for   synset   in   syn.hyponyms()   for   lemma   in
synset.lemmas()])
#the semantic similarity in WordNet
vehicle = wordnet.synset('vehicle.n.01')
car = wordnet.synset('car.n.01')
print(car.lowest_common_hypernyms(vehicle))
```

OUTPUT:

```
[Synset('computer.n.01'), Synset('calculator.n.01')]
['computer', 'computing_machine', 'computing_device', 'data_processor', 'electronic_computer',
 'information_processing_system']
Synset('computer.n.01') --> ['computer', 'computing_machine', 'computing_device',
 'data_processor', 'electronic_computer', 'information_processing_system']
Synset('calculator.n.01') --> ['calculator', 'reckoner', 'figurer', 'estimator', 'computer']
[Lemma('computer.n.01.computer'), Lemma('computer.n.01.computing_machine'), Lemma('computer.n
 .01.computing_device'), Lemma('computer.n.01.data_processor'), Lemma('computer.n.01
 .electronic_computer'), Lemma('computer.n.01.information_processing_system')]
Synset('computer.n.01')
computing_device
<bound method _WordNetObject.hyponyms of Synset('computer.n.01')>
['analog_computer', 'analogue_computer', 'digital_computer', 'home_computer', 'node', 'client',
  'guest', 'number_cruncher', 'pari-mutuel_machine', 'totalizer', 'totaliser', 'totalizator',
 'totalisator', 'predictor', 'server', 'host', 'Turing_machine', 'web_site', 'website',
 'internet_site', 'site']
[Synset('vehicle.n.01')]
```

[C] AIM: Write a program using python to find synonym and antonym of word "active" using Wordnet.

CODE:

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
print( wordnet.synsets("active"))
print(wordnet.lemma('active.a.01.active').antonyms())
```

OUTPUT:

```
[Synset('active_agent.n.01'), Synset('active_voice.n.01'), Synset('active.n.03'), Synset
 ('active.a.01'), Synset('active.s.02'), Synset('active.a.03'), Synset('active.s.04'),
 Synset('active.a.05'), Synset('active.a.06'), Synset('active.a.07'), Synset('active.s.08'),
 Synset('active.a.09'), Synset('active.a.10'), Synset('active.a.11'), Synset('active.a.12'),
 Synset('active.a.13'), Synset('active.a.14')]
[Lemma('inactive.a.02.inactive')]
```

[D] AIM: Compare two nouns.

CODE:

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
syn1 = wordnet.synsets('football')
syn2 = wordnet.synsets('soccer')
# A word may have multiple synsets, so need to compare each synset of word1
with synset of word2
for s1 in syn1:
 for s2 in syn2:
     print("Path similarity of: ")
     print(s1, '(', s1.pos(), ')', '[', s1.definition(), ']')
     print(s2, '(', s2.pos(), ')', '[', s2.definition(), ']')
     print(" is", s1.path_similarity(s2))
     print()
```

OUTPUT:

```
Path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with a ball (round or oval) in
 which two teams try to kick or carry or propel the ball into each other's goal ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players try to kick or
 head a ball into the opponents' goal ]
 is 0.5

Path similarity of:
Synset('football.n.02') ( n ) [ the inflated oblong ball used in playing American football ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players try to kick or
 head a ball into the opponents' goal ]
 is 0.05
```

[E] AIM: Handling stopwords:

I) Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List.

CODE:

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in
stopwords.words()]
print(tokens_without_sw)
#add the word play to the NLTK stop word collection
all_stopwords = stopwords.words('english')
all_stopwords.append('play')
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in
all_stopwords]
print(tokens_without_sw)
#remove ‘not’ from stop word collection
all_stopwords.remove('not')
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in
all_stopwords]
print(tokens_without_sw)
```

OUTPUT:

```
['Yashesh', 'likes', 'play', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'however', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'however', 'not', 'fond', 'tennis', '.']
```

II) Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List.

CODE:

```
import gensim
from nltk.tokenize import word_tokenize
from gensim.parsing.preprocessing import remove_stopwords
text = "Yashesh likes to play football, however he is not too fond of tennis."
filtered_sentence = remove_stopwords(text)
print(filtered_sentence)
all_stopwords = gensim.parsing.preprocessing.STOPWORDS
print(all_stopwords)
'''The following script adds likes and play to the list of stop words in
Gensim:'''
from gensim.parsing.preprocessing import STOPWORDS
all_stopwords_gensim = STOPWORDS.union(set(['likes', 'play']))
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in
all_stopwords_gensim]
print(tokens_without_sw)
'''Output:
['Yashesh', 'football', ',', 'fond', 'tennis', '.']
The following script removes the word "not" from the set of stop words in
Gensim:'''
from gensim.parsing.preprocessing import STOPWORDS
all_stopwords_gensim = STOPWORDS
sw_list = {"not"}
all_stopwords_gensim = STOPWORDS.difference(sw_list)
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in
all_stopwords_gensim]
print(tokens_without_sw)
```

OUTPUT:

```
Yashesh likes play football, fond tennis.
frozenset({'yourself', 'get', 'he', 'there', 'we', 'serious', 'de', 'but', 'neither', 'its',
 'after', 'among', 'did', 'part', 'itself', 'when', 'perhaps', 'less', 'again', 'co', 'none',
 'whenever', 'his', 'is', 'someone', 'yourselves', 'around', 'with', 'therein', 'while', 'one',
  'anywhere', 'up', 'various', 'via', 'rather', 'hereby', 'else', 'fire', 'the', 'alone', 'an',
  'former', 'wherein', 'us', 'due', 'many', 'everything', 'which', 'why', 'meanwhile',
 'cannot', 'first', 'since', 'fifty', 'hundred', 'anyone', 'hasnt', 'somewhere', 'about',
 'already', 'mostly', 'therefore', 'under', 'becoming', 'take', 'might', 'everywhere', 'were',
 'himself', 'twelve', 'him', 'am', 'using', 'don', 'themselves', 'her', 'would', 'herself',
 'eleven', 'full', 'becomes', 'hereafter', 'nine', 'thick', 'seems', 'last', 'once', 'was',
 'couldnt', 'for', 'nowhere', 'every', 'whither', 'not', 'had', 'it', 'will', 'sixty', 'mine',
 'become', 'fifteen', 'against', 'without', 'almost', 'five', 'make', 'above', 'side', 'off',
 'me', 'go', 'indeed', 'those', 'name', 'does', 'least', 'my', 'own', 'etc', 'whom', 'well',
 'ie', 'myself', 'they', 'very', 'six', 'un', 'now', 'whether', 'whole', 'thereupon', 'next',
 'front', 'besides', 'really', 'sometimes', 'top', 'though', 'are', 'back', 'give', 'see',
 'each', 'may', 'and', 'sincere', 'during', 'yet', 'onto', 'three', 'mill', 'thru', 'so',
 'until', 'wherever', 'namely', 'otherwise', 'ourselves', 'down', 'con', 'amoungst', 'whence',
 'something', 'on', 'out', 'hereupon', 'cant', 'them', 'then', 'their', 'anything', 'thereby',
 'here', 'never', 'anyhow', 'done', 'should', 'ever', 'than', 'your', 'a', 'more', 'latter',
 'unless', 'what', 'be', 'have', 'however', 'i', 'enough', 'do', 'became', 'km', 'ours', 'all',
```

```
  'this', 're', 'yours', 'herein', 'just', 'along', 'such', 'used', 'whatever', 'or', 'both',
 'empty', 'where', 'hers', 'within', 'being', 'to', 'thus', 'two', 'describe', 'regarding',
 'between', 'eight', 'keep', 'another', 'she', 'whereby', 'other', 'please', 'these',
 'nevertheless', 'whereupon', 'can', 'bottom', 'made', 'eg', 'into', 'too', 'nobody',
 'because', 'behind', 'who', 'how', 'you', 'below', 'towards', 'formerly', 'per', 'found',
 'thereafter', 'must', 'computer', 'fill', 'somehow', 'whoever', 'been', 'beside', 'third',
 'upon', 'kg', 'seemed', 'interest', 'forty', 'beyond', 'moreover', 'same', 'as', 'before',
 'amongst', 'further', 'everyone', 'across', 'afterwards', 'cry', 'bill', 'most', 'whereas',
 'whereafter', 'seem', 'elsewhere', 'seeming', 'that', 'quite', 'has', 'whose', 'our', 'at',
 'could', 'show', 'some', 'of', 'over', 'even', 'detail', 'always', 'any', 'say', 'nothing',
 'thence', 'doing', 'few', 'much', 'although', 'hence', 'several', 'still', 'nor', 'together',
 'throughout', 'doesn', 'twenty', 'anyway', 'also', 'by', 'ten', 'through', 'ltd', 'inc',
 'sometime', 'in', 'only', 'latterly', 'system', 'thin', 'move', 'didn', 'put', 'if', 'toward',
  'noone', 'from', 'often', 'four', 'amount', 'except', 'find', 'beforehand', 'call', 'either',
  'others', 'no'})
['Yashesh', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'play', 'football', ',', 'not', 'fond', 'tennis', '.']
```

III) Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List.

CODE:

```
import spacy
import nltk
from nltk.tokenize import word_tokenize
spacy.cli.download("en_core_web_sm")
sp = spacy.load('en_core_web_sm')
#add the word play to the NLTK stop word collection
all_stopwords = sp.Defaults.stop_words
all_stopwords.add("play")
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
#remove 'not' from stop word collection
all_stopwords.remove('not')
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
```

OUTPUT:

```
You can now load the package via spacy.load('en_core_web_sm')
['Yashesh', 'likes', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'not', 'fond', 'tennis', '.']
```

PRACTICAL 4

[A] AIM: Text Tokenization: Tokenization using Python's split() function.

CODE:

```
text = """ This tool is an a beta stage. Alexa developers can use Get Metrics API to
seamlessly analyse metric. It also supports custom skill model, prebuilt Flash Briefing
model, and the Smart Home Skill API. You can use this tool for creation of monitors,
alarms, and dashboards that spotlight changes. The release of these three tools will
enable developers to create visual rich skills for Alexa devices with screens. Amazon
describes these tools as the collection of tech and tools for creating visually rich and
interactive voice experiences. """
data = text.split('.')
for i in data:
    print (i)
```

OUTPUT:

```
 This tool is an a beta stage
 Alexa developers can use Get Metrics API to
seamlessly analyse metric
 It also supports custom skill model, prebuilt Flash Briefing
model, and the Smart Home Skill API
 You can use this tool for creation of monitors,
alarms, and dashboards that spotlight changes
 The release of these three tools will
enable developers to create visual rich skills for Alexa devices with screens
 Amazon
describes these tools as the collection of tech and tools for creating visually rich and
interactive voice experiences
```

[B] AIM: Tokenization using Regular Expressions (RegEx).

CODE:

```
import nltk
# import RegexpTokenizer() method from nltk
from nltk.tokenize import RegexpTokenizer
# Create a reference variable for Class RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps = True)
# Create a string input
str = "I love to study Natural Language Processing in Python"
# Use tokenize method
tokens = tk.tokenize(str)
print(tokens)
```

OUTPUT:

```
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
```

[C] AIM: Tokenization using NLTK.

CODE:

```
import nltk
from nltk.tokenize import word_tokenize
# Create a string input
str = "I love to study Natural Language Processing in Python"
# Use tokenize method
print(word_tokenize(str))
```

OUTPUT:

```
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
```

[D] AIM: Tokenization using the spaCy library.

CODE:

```
import spacy
nlp = spacy.blank("en")
# Create a string input
str = "I love to study Natural Language Processing in Python"
# Create an instance of document;
# doc object is a container for a sequence of Token objects.
doc = nlp(str)
# Read the words; Print the words
words = [word.text for word in doc]
print(words)
```

OUTPUT:

```
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
```

[E] AIM: Tokenization using Keras.

CODE:

```
#pip install keras
#pip install tensorflow
import keras
from keras.preprocessing.text import text_to_word_sequence
# Create a string input
str = "I love to study Natural Language Processing in Python"
# tokenizing the text
tokens = text_to_word_sequence(str)
print(tokens)
```

OUTPUT:

```
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
```

[F] AIM: Tokenization using Gensim.

CODE:

```
#pip install gensim
from gensim.utils import tokenize
# Create a string input
str = "I love to study Natural Language Processing in Python"
# tokenizing the text
print(list(tokenize(str)))
```

OUTPUT:

```
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
```

PRACTICAL 5

AIM: Illustrate part of speech tagging.

[A] Part of speech Tagging and chunking of user defined text.

[i] sentence tokenization

[ii]word tokenization

[iii]Part of speech Tagging and chunking of user defined text.

CODE:

```
import nltk
from nltk import tokenize
nltk.download('punkt')
from nltk import tag
from nltk import chunk
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
para = "Hello! My name is UDIT. Today you'll be learning NLTK."
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n===================\n",sents)
# word tokenization
print("\nword tokenization\n===================\n")
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)

# POS Tagging
tagged_words = []
for index in range(len(sents)):
    tagged_words.append(tag.pos_tag(words))
print("\nPOS Tagging\n===========\n",tagged_words)

# chunking
tree = []
```

```
for index in range(len(sents)):
    tree.append(chunk.ne_chunk(tagged_words[index]))
print("\nchunking\n========\n")
print(tree)
```

OUTPUT:

```
sentence tokenization
====================
 ['Hello!', 'My name is UDIT.', "Today you'll be learning NLTK."]

word tokenization
===================

['Hello', '!']
['My', 'name', 'is', 'UDIT', '.']
['Today', 'you', "'ll", 'be', 'learning', 'NLTK', '.']
```

```
POS Tagging
===========
 [[('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('you',
  'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'),
  ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')]]

chunking
========

[Tree('S', [('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.',
 '.')]), Tree('S', [('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]),
 ('.', '.')]), Tree('S', [('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK',
 'NNP')]), ('.', '.')])]
```

[B] AIM: Named Entity recognition of user defined text.

CODE:

```
import spacy
# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")
# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
"Google in 2007, few people outside of the company took him "
"seriously. “I can tell you very senior CEOs of major American "
"car companies would shake my hand and turn away because I wasn’t "
"worth talking to,” said Thrun, in an interview with Recode earlier "
"this week.")
doc = nlp(text)
# Analyse syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])
```

OUTPUT:

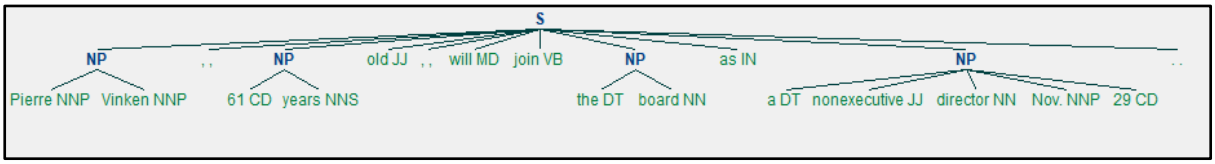
```
Noun phrases: ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people', 'the company',
 'him', 'I', 'you', 'very senior CEOs', 'major American car companies', 'my hand', 'I',
 'Thrun', 'an interview', 'Recode']
Verbs: ['start', 'work', 'drive', 'take', 'tell', 'shake', 'turn', 'talk', 'say']
```

[C] AIM: Named Entity recognition with diagram using NLTK corpus – treebank POS Tagging, chunking and NER.

CODE:

```
import nltk
nltk.download('treebank')
from nltk.corpus import treebank_chunk
treebank_chunk.tagged_sents()[0]
treebank_chunk.chunked_sents()[0]
treebank_chunk.chunked_sents()[0].draw()
```

OUTPUT:



PRACTICAL 6

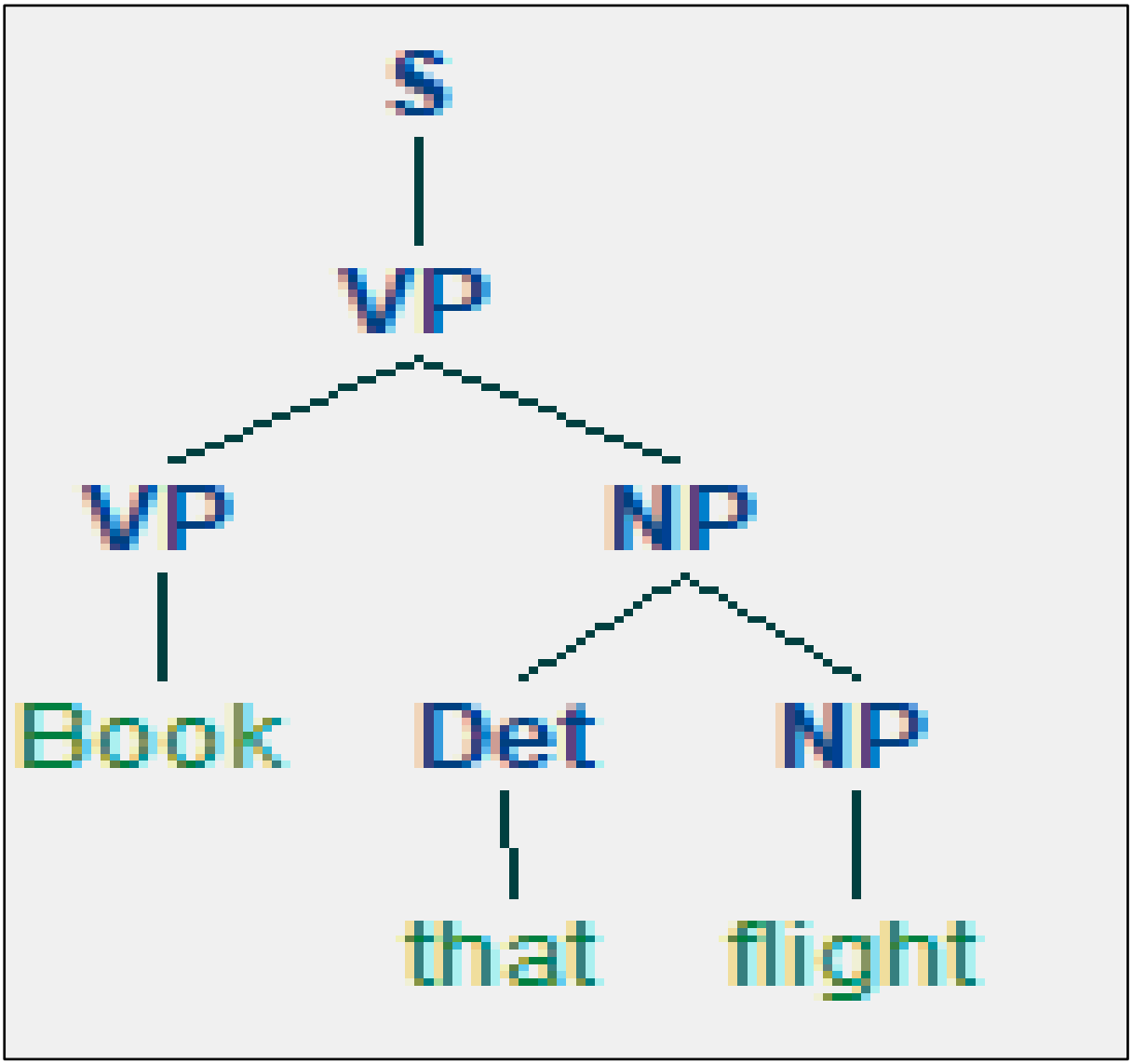
AIM: Finite state automata.

[A] Define grammar using nltk. Analyze a sentence using the same.

CODE:

```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> VP
VP -> VP NP
NP -> Det NP
Det -> 'that'
NP -> singular Noun
NP -> 'flight'
VP -> 'Book'
""")
sentence = "Book that flight"
for index in range(len(sentence)):
    all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
    print(tree)
    tree.draw()
```

OUTPUT:



[B] Accept the input string with Regular expression of Finite Automaton: 101+.

CODE:

```
def FA(s):
    #if the length is less than 3 then it can't be accepted, Therefore end
the process.
    if len(s)<3:
        return "Rejected"
    # first three characters are fixed. Therefore, checking them using index
    if s[0] == '1':
        if s[1] == '0':
            if s[2] == '1':
    # After index 2 only "1" can appear. Therefore break the process if any
other character is detected
                for i in range(3, len(s)):
                    if s[i] != '1':
                        return "Rejected"
                return "Accepted"  # if all 4 nested if true
            return "Rejected"  # else of 3rd if
        return "Rejected"  # else of 2nd if
    return "Rejected"  # else of 1st if

inputs = ['1', '10101', '101', '10111', '01010', '100', '', '10111101',
'1011111']
for i in inputs:
    print(FA(i))
```

OUTPUT:

```
Rejected

Rejected

Accepted

Accepted

Rejected

Rejected

Rejected

Rejected

Accepted
```

[C] Accept the input string with Regular expression of FA: (a+b)*bba.

CODE:

```
def FA(s):

    size = 0
    # scan complete string and make sure that it contains only 'a' & 'b'
    for i in s:
        if i == 'a' or i == 'b':
            size += 1
        else:
            return "Rejected"
    # After checking that it contains only 'a' & 'b'
    # check it's length it should be 3 atleast
    if size >= 3:
    # check the last 3 elements
        if s[size - 3] == 'b':
            if s[size - 2] == 'b':
                if s[size - 1] == 'a':
                    return "Accepted"  # if all 4 if true
                return "Rejected"  # else of 4th if
            return "Rejected"  # else of 3rd if
        return "Rejected"  # else of 2nd if
    return "Rejected"  # else of 1st if

inputs = ['bba', 'ababbba', 'abba', 'abb', 'baba', 'bbb', '']
for i in inputs:
    print(FA(i))
```

OUTPUT:

```
Accepted

Accepted

Accepted

Rejected

Rejected

Rejected

Rejected
```

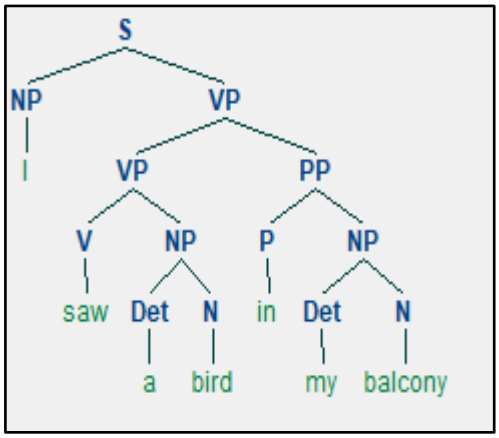
[D] Implementation of Deductive Chart Parsing using context free grammar and a given sentence.

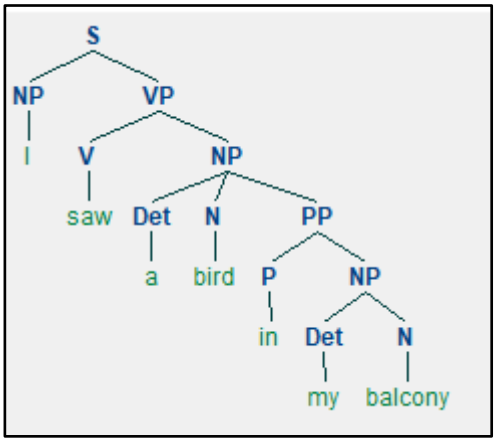
CODE:

```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'a' | 'my'
N -> 'bird' | 'balcony'
V -> 'saw'
P -> 'in'
""")
sentence = "I saw a bird in my balcony"
for index in range(len(sentence)):
    all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)
# all_tokens = ['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
    print(tree)
    tree.draw()
```

OUTPUT:

```
['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
(S
  (NP I)
  (VP
    (VP (V saw) (NP (Det a) (N bird)))
    (PP (P in) (NP (Det my) (N balcony)))))
(S
  (NP I)
  (VP
    (V saw)
    (NP (Det a) (N bird) (PP (P in) (NP (Det my) (N balcony))))))
```





PRACTICAL 7

AIM: Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer and WordNetLemmatizer.

[A] PorterStemmer.

CODE:

```
import nltk
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
print(word_stemmer.stem('writing'))
```

OUTPUT:

```
write
```

[B] LancasterStemmer.

CODE:

```
import nltk
from nltk.stem import LancasterStemmer
Lanc_stemmer = LancasterStemmer()
print(Lanc_stemmer.stem('writing'))
```

OUTPUT:

```
writ
```

[C] RegexpStemmer.

CODE:

```
import nltk
from nltk.stem import RegexpStemmer
Reg_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4)
print(Reg_stemmer.stem('writing'))
```

OUTPUT:

```
writ
```

[D] SnowballStemmer.

CODE:

```
import nltk
from nltk.stem import SnowballStemmer
english_stemmer = SnowballStemmer('english')
print(english_stemmer.stem ('writing'))
```

OUTPUT:

```
write
```

[E] WordNetLemmatizer.

CODE:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print("word :\tlemma")
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

OUTPUT:

```
word :  lemma
rocks : rock
corpora : corpus
better : good
```

PRACTICAL 8

AIM: Implement Naive Bayes classifier.

CODE:

```
#pip install pandas
#pip install sklearn
import pandas as pd
import numpy as np
sms_data = pd.read_csv("spam.csv", encoding='latin-1')
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
stemming = PorterStemmer()
corpus = []
for i in range (0,len(sms_data)):
    s1 = re.sub('[^a-zA-Z]',repl = ' ',string = sms_data['v2'][i])
    s1.lower()
    s1 = s1.split()
    s1 = [stemming.stem(word) for word in s1 if word not in
set(stopwords.words('english'))]
    s1 = ' '.join(s1)
    corpus.append(s1)
from sklearn.feature_extraction.text import CountVectorizer
countvectorizer =CountVectorizer()
x = countvectorizer.fit_transform(corpus).toarray()
print(x)
y = sms_data['v1'].values
print(y)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,
stratify=y,random_state=2)
#Multinomial Naïve Bayes.
from sklearn.naive_bayes import MultinomialNB
multinomialnb = MultinomialNB()
```

```
multinomialnb.fit(x_train,y_train)
# Predicting on test data:
y_pred = multinomialnb.predict(x_test)
print(y_pred)
#Results of our Models
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
print(classification_report(y_test,y_pred))
print("accuracy_score: ",accuracy_score(y_test,y_pred))
```

OUTPUT:

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
['ham' 'ham' 'spam' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
              precision    recall  f1-score   support

         ham       0.99      0.99      0.99      1448
        spam       0.92      0.93      0.92       224

    accuracy                           0.98      1672
   macro avg       0.95      0.96      0.96      1672
weighted avg       0.98      0.98      0.98      1672

accuracy_score:  0.979066985645933
```

PRACTICAL 9

[A] AIM: Speech Tagging:

[I] Speech tagging using spacy.

CODE:

```
import spacy
spacy.cli.download("en_core_web_sm")
sp = spacy.load('en_core_web_sm')
sen = sp(u"I like to play football. I hated it in my childhood though")
print(sen.text)
print(sen[7].pos_)
print(sen[7].tag_)
print(spacy.explain(sen[7].tag_))
for word in sen:
    print(f'{word.text:{12}}        {word.pos_:{10}}        {word.tag_:{8}}
{spacy.explain(word.tag_)}')

sen = sp(u'Can you google it?')
word = sen[2]
print(f'{word.text:{12}}          {word.pos_:{10}}          {word.tag_:{8}}
{spacy.explain(word.tag_)}')
sen = sp(u'Can you search it on google?')
word = sen[5]
print(f'{word.text:{12}}          {word.pos_:{10}}          {word.tag_:{8}}
{spacy.explain(word.tag_)}')
#Finding the Number of POS Tags
sen = sp(u"I like to play football. I hated it in my childhood though")
num_pos = sen.count_by(spacy.attrs.POS)
print(num_pos)
for k,v in sorted(num_pos.items()):
    print(f'{k}. {sen.vocab[k].text:{8}}: {v}')
#Visualizing Parts of Speech Tags
from spacy import displacy
sen = sp(u"I like to play football. I hated it in my childhood though")
```

```
displacy.serve(sen, style='dep', options={'distance': 120})
```

OUTPUT:

```
I like to play football. I hated it in my childhood though
VERB
VBD
verb, past tense
I            PRON       PRP      pronoun, personal
like         VERB       VBP      verb, non-3rd person singular present
to           PART       TO       infinitival "to"
play         VERB       VB       verb, base form
football     NOUN       NN       noun, singular or mass
.            PUNCT      .        punctuation mark, sentence closer
I            PRON       PRP      pronoun, personal
hated        VERB       VBD      verb, past tense
it           PRON       PRP      pronoun, personal
in           ADP        IN       conjunction, subordinating or preposition
my           PRON       PRP$     pronoun, possessive
childhood    NOUN       NN       noun, singular or mass
though       ADV        RB       adverb
google       VERB       VB       verb, base form
google       PROPN      NNP      noun, proper singular
85. ADP     : 1
86. ADV     : 1
92. NOUN    : 2
94. PART    : 1
95. PRON    : 4
97. PUNCT   : 1
100. VERB    : 3

Using the 'dep' visualizer
Serving on http://0.0.0.0:5000 ...
```

[II] Speech tagging using nktl.

CODE:

```
import nltk
nltk.download('state_union')
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer
#create our training and testing data:
train_text = state_union.raw("2005-GWBush.txt")
sample_text = state_union.raw("2006-GWBush.txt")
#train the Punkt tokenizer like:
custom_sent_tokenizer = PunktSentenceTokenizer(train_text)
# tokenize:
tokenized = custom_sent_tokenizer.tokenize(sample_text)
def process_content():
    try:
        for i in tokenized[:2]:
            words = nltk.word_tokenize(i)
            tagged = nltk.pos_tag(words)
            print(tagged)
    except Exception as e:
        print(str(e))

process_content()
```

OUTPUT:

```
[('PRESIDENT', 'NNP'), ('GEORGE', 'NNP'), ('W.', 'NNP'), ('BUSH', 'NNP'), ("'S", 'POS'),
 ('ADDRESS', 'NNP'), ('BEFORE', 'IN'), ('A', 'NNP'), ('JOINT', 'NNP'), ('SESSION', 'NNP'),
 ('OF', 'IN'), ('THE', 'NNP'), ('CONGRESS', 'NNP'), ('ON', 'NNP'), ('THE', 'NNP'), ('STATE',
 'NNP'), ('OF', 'IN'), ('THE', 'NNP'), ('UNION', 'NNP'), ('January', 'NNP'), ('31', 'CD'),
 (',', ','), ('2006', 'CD'), ('THE', 'NNP'), ('PRESIDENT', 'NNP'), (':', ':'), ('Thank',
 'NNP'), ('you', 'PRP'), ('all', 'DT'), ('.', '.')]
[('Mr.', 'NNP'), ('Speaker', 'NNP'), (',', ','), ('Vice', 'NNP'), ('President', 'NNP'),
 ('Cheney', 'NNP'), (',', ','), ('members', 'NNS'), ('of', 'IN'), ('Congress', 'NNP'), (',',
 ','), ('members', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('Supreme', 'NNP'), ('Court', 'NNP'),
 ('and', 'CC'), ('diplomatic', 'JJ'), ('corps', 'NN'), (',', ','), ('distinguished', 'JJ'),
 ('guests', 'NNS'), (',', ','), ('and', 'CC'), ('fellow', 'JJ'), ('citizens', 'NNS'), (':',
 ':'), ('Today', 'VB'), ('our', 'PRP$'), ('nation', 'NN'), ('lost', 'VBD'), ('a', 'DT'),
 ('beloved', 'VBN'), (',', ','), ('graceful', 'JJ'), (',', ','), ('courageous', 'JJ'),
 ('woman', 'NN'), ('who', 'WP'), ('called', 'VBD'), ('America', 'NNP'), ('to', 'TO'), ('its',
 'PRP$'), ('founding', 'NN'), ('ideals', 'NNS'), ('and', 'CC'), ('carried', 'VBD'), ('on',
 'IN'), ('a', 'DT'), ('noble', 'JJ'), ('dream', 'NN'), ('.', '.')]
```

[B] AIM: Statistical Parsing:

[I] Usage of Give and Gave in the Penn Treebank sample.

CODE:

```
#probabilitistic parser
#Usage of Give and Gave in the Penn Treebank sample
import nltk
import nltk.parse.viterbi
import nltk.parse.pchart
def give(t):
    return t.label() == 'VP' and len(t) > 2 and t[1].label() == 'NP'\
        and (t[2].label() == 'PP-DTV' or t[2].label() == 'NP')\
        and ('give' in t[0].leaves() or 'gave' in t[0].leaves())

def sent(t):
    return ' '.join(token for token in t.leaves() if token[0] not in '*-0')
def print_node(t, width):
    output = "%s %s: %s / %s: %s" %\
        (sent(t[0]), t[1].label(), sent(t[1]), t[2].label(), sent(t[2]))
    if len(output) > width:
        output = output[:width] + "..."
    print (output)
for tree in nltk.corpus.treebank.parsed_sents():
    for t in tree.subtrees(give):
        print_node(t, 72)
```

OUTPUT:

```
gave NP: the chefs / NP: a standing ovation
give NP: advertisers / NP: discounts for maintaining or increasing ad sp...
give NP: it / PP-DTV: to the politicians
gave NP: them / NP: similar help
give NP: them / NP:
give NP: only French history questions / PP-DTV: to students in a Europe...
give NP: federal judges / NP: a raise
give NP: consumers / NP: the straight scoop on the U.S. waste crisis
gave NP: Mitsui / NP: access to a high-tech medical product
give NP: Mitsubishi / NP: a window on the U.S. glass industry
give NP: much thought / PP-DTV: to the rates she was receiving , nor to ...
give NP: your Foster Savings Institution / NP: the gift of hope and free...
give NP: market operators / NP: the authority to suspend trading in futu...
gave NP: quick approval / PP-DTV: to $ 3.18 billion in supplemental appr...
give NP: the Transportation Department / NP: up to 50 days to review any...
give NP: the president / NP: such power
give NP: me / NP: the heebie-jeebies
give NP: holders / NP: the right , but not the obligation , to buy a cal...
gave NP: Mr. Thomas / NP: only a `` qualified '' rating , rather than ``...
give NP: the president / NP: line-item veto power
```

[II] Probabilistic Parser.

CODE:

```
import nltk
from nltk import PCFG
grammar = PCFG.fromstring('''
NP -> NNS [0.5] | JJ NNS [0.3] | NP CC NP [0.2]
NNS -> "men" [0.1] | "women" [0.2] | "children" [0.3] | NNS CC NNS [0.4]
JJ -> "old" [0.4] | "young" [0.6]
CC -> "and" [0.9] | "or" [0.1]
''')
print(grammar)
viterbi_parser = nltk.ViterbiParser(grammar)
token = "old men and women".split()
obj = viterbi_parser.parse(token)
print("Output: ")
for x in obj:
    print(x)
```

OUTPUT:

```
Grammar with 11 productions (start state = NP)
    NP -> NNS [0.5]
    NP -> JJ NNS [0.3]
    NP -> NP CC NP [0.2]
    NNS -> 'men' [0.1]
    NNS -> 'women' [0.2]
    NNS -> 'children' [0.3]
    NNS -> NNS CC NNS [0.4]
    JJ -> 'old' [0.4]
    JJ -> 'young' [0.6]
    CC -> 'and' [0.9]
    CC -> 'or' [0.1]
Output:
(NP (JJ old) (NNS (NNS men) (CC and) (NNS women))) (p=0.000864)
```

PRACTICAL 10

[A] AIM: Multiword Expressions in NLP.

CODE:

```
# Multiword Expressions in NLP
from nltk.tokenize import MWETokenizer
from nltk import sent_tokenize, word_tokenize
s = '''Good cake cost Rs.1500\kg in Mumbai. Please buy me one of
them.\n\nThanks.'''
mwe = MWETokenizer([('New', 'York'), ('Hong', 'Kong')], separator='_')
for sent in sent_tokenize(s):
    print(mwe.tokenize(word_tokenize(sent)))
```

OUTPUT:

```
['Good', 'cake', 'cost', 'Rs.1500\\kg', 'in', 'Mumbai', '.']
['Please', 'buy', 'me', 'one', 'of', 'them', '.']
['Thanks', '.']
```

[B] AIM: Normalized Web Distance and Word Similarity.

CODE:

```
import numpy as np
import re
import textdistance # pip install textdistance
# we will need scikit-learn>=0.21
import sklearn #pip install sklearn
from sklearn.cluster import AgglomerativeClustering
texts = [
'Reliance supermarket', 'Reliance hypermarket', 'Reliance', 'Reliance',
'Reliance downtown', 'Reliance market',
'Mumbai', 'Mumbai Hyper', 'Mumbai dxb', 'mumbai airport',
'k.m trading', 'KM Trading', 'KM trade', 'K.M. Trading', 'KM.Trading'
]

def normalize(text):
    """ Keep only lower-cased text and numbers"""
    return re.sub('[^a-z0-9]+', ' ', text.lower())

def group_texts(texts, threshold=0.4):
    """ Replace each text with the representative of its cluster"""
    normalized_texts = np.array([normalize(text) for text in texts])
    distances = 1 - np.array([[textdistance.jaro_winkler(one, another) for
one in normalized_texts] for another in normalized_texts])
    clustering = AgglomerativeClustering(
    distance_threshold=threshold, # this parameter needs to be tuned
carefully
    metric="precomputed", linkage="complete", n_clusters=None
    ).fit(distances)
    centers = dict()
    for cluster_id in set(clustering.labels_):
        index = clustering.labels_ == cluster_id
        centrality = distances[:, index][index].sum(axis=1)
        centers[cluster_id] = normalized_texts[index][centrality.argmin()]
    return [centers[i] for i in clustering.labels_]
```

```
print(group_texts(texts))
```

OUTPUT:

```
['reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'mumbai', 'mumbai',
 'mumbai', 'mumbai', 'km trading', 'km trading', 'km trading', 'km trading', 'km trading']
```

[C] AIM: Word Sense Disambiguation.

CODE:

```
#Word Sense Disambiguation
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet as wn
def get_first_sense(word, pos=None):
    if pos:
        synsets = wn.synsets(word,pos)
    else:
        synsets = wn.synsets(word)
    return synsets[0]
best_synset = get_first_sense('bank')
print ('%s: %s' % (best_synset.name, best_synset.definition))
best_synset = get_first_sense('set','n')
print ('%s: %s' % (best_synset.name, best_synset.definition))
best_synset = get_first_sense('set','v')
print ('%s: %s' % (best_synset.name, best_synset.definition))
```

OUTPUT:

```
<bound method Synset.name of Synset('bank.n.01')>: <bound method Synset.definition of
 Synset('bank.n.01')>
<bound method Synset.name of Synset('set.n.01')>: <bound method Synset.definition of Synset
 ('set.n.01')>
<bound method Synset.name of Synset('put.v.01')>: <bound method Synset.definition of Synset
 ('put.v.01')>
```

University of Mumbai

PRACTICAL JOURNAL – ELECTIVE II

PSIT4P3a

Deep Learning

SUBMITTED BY

Khan Hajra Mohammed Rashid

SEAT NO 40150

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 Ms. **Khan Hajra Mohammed Rashid Seat No. 40150** studying in **Master of Science** in **Information Technology Part II Semester IV** has satisfactorily completed the Practical of **PSIT4P3a Deep Learning** 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

9	10/05/2023	Implementation of convolutional neural network to predict numbers from number Images.	50	
10	17/05/2023	Denoising of images using autoencoder.	53	

PRACTICAL 1

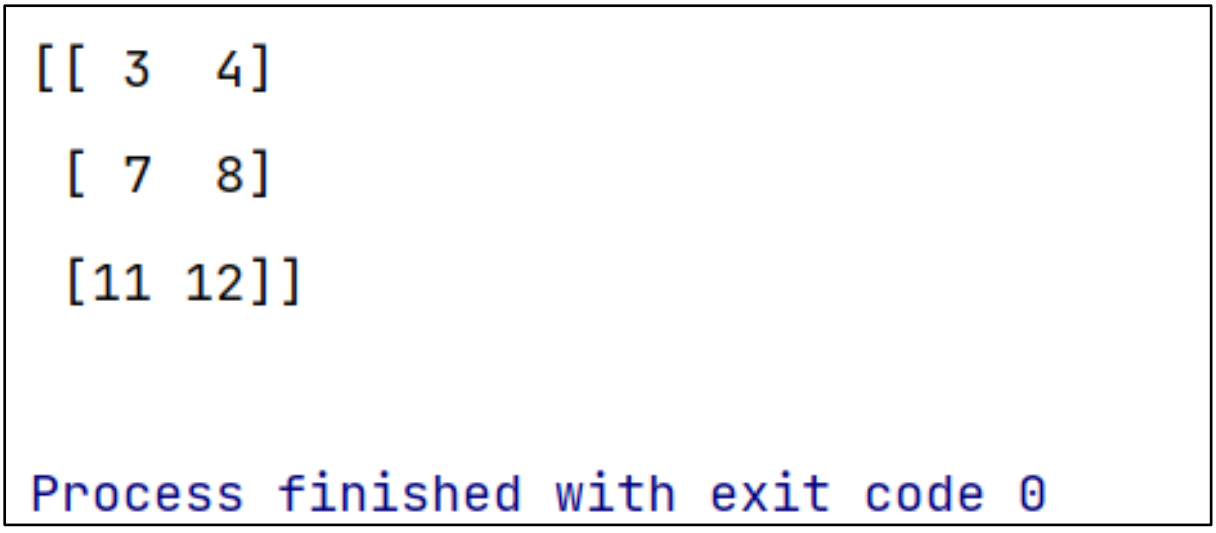
[A] AIM: Perform basic mathematics operations in python.

[I] Addition of matrices.

CODE:

```
import numpy as np
A = np.array([[1, 2], [3, 4], [5, 6]])
A
B = np.array([[2, 5], [7, 4], [4, 3]])
B
# Add matrices A and B
C = A + B
print(C)
```

OUTPUT:



[II] Multiplication of matrices.

CODE:

```
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
print(A)
B = np.array([[2, 7], [1, 2], [3, 6]])
print(B)
C = A.dot(B)
print(C)
```

OUTPUT:

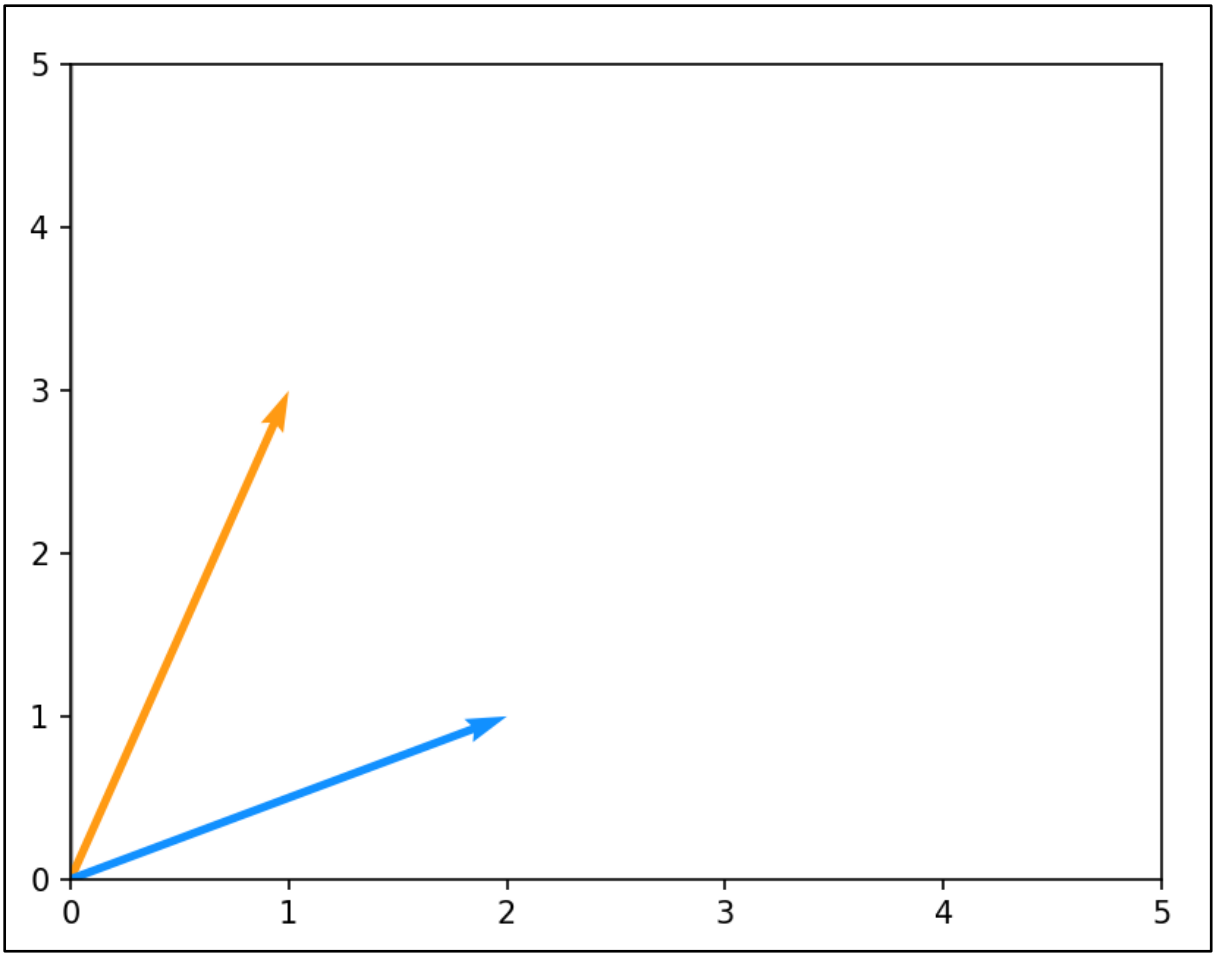
```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[2 7]
 [1 2]
 [3 6]]
[[ 13  29]
 [ 31  74]
 [ 49 119]
 [ 67 164]]
```

[III] Linear Combination.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt


def plotVectors(vecs, cols, alpha =1 ):
    plt.figure()
    plt.axvline(x=0, color = '#A9A9A9', zorder = 0)
    for i in range(len(vecs)):
        x = np.concatenate([[0,0],vecs[i]])
        plt.quiver([x[0]],
               [x[1]],
               [x[2]],
               [x[3]],
               angles='xy', scale_units='xy', scale=1,color=cols[i],
alpha=alpha)
orange= '#FF9A13'
blue= '#1190FF'
plotVectors([[1,3],[2,1]],[orange,blue])
plt.xlim(0,5)
plt.ylim(0,5)
plt.show()
```

OUTPUT:

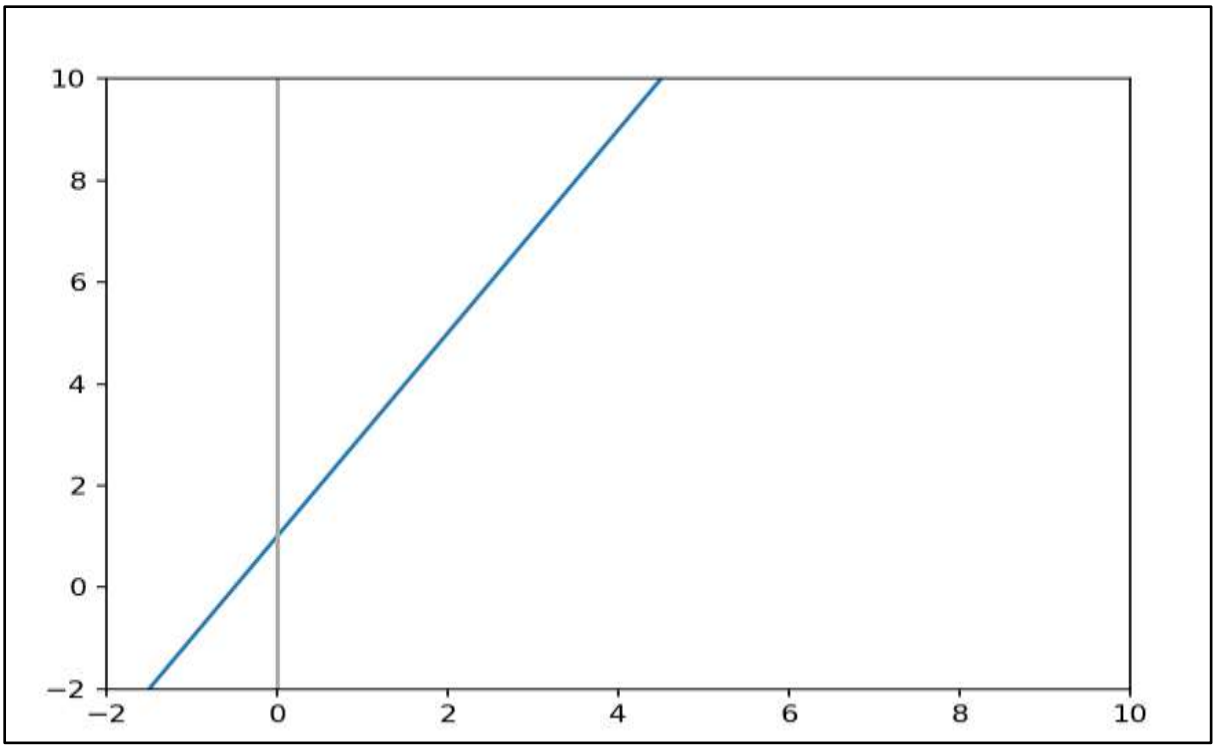


[IVa] Linear Equation.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt


x = np.arange(-10,10)
y = 2*x + 1


plt.figure()
plt.plot(x,y)
plt.xlim(-2,10)
plt.ylim(-2,10)


plt.axvline(x=0, color = '#A9A9A9')
plt.show()
```

OUTPUT:

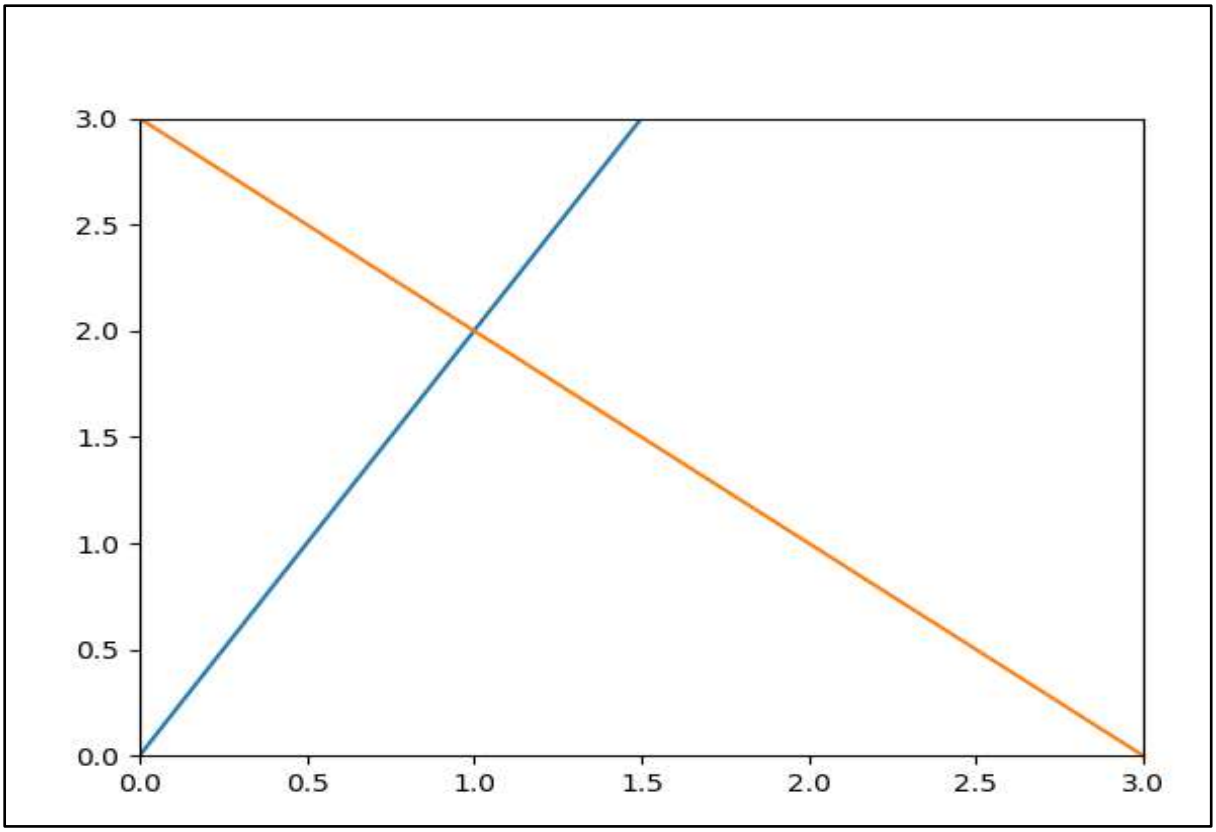


[IVb] Linear Equation.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt


x = np.arange(-10,10)
y = 2*x
y1 = -x + 3
plt.figure()
plt.plot(x,y)
plt.plot(x,y1)
plt.xlim(0,3)
plt.ylim(0,3)
plt.axvline(x=0, color='grey')
plt.show()
```

OUTPUT:

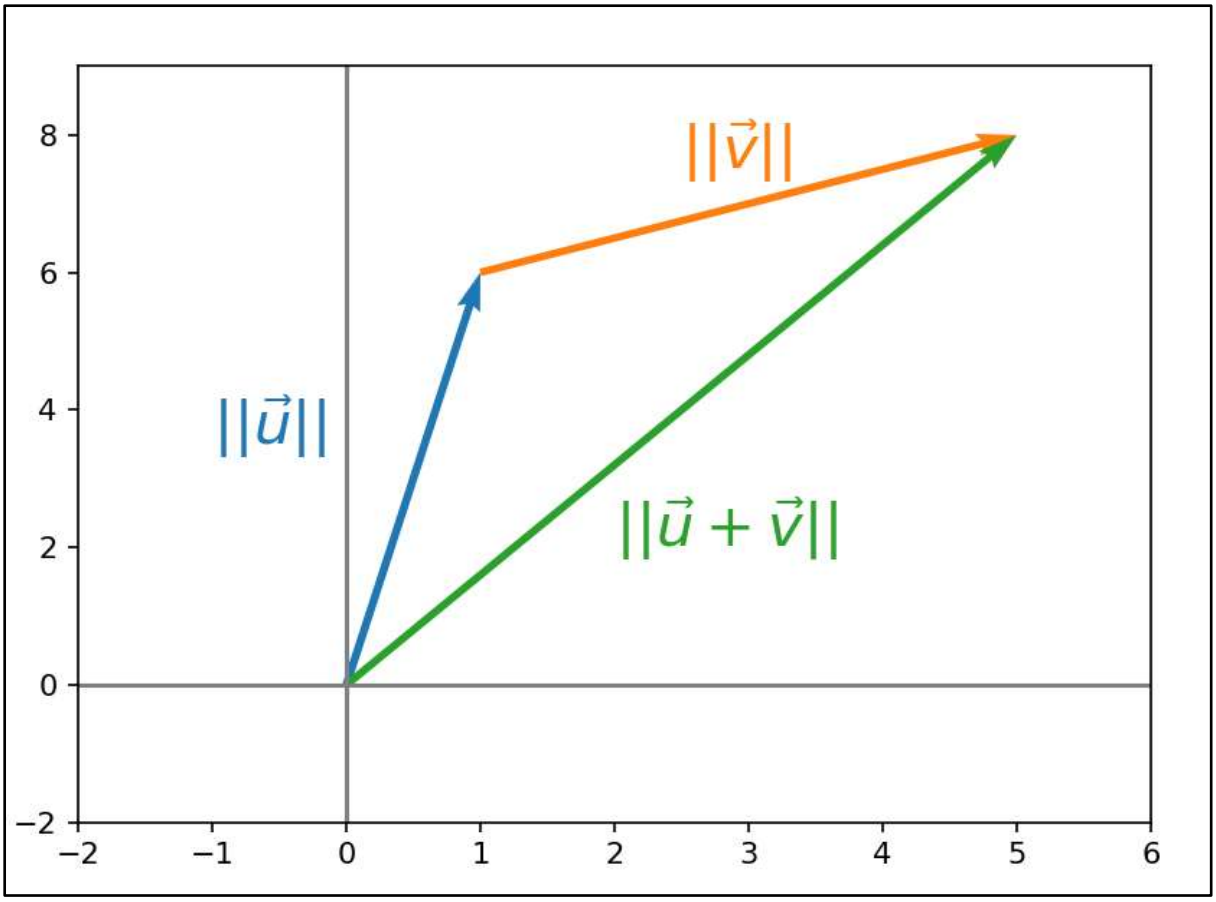


[V] Norm.

CODE:

```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


u = [0,0,1,6]

v = [0,0,4,2]

u_bis = [1,6,v[2],v[3]]

w = [0,0,5,8]

plt.quiver([u[0], u_bis[0], w[0]],

           [u[1], u_bis[1], w[1]],

           [u[2], u_bis[2], w[2]],

           [u[3], u_bis[3], w[3]],

           angles = 'xy', scale_units = 'xy', scale = 1, color =
sns.color_palette())


plt.xlim(-2,6)

plt.ylim(-2,9)

plt.axvline(x=0, color='grey')

plt.axhline(y=0, color='grey')

plt.text(-1, 3.5, r'$||\vec{u}||$', color = sns.color_palette()[0], size
=20)

plt.text(2.5, 7.5, r'$||\vec{v}||$', color = sns.color_palette()[1], size
=20)

plt.text(2, 2, r'$||\vec{u}+\vec{v}||$', color = sns.color_palette()[2],
size =20)

plt.show()
```

OUTPUT:



[VI] Symmetric Matrices.

CODE:

```
import numpy as np


A = np.array([[2,4,-1],[4,-8,0],[-1,0,3]])
print(A)
print(A.T)
```

OUTPUT:

```
C:\Users\Admin\PycharmProjects\DLPracs
[[ 2  4 -1]
 [ 4 -8  0]
 [-1  0  3]]
[[ 2  4 -1]
 [ 4 -8  0]
 [-1  0  3]]
```

[B] AIM: Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow.

CODE:

```
import tensorflow as tf

print("Matrix Multiplication Demo")

x=tf.constant([1,2,3,4,5,6],shape=[2,3])

print(x)

y=tf.constant([7,8,9,10,11,12],shape=[3,2])

print(y)

z=tf.matmul(x,y)

print("Product:",z)

e_matrix_A=tf.random.uniform([2,2],minval=3,maxval=10,dtype=tf.float32,name
="matrixA")

print("Matrix A:\n{}\n\n".format(e_matrix_A))

eigen_values_A,eigen_vectors_A=tf.linalg.eigh(e_matrix_A)

print("Eigen Vectors:\n{}\n\n Eigen  Values:

\n{}\n".format(eigen_vectors_A,eigen_values_A))
```

OUTPUT:

```
Matrix Multiplication Demo
tf.Tensor(
[[1 2 3]
 [4 5 6]], shape=(2, 3), dtype=int32)
tf.Tensor(
[[ 7  8]
 [ 9 10]
 [11 12]], shape=(3, 2), dtype=int32)
Product: tf.Tensor(
[[ 58  64]
 [139 154]], shape=(2, 2), dtype=int32)
Matrix A:
[[9.669554  9.363649 ]
 [6.5489373 6.7324896]]


Eigen Vectors:
[[-0.6249776   0.78064275]
 [ 0.78064275  0.6249776 ]]

 Eigen   Values:
[ 1.4894521 14.912593 ]


Process finished with exit code 0
```

PRACTICAL 2

AIM: Solving XOR problem using deep feed forward network.

CODE:

```
import numpy as np
from keras.layers import Dense
from keras.models import Sequential
model=Sequential()
model.add(Dense(units=2,activation='relu',input_dim=2))
model.add(Dense(units=1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
print(model.summary())
print(model.get_weights())
X=np.array([[0.,0.],[0.,1.],[1.,0.],[1.,1.]])
Y=np.array([0.,1.,1.,0.])
model.fit(X,Y,epochs=900,batch_size=4)
print(model.get_weights())
print(model.predict(X,batch_size=4))
```

OUTPUT:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 6

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
None
[array([[ 0.36801147, -1.188614  ],
       [ 0.7940141 ,  0.95810044]], dtype=float32), array([0., 0.], dtype=float32), array([[-0.81275207],
       [-0.59155834]], dtype=float32), array([0.], dtype=float32)]
```

```
Epoch 1/1000
1/1 [==============================] - 2s 2s/step - loss: 0.8370 - accuracy: 0.5000
Epoch 2/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.8360 - accuracy: 0.2500
Epoch 3/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.8349 - accuracy: 0.2500
Epoch 4/1000
1/1 [==============================] - 0s 10ms/step - loss: 0.8339 - accuracy: 0.2500
Epoch 5/1000
1/1 [==============================] - 0s 15ms/step - loss: 0.8328 - accuracy: 0.2500
Epoch 6/1000
1/1 [==============================] - 0s 10ms/step - loss: 0.8318 - accuracy: 0.2500
Epoch 7/1000
1/1 [==============================] - 0s 10ms/step - loss: 0.8308 - accuracy: 0.2500
Epoch 8/1000
1/1 [==============================] - 0s 18ms/step - loss: 0.8298 - accuracy: 0.2500
Epoch 9/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.8288 - accuracy: 0.2500
Epoch 10/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.8277 - accuracy: 0.2500
```

```
Epoch 11/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.8267 - accuracy: 0.2500
Epoch 12/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.8257 - accuracy: 0.2500
Epoch 13/1000
1/1 [==============================] - 0s 17ms/step - loss: 0.8247 - accuracy: 0.2500
Epoch 14/1000
1/1 [==============================] - 0s 10ms/step - loss: 0.8238 - accuracy: 0.2500
Epoch 15/1000
1/1 [==============================] - 0s 106ms/step - loss: 0.8228 - accuracy: 0.2500
Epoch 16/1000
1/1 [==============================] - 0s 13ms/step - loss: 0.8218 - accuracy: 0.2500
Epoch 17/1000
1/1 [==============================] - 0s 14ms/step - loss: 0.8208 - accuracy: 0.2500
Epoch 18/1000
1/1 [==============================] - 0s 30ms/step - loss: 0.8199 - accuracy: 0.2500
Epoch 19/1000
1/1 [==============================] - 0s 32ms/step - loss: 0.8189 - accuracy: 0.2500
Epoch 20/1000
1/1 [==============================] - 0s 19ms/step - loss: 0.8179 - accuracy: 0.2500
```

```
Epoch 995/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.5997 - accuracy: 0.7500
Epoch 996/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.5996 - accuracy: 0.7500
Epoch 997/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.5994 - accuracy: 0.7500
Epoch 998/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.5993 - accuracy: 0.7500
Epoch 999/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.5992 - accuracy: 0.7500
Epoch 1000/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.5991 - accuracy: 0.7500
[array([[ 0.6527725 , -1.188614  ],
       [ 0.65223974,  0.47420728]], dtype=float32), array([-0.6530008 , -0.48389307], dtype=float32), array([[-1.389874 ],
       [-0.2258762]], dtype=float32), array([0.18355492], dtype=float32)]
1/1 [==============================] - 0s 176ms/step
[[0.5457603 ]
 [0.5457603 ]
 [0.5457603 ]
 [0.32680774]]
```

PRACTICAL 3

AIM: Implementing deep neural network for performing binary classification task.

CODE:

```
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense
dataset = loadtxt('diabetes.csv',delimiter= ',',skiprows = 1)
X = dataset[:,0:8]
Y = dataset[:,8]
print(X)
print(Y)
model =Sequential()
model.add(Dense(12,input_dim=8,activation='relu'))
model.add(Dense(8,input_dim=8,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
model.fit(X,Y,batch_size=12,epochs=500)
Accuracy=model.evaluate(X,Y)
print('Accuracy of model is ',(Accuracy*100))
prediction=model.predict(X)
exec('for i in range(5):print(X[i].tolist(),prediction[i],Y[i])')
```

OUTPUT:

```
Epoch 1/150
77/77 [==============================] - 2s 3ms/step - loss: 2.5263 - accuracy: 0.5586
Epoch 2/150
77/77 [==============================] - 0s 3ms/step - loss: 1.1873 - accuracy: 0.5885
Epoch 3/150
77/77 [==============================] - 0s 3ms/step - loss: 1.0318 - accuracy: 0.5443
Epoch 4/150
77/77 [==============================] - 0s 3ms/step - loss: 0.9486 - accuracy: 0.5404
Epoch 5/150
77/77 [==============================] - 0s 3ms/step - loss: 0.8373 - accuracy: 0.5651
Epoch 6/150
77/77 [==============================] - 0s 3ms/step - loss: 0.7979 - accuracy: 0.5794
Epoch 7/150
77/77 [==============================] - 0s 3ms/step - loss: 0.7693 - accuracy: 0.5951
Epoch 8/150
77/77 [==============================] - 0s 3ms/step - loss: 0.7384 - accuracy: 0.6094
Epoch 9/150
77/77 [==============================] - 0s 3ms/step - loss: 0.7010 - accuracy: 0.6289
Epoch 10/150
77/77 [==============================] - 0s 4ms/step - loss: 0.6959 - accuracy: 0.6328
Epoch 11/150
77/77 [==============================] - 0s 5ms/step - loss: 0.6670 - accuracy: 0.6523
```

```
Accuracy of model is [0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375,
0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0
.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125, 0.7734375, 0.479278564453125,
```

```
24/24 [==============================] - 0s 2ms/step
[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0] [0.73097056] 1.0
[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0] [0.11377989] 0.0
[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0] [0.87028414] 1.0
[1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0] [0.08588263] 0.0
[0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0] [0.66002005] 1.0
```

PRACTICAL 4

[A] AIM: Using deep feed forward network with two hidden layers for performing multiclass classification and predicting the class.

CODE:

```
import numpy as np
from sklearn.datasets import load_iris
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split

# Load the iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Convert target variable to one-hot encoded format
y = to_categorical(y)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Define the model
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(y_train.shape[1], activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.2)
```

```
# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test)
print('Test accuracy:', accuracy)
```

OUTPUT:

```
Epoch 1/100
3/3 [==============================] - 1s 73ms/step - loss: 1.2578 - accuracy: 0.3646 - val_loss: 1.2153 - val_accuracy: 0.2083
Epoch 2/100
3/3 [==============================] - 0s 11ms/step - loss: 1.0729 - accuracy: 0.3750 - val_loss: 1.0499 - val_accuracy: 0.5000
Epoch 3/100
3/3 [==============================] - 0s 12ms/step - loss: 0.9695 - accuracy: 0.6667 - val_loss: 0.9646 - val_accuracy: 0.4167
Epoch 4/100
3/3 [==============================] - 0s 12ms/step - loss: 0.9031 - accuracy: 0.6875 - val_loss: 0.9082 - val_accuracy: 0.5000
Epoch 5/100
3/3 [==============================] - 0s 12ms/step - loss: 0.8462 - accuracy: 0.6979 - val_loss: 0.8486 - val_accuracy: 0.5000
Epoch 6/100
3/3 [==============================] - 0s 12ms/step - loss: 0.7968 - accuracy: 0.7188 - val_loss: 0.7993 - val_accuracy: 0.5833
Epoch 7/100
3/3 [==============================] - 0s 12ms/step - loss: 0.7542 - accuracy: 0.8021 - val_loss: 0.7638 - val_accuracy: 0.7083
Epoch 8/100
3/3 [==============================] - 0s 12ms/step - loss: 0.7192 - accuracy: 0.9271 - val_loss: 0.7258 - val_accuracy: 0.9583
Epoch 9/100
3/3 [==============================] - 0s 12ms/step - loss: 0.6898 - accuracy: 0.9479 - val_loss: 0.6879 - val_accuracy: 0.9583
Epoch 10/100
3/3 [==============================] - 0s 13ms/step - loss: 0.6613 - accuracy: 0.8646 - val_loss: 0.6570 - val_accuracy: 0.9583
Epoch 11/100
3/3 [==============================] - 0s 11ms/step - loss: 0.6327 - accuracy: 0.8229 - val_loss: 0.6339 - val_accuracy: 0.9583
Epoch 12/100
3/3 [==============================] - 0s 11ms/step - loss: 0.6055 - accuracy: 0.8750 - val_loss: 0.6152 - val_accuracy: 0.9583
Epoch 13/100
3/3 [==============================] - 0s 13ms/step - loss: 0.5821 - accuracy: 0.9479 - val_loss: 0.6024 - val_accuracy: 1.0000
Epoch 14/100
3/3 [==============================] - 0s 13ms/step - loss: 0.5601 - accuracy: 0.9583 - val_loss: 0.5797 - val_accuracy: 1.0000
Epoch 15/100
3/3 [==============================] - 0s 13ms/step - loss: 0.5368 - accuracy: 0.9688 - val_loss: 0.5649 - val_accuracy: 1.0000
Epoch 16/100
3/3 [==============================] - 0s 13ms/step - loss: 0.5164 - accuracy: 0.9688 - val_loss: 0.5517 - val_accuracy: 0.9583
Epoch 17/100
3/3 [==============================] - 0s 12ms/step - loss: 0.4991 - accuracy: 0.9792 - val_loss: 0.5404 - val_accuracy: 0.9583
Epoch 18/100
3/3 [==============================] - 0s 11ms/step - loss: 0.4832 - accuracy: 0.9792 - val_loss: 0.5241 - val_accuracy: 0.9583
Epoch 19/100
3/3 [==============================] - 0s 10ms/step - loss: 0.4679 - accuracy: 0.9688 - val_loss: 0.5122 - val_accuracy: 0.9583
```

```
3/3 [==============================] - 0s 14ms/step - loss: 0.1055 - accuracy: 0.9688 - val_loss: 0.0829 - val_accuracy: 1.0000
Epoch 99/100
3/3 [==============================] - 0s 14ms/step - loss: 0.1041 - accuracy: 0.9688 - val_loss: 0.0790 - val_accuracy: 1.0000
Epoch 100/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1025 - accuracy: 0.9688 - val_loss: 0.0853 - val_accuracy: 1.0000
1/1 [==============================] - 0s 20ms/step - loss: 0.1196 - accuracy: 0.9333
Test accuracy: 0.9333333373069763
```

[B] AIM: Using a deep feed forward network with two hidden layers for performing classification and predicting the probability of class.

CODE:

```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import make_blobs
from sklearn.preprocessing import MinMaxScaler

X, Y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
scalar = MinMaxScaler()
scalar.fit(X)
X = scalar.transform(X)
models = keras.Sequential()
models.add(Dense(4, input_dim=2, activation='relu'))
models.add(Dense(4, activation='relu'))
models.add(Dense(1, activation='sigmoid'))
models.compile(loss='binary_crossentropy', optimizer='adam')
models.fit(X, Y, epochs=500)
import numpy as np

Xnew, Yreal = make_blobs(n_samples=3, centers=2, n_features=2,
random_state=1)
Xnew = scalar.transform(Xnew)
Yclass = np.argmax(models.predict(Xnew), axis=-1)
Ynew = models.predict(Xnew)
for i in range(len(Xnew)):
print("X=%s,Predicted_probability=%s,Predicted_class=%s" % (Xnew[i],
Ynew[i], Yclass[i]))
```

OUTPUT:

```
Epoch 1/500
4/4 [==============================] - 1s 2ms/step - loss: 1.0409
Epoch 2/500
4/4 [==============================] - 0s 1ms/step - loss: 1.0221
Epoch 3/500
4/4 [==============================] - 0s 1ms/step - loss: 1.0026
Epoch 4/500
4/4 [==============================] - 0s 990us/step - loss: 0.9839
Epoch 5/500
4/4 [==============================] - 0s 999us/step - loss: 0.9667
Epoch 6/500
4/4 [==============================] - 0s 1ms/step - loss: 0.9513
Epoch 7/500
4/4 [==============================] - 0s 999us/step - loss: 0.9345
Epoch 8/500
4/4 [==============================] - 0s 1ms/step - loss: 0.9191
Epoch 9/500
4/4 [==============================] - 0s 661us/step - loss: 0.9056
Epoch 10/500
4/4 [==============================] - 0s 1ms/step - loss: 0.8912
Epoch 11/500
4/4 [==============================] - 0s 1ms/step - loss: 0.8772
Epoch 12/500
4/4 [==============================] - 0s 667us/step - loss: 0.8656
Epoch 13/500
4/4 [==============================] - 0s 1ms/step - loss: 0.8546
Epoch 14/500
4/4 [==============================] - 0s 1ms/step - loss: 0.8423
Epoch 15/500
4/4 [==============================] - 0s 1ms/step - loss: 0.8290
Epoch 16/500
4/4 [==============================] - 0s 667us/step - loss: 0.8169
Epoch 17/500
4/4 [==============================] - 0s 666us/step - loss: 0.8043
Epoch 18/500
4/4 [==============================] - 0s 1ms/step - loss: 0.7907
Epoch 19/500
4/4 [==============================] - 0s 1ms/step - loss: 0.7784
```

```
Epoch 485/500
4/4 [==============================] - 0s 1ms/step - loss: 0.0257
Epoch 486/500
4/4 [==============================] - 0s 1ms/step - loss: 0.0255
Epoch 487/500
4/4 [==============================] - 0s 1ms/step - loss: 0.0253
Epoch 488/500
4/4 [==============================] - 0s 999us/step - loss: 0.0252
Epoch 489/500
4/4 [==============================] - 0s 1ms/step - loss: 0.0250
Epoch 490/500
4/4 [==============================] - 0s 989us/step - loss: 0.0248
Epoch 491/500
4/4 [==============================] - 0s 0s/step - loss: 0.0247
Epoch 492/500
4/4 [==============================] - 0s 0s/step - loss: 0.0245
Epoch 493/500
4/4 [==============================] - 0s 0s/step - loss: 0.0243
Epoch 494/500
4/4 [==============================] - 0s 0s/step - loss: 0.0242
Epoch 495/500
4/4 [==============================] - 0s 6ms/step - loss: 0.0240
Epoch 496/500
4/4 [==============================] - 0s 665us/step - loss: 0.0239
Epoch 497/500
4/4 [==============================] - 0s 1ms/step - loss: 0.0237
Epoch 498/500
4/4 [==============================] - 0s 667us/step - loss: 0.0236
Epoch 499/500
4/4 [==============================] - 0s 1ms/step - loss: 0.0234
Epoch 500/500
4/4 [==============================] - 0s 2ms/step - loss: 0.0233
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 18ms/step
X=[0.89337759 0.65864154],Predicted_probability=[0.01474354],Predicted_class=0
X=[0.29097707 0.12978982],Predicted_probability=[0.9631602],Predicted_class=0
X=[0.78082614 0.75391697],Predicted_probability=[0.012025],Predicted_class=0
```

[C] AIM: Using a deep feed forward network with two hidden layers for performing linear regression and predicting values.

CODE:

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import make_regression
from sklearn.preprocessing import MinMaxScaler

X,Y=make_regression(n_samples=100,n_features=2,noise=0.1,random_state=1)
scalarX,scalarY=MinMaxScaler(),MinMaxScaler()

scalarX.fit(X)
scalarY.fit(Y.reshape(100,1))
X=scalarX.transform(X)
Y=scalarY.transform(Y.reshape(100,1))

model=Sequential()
model.add(Dense(4,input_dim=2,activation='relu'))
model.add(Dense(4,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='mse',optimizer='adam')
model.fit(X,Y,epochs=1000,verbose=0)

Xnew,a=make_regression(n_samples=3,n_features=2,noise=0.1,random_state=1)
Xnew=scalarX.transform(Xnew)

Ynew=model.predict(Xnew)

for i in range(len(Xnew)):
    print("X=%s,Predicted=%s"%(Xnew[i],Ynew[i]))
```

OUTPUT:

```
1/1 [==============================] - 0s 126ms/step
X=[0.29466096 0.30317302],Predicted=[0.18755545]
X=[0.39445118 0.79390858],Predicted=[0.7539139]
X=[0.02884127 0.6208843 ],Predicted=[0.3981339]
```

PRACTICAL 5

[A] AIM: Evaluating feed forward deep network for regression using KFold cross validation.

CODE:

```
import numpy as np
from sklearn.model_selection import KFold
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Generate sample data
X = np.random.rand(1000, 10)
y = np.sum(X, axis=1)

# Define KFold cross-validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize list to store evaluation metrics
eval_metrics = []

# Iterate through each fold
for train_index, test_index in kfold.split(X):
   # Split data into training and testing sets
   X_train, X_test = X[train_index], X[test_index]
   y_train, y_test = y[train_index], y[test_index]

   # Define and compile model
   model = Sequential()
   model.add(Dense(64, activation='relu', input_dim=10))
   model.add(Dense(1))
   model.compile(optimizer='adam', loss='mse', metrics=['mae'])

   # Fit model to training data
   model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=0)

   # Evaluate model on testing data
   eval_metrics.append(model.evaluate(X_test, y_test))
```

```
# Print average evaluation metrics across all folds
print("Average evaluation metrics:")
print("Loss:", np.mean([m[0] for m in eval_metrics]))
print("MAE:", np.mean([m[1] for m in eval_metrics]))
```

OUTPUT:

```
7/7 [==============================] - 0s 1ms/step - loss: 3.7221e-04 - mae: 0.0139
7/7 [==============================] - 0s 1ms/step - loss: 1.7676e-04 - mae: 0.0104
7/7 [==============================] - 0s 837us/step - loss: 5.5438e-04 - mae: 0.0157
7/7 [==============================] - 0s 843us/step - loss: 4.1214e-04 - mae: 0.0163
7/7 [==============================] - 0s 845us/step - loss: 3.4817e-04 - mae: 0.0105
Average evaluation metrics:
Loss: 0.00037273057969287036
MAE: 0.013363478891551494
```

[B] AIM: Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.

CODE:

```
from keras.models import Sequential
from keras.layers import Dense
from scikeras.wrappers import KerasClassifier
from keras.utils import to_categorical
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn import datasets
from sklearn.pipeline import Pipeline

# load dataset
dataset = datasets.load_iris()
X = dataset.data[:, 0:4].astype(float)
Y = dataset.target

# encode class values as integers
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)

# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = to_categorical(encoded_Y)

# define baseline model
def baseline_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim=4, activation='relu'))
    model.add(Dense(3, activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model
```

```
estimator = KerasClassifier(model=baseline_model, epochs=200, batch_size=5,
verbose=0)
kfold = KFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X, dummy_y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean() * 100, results.std() *
100))
```

OUTPUT:

```
Baseline: 96.67% (3.33%)

Process finished with exit code 0
```

PRACTICAL 6

AIM: Implementing regularization to avoid overfitting in binary classification.

[A] Without regularization.

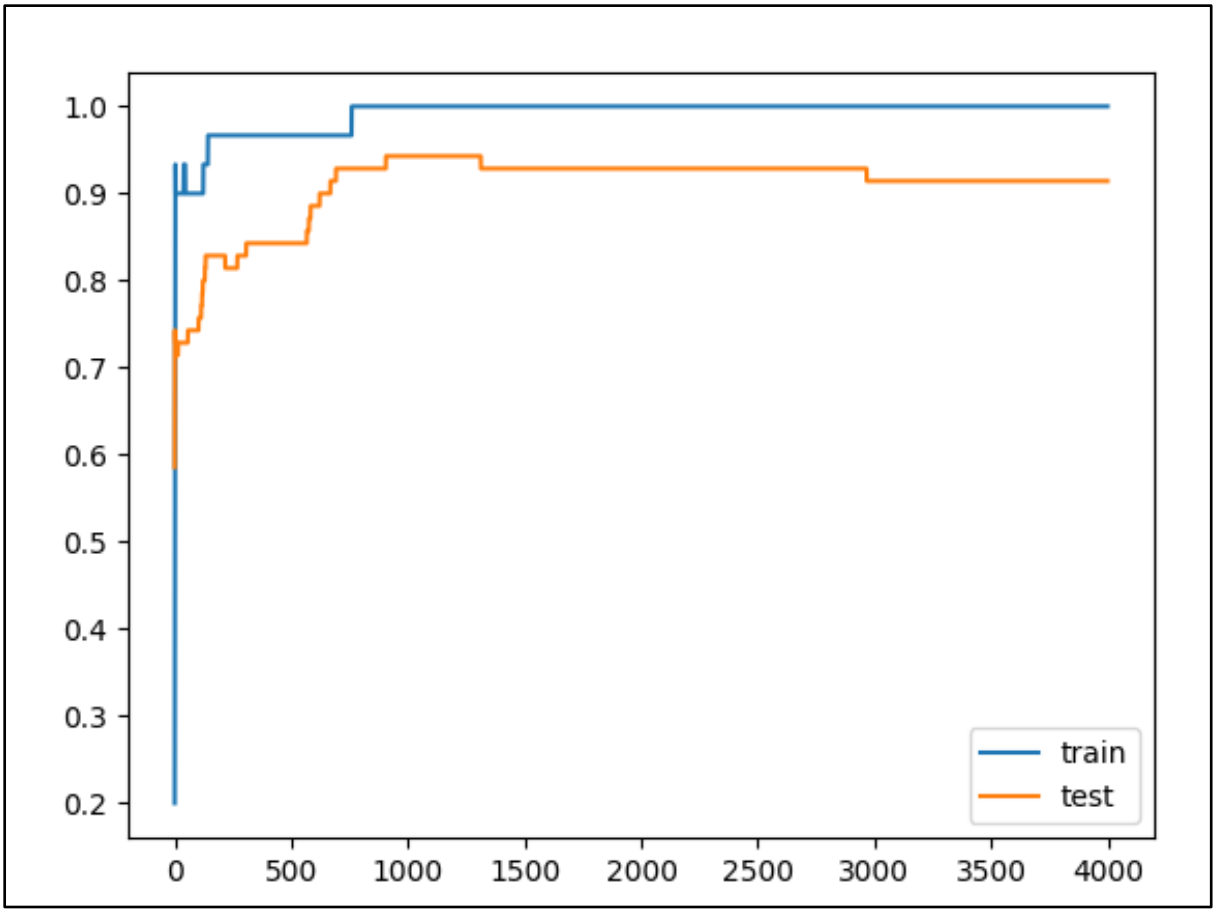
CODE:

```
from matplotlib import pyplot
from sklearn.datasets import make_moons
from keras.models import Sequential
from keras.layers import Dense
X,Y=make_moons(n_samples=100,noise=0.2,random_state=1)
n_train=30
trainX,testX=X[:n_train,:],X[n_train:]
trainY,testY=Y[:n_train],Y[n_train:]
model= Sequential()
model.add(Dense(500,input_dim=2,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
history=model.fit(trainX,trainY,validation_data=(testX,testY),epochs=4000)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test')
pyplot.legend()
pyplot.show()
```

OUTPUT:

```
Epoch 1/4000
1/1 [==============================] - 1s 720ms/step - loss: 0.7134 - accuracy: 0.2000 - val_loss: 0.6934 - val_accuracy: 0.5857
Epoch 2/4000
1/1 [==============================] - 0s 24ms/step - loss: 0.6956 - accuracy: 0.5000 - val_loss: 0.6818 - val_accuracy: 0.7429
Epoch 3/4000
1/1 [==============================] - 0s 25ms/step - loss: 0.6783 - accuracy: 0.8333 - val_loss: 0.6706 - val_accuracy: 0.7429
Epoch 4/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.6616 - accuracy: 0.9333 - val_loss: 0.6598 - val_accuracy: 0.7286
Epoch 5/4000
1/1 [==============================] - 0s 25ms/step - loss: 0.6453 - accuracy: 0.9000 - val_loss: 0.6493 - val_accuracy: 0.7286
Epoch 6/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.6295 - accuracy: 0.9000 - val_loss: 0.6393 - val_accuracy: 0.7286
Epoch 7/4000
1/1 [==============================] - 0s 26ms/step - loss: 0.6141 - accuracy: 0.9000 - val_loss: 0.6296 - val_accuracy: 0.7286
Epoch 8/4000
1/1 [==============================] - 0s 24ms/step - loss: 0.5992 - accuracy: 0.9000 - val_loss: 0.6204 - val_accuracy: 0.7286
Epoch 9/4000
1/1 [==============================] - 0s 25ms/step - loss: 0.5848 - accuracy: 0.9000 - val_loss: 0.6114 - val_accuracy: 0.7286
Epoch 10/4000
1/1 [==============================] - 0s 24ms/step - loss: 0.5708 - accuracy: 0.9000 - val_loss: 0.6029 - val_accuracy: 0.7286
Epoch 11/4000
1/1 [==============================] - 0s 24ms/step - loss: 0.5572 - accuracy: 0.9000 - val_loss: 0.5946 - val_accuracy: 0.7143
Epoch 12/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.5440 - accuracy: 0.9000 - val_loss: 0.5867 - val_accuracy: 0.7143
Epoch 13/4000
1/1 [==============================] - 0s 25ms/step - loss: 0.5311 - accuracy: 0.9000 - val_loss: 0.5792 - val_accuracy: 0.7143
Epoch 14/4000
1/1 [==============================] - 0s 27ms/step - loss: 0.5187 - accuracy: 0.9000 - val_loss: 0.5719 - val_accuracy: 0.7143
Epoch 15/4000
1/1 [==============================] - 0s 28ms/step - loss: 0.5066 - accuracy: 0.9000 - val_loss: 0.5649 - val_accuracy: 0.7143
Epoch 16/4000
1/1 [==============================] - 0s 27ms/step - loss: 0.4949 - accuracy: 0.9000 - val_loss: 0.5583 - val_accuracy: 0.7286
```

```
Epoch 3987/4000
1/1 [==============================] - 0s 26ms/step - loss: 2.1828e-04 - accuracy: 1.0000 - val_loss: 0.4914 - val_accuracy: 0.9143
Epoch 3988/4000
1/1 [==============================] - 0s 24ms/step - loss: 2.1811e-04 - accuracy: 1.0000 - val_loss: 0.4914 - val_accuracy: 0.9143
Epoch 3989/4000
1/1 [==============================] - 0s 26ms/step - loss: 2.1793e-04 - accuracy: 1.0000 - val_loss: 0.4915 - val_accuracy: 0.9143
Epoch 3990/4000
1/1 [==============================] - 0s 23ms/step - loss: 2.1775e-04 - accuracy: 1.0000 - val_loss: 0.4915 - val_accuracy: 0.9143
Epoch 3991/4000
1/1 [==============================] - 0s 23ms/step - loss: 2.1757e-04 - accuracy: 1.0000 - val_loss: 0.4916 - val_accuracy: 0.9143
Epoch 3992/4000
1/1 [==============================] - 0s 21ms/step - loss: 2.1739e-04 - accuracy: 1.0000 - val_loss: 0.4917 - val_accuracy: 0.9143
Epoch 3993/4000
1/1 [==============================] - 0s 23ms/step - loss: 2.1723e-04 - accuracy: 1.0000 - val_loss: 0.4917 - val_accuracy: 0.9143
Epoch 3994/4000
1/1 [==============================] - 0s 21ms/step - loss: 2.1706e-04 - accuracy: 1.0000 - val_loss: 0.4918 - val_accuracy: 0.9143
Epoch 3995/4000
1/1 [==============================] - 0s 22ms/step - loss: 2.1687e-04 - accuracy: 1.0000 - val_loss: 0.4918 - val_accuracy: 0.9143
Epoch 3996/4000
1/1 [==============================] - 0s 26ms/step - loss: 2.1669e-04 - accuracy: 1.0000 - val_loss: 0.4918 - val_accuracy: 0.9143
Epoch 3997/4000
1/1 [==============================] - 0s 25ms/step - loss: 2.1651e-04 - accuracy: 1.0000 - val_loss: 0.4919 - val_accuracy: 0.9143
Epoch 3998/4000
1/1 [==============================] - 0s 24ms/step - loss: 2.1633e-04 - accuracy: 1.0000 - val_loss: 0.4920 - val_accuracy: 0.9143
Epoch 3999/4000
1/1 [==============================] - 0s 24ms/step - loss: 2.1615e-04 - accuracy: 1.0000 - val_loss: 0.4920 - val_accuracy: 0.9143
Epoch 4000/4000
1/1 [==============================] - 0s 24ms/step - loss: 2.1598e-04 - accuracy: 1.0000 - val_loss: 0.4921 - val_accuracy: 0.9143
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
500
1000
1500
2000
2500
3000
3500
4000
train
test

[B] Implementing L2 regularization.

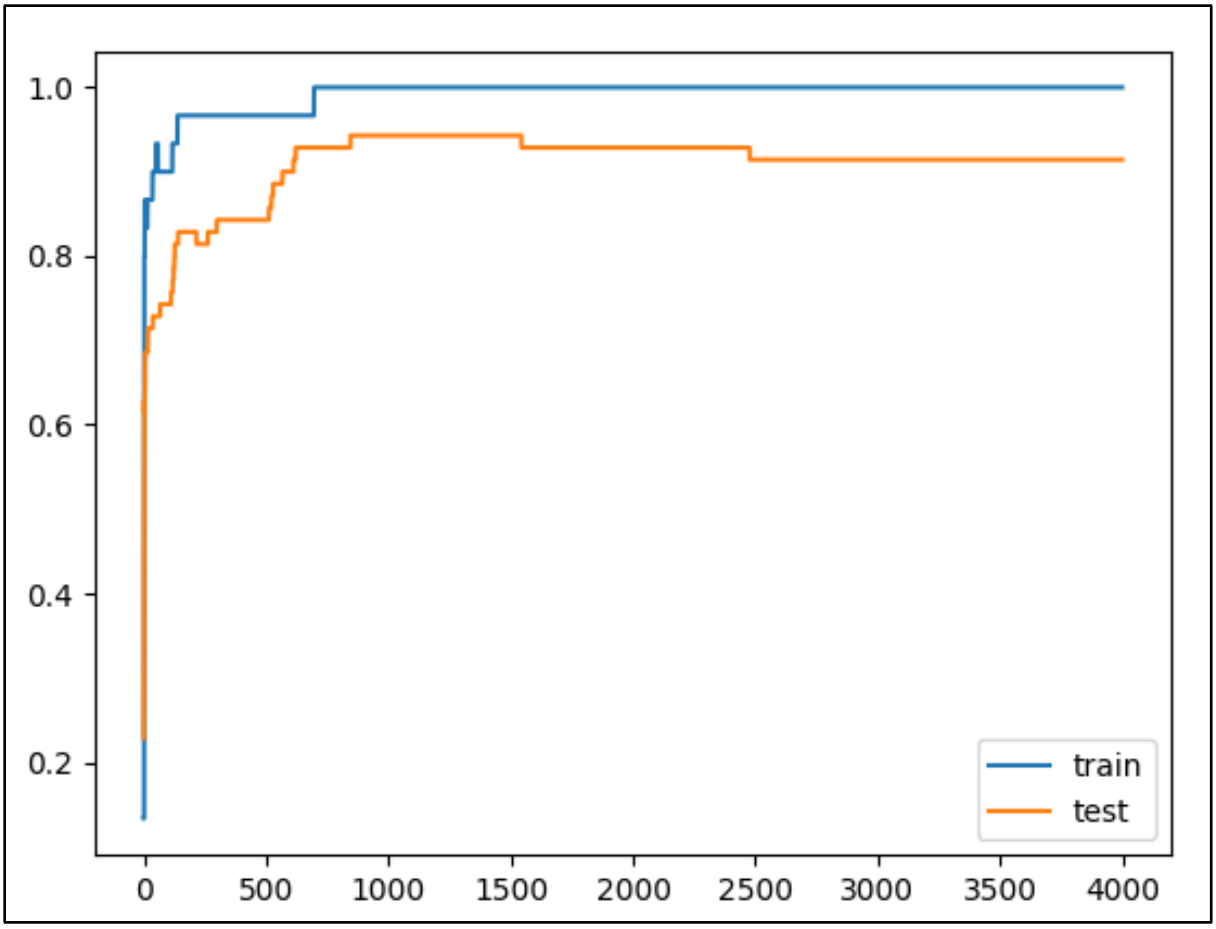
CODE:

```
from matplotlib import pyplot
from sklearn.datasets import make_moons
from keras.models import Sequential
from keras.layers import Dense
from keras.regularizers import l1_l2
X,Y=make_moons(n_samples=100,noise=0.2,random_state=1)
n_train=30
trainX,testX=X[:n_train,:],X[n_train:]
trainY,testY=Y[:n_train],Y[n_train:]
model= Sequential()
model.add(Dense(500,input_dim=2,activation='relu',kernel_regularizer=l1_l2(
l1=0.001,l2=0.001)))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
history=model.fit(trainX,trainY,validation_data=(testX,testY),epochs=4000)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test')
pyplot.legend()
pyplot.show()
```

OUTPUT:

```
Epoch 1/4000
1/1 [==============================] - 1s 514ms/step - loss: 0.7230 - accuracy: 0.1333 - val_loss: 0.7073 - val_accuracy: 0.2286
Epoch 2/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.7060 - accuracy: 0.1333 - val_loss: 0.6964 - val_accuracy: 0.6000
Epoch 3/4000
1/1 [==============================] - 0s 21ms/step - loss: 0.6894 - accuracy: 0.8000 - val_loss: 0.6858 - val_accuracy: 0.6286
Epoch 4/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.6733 - accuracy: 0.8000 - val_loss: 0.6756 - val_accuracy: 0.6143
Epoch 5/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.6576 - accuracy: 0.8667 - val_loss: 0.6657 - val_accuracy: 0.6286
Epoch 6/4000
1/1 [==============================] - 0s 22ms/step - loss: 0.6424 - accuracy: 0.8667 - val_loss: 0.6562 - val_accuracy: 0.6571
Epoch 7/4000
1/1 [==============================] - 0s 24ms/step - loss: 0.6276 - accuracy: 0.8333 - val_loss: 0.6470 - val_accuracy: 0.6857
Epoch 8/4000
1/1 [==============================] - 0s 22ms/step - loss: 0.6133 - accuracy: 0.8333 - val_loss: 0.6382 - val_accuracy: 0.6857
Epoch 9/4000
1/1 [==============================] - 0s 21ms/step - loss: 0.5993 - accuracy: 0.8333 - val_loss: 0.6297 - val_accuracy: 0.6857
Epoch 10/4000
1/1 [==============================] - 0s 21ms/step - loss: 0.5858 - accuracy: 0.8333 - val_loss: 0.6214 - val_accuracy: 0.6857
Epoch 11/4000
1/1 [==============================] - 0s 22ms/step - loss: 0.5726 - accuracy: 0.8333 - val_loss: 0.6135 - val_accuracy: 0.6857
Epoch 12/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.5598 - accuracy: 0.8333 - val_loss: 0.6059 - val_accuracy: 0.6857
```

```
Epoch 3989/4000
1/1 [==============================] - 0s 23ms/step - loss: 1.8910e-04 - accuracy: 1.0000 - val_loss: 0.4993 - val_accuracy: 0.9143
Epoch 3990/4000
1/1 [==============================] - 0s 23ms/step - loss: 1.8894e-04 - accuracy: 1.0000 - val_loss: 0.4993 - val_accuracy: 0.9143
Epoch 3991/4000
1/1 [==============================] - 0s 23ms/step - loss: 1.8880e-04 - accuracy: 1.0000 - val_loss: 0.4994 - val_accuracy: 0.9143
Epoch 3992/4000
1/1 [==============================] - 0s 30ms/step - loss: 1.8866e-04 - accuracy: 1.0000 - val_loss: 0.4994 - val_accuracy: 0.9143
Epoch 3993/4000
1/1 [==============================] - 0s 29ms/step - loss: 1.8851e-04 - accuracy: 1.0000 - val_loss: 0.4995 - val_accuracy: 0.9143
Epoch 3994/4000
1/1 [==============================] - 0s 29ms/step - loss: 1.8836e-04 - accuracy: 1.0000 - val_loss: 0.4995 - val_accuracy: 0.9143
Epoch 3995/4000
1/1 [==============================] - 0s 26ms/step - loss: 1.8821e-04 - accuracy: 1.0000 - val_loss: 0.4996 - val_accuracy: 0.9143
Epoch 3996/4000
1/1 [==============================] - 0s 24ms/step - loss: 1.8806e-04 - accuracy: 1.0000 - val_loss: 0.4996 - val_accuracy: 0.9143
Epoch 3997/4000
1/1 [==============================] - 0s 21ms/step - loss: 1.8792e-04 - accuracy: 1.0000 - val_loss: 0.4997 - val_accuracy: 0.9143
Epoch 3998/4000
1/1 [==============================] - 0s 22ms/step - loss: 1.8777e-04 - accuracy: 1.0000 - val_loss: 0.4997 - val_accuracy: 0.9143
Epoch 3999/4000
1/1 [==============================] - 0s 22ms/step - loss: 1.8762e-04 - accuracy: 1.0000 - val_loss: 0.4998 - val_accuracy: 0.9143
Epoch 4000/4000
1/1 [==============================] - 0s 28ms/step - loss: 1.8747e-04 - accuracy: 1.0000 - val_loss: 0.4998 - val_accuracy: 0.9143
```


1.0
0.8
0.6
0.4
0.2
0
500
1000
1500
2000
2500
3000
3500
4000
train
test

[C] Replacing L2 regularizer with L1 regularizer.

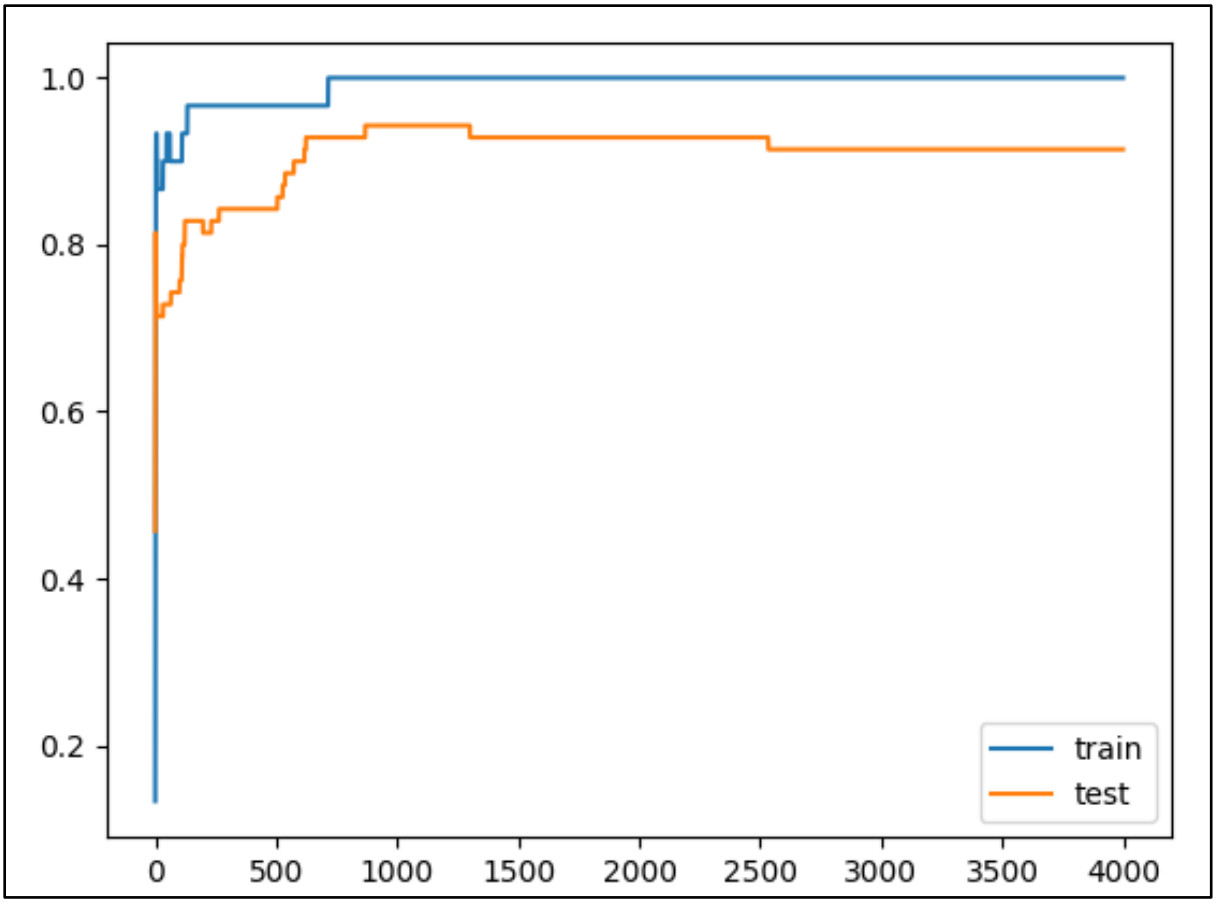
CODE:

```
from matplotlib import pyplot
from sklearn.datasets import make_moons
from keras.models import Sequential
from keras.layers import Dense
from keras.regularizers import l1_l2
X,Y=make_moons(n_samples=100,noise=0.2,random_state=1)
n_train=30
trainX,testX=X[:n_train,:],X[n_train:]
trainY,testY=Y[:n_train],Y[n_train:]
model= Sequential()
model.add(Dense(500,input_dim=2,activation='relu',kernel_regularizer=l1_l2(
l1=0.001,l2=0.001)))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
history=model.fit(trainX,trainY,validation_data=(testX,testY),epochs=4000)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test')
pyplot.legend()
pyplot.show()
```

OUTPUT:

```
Epoch 1/4000
1/1 [==============================] - 1s 513ms/step - loss: 0.7198 - accuracy: 0.1333 - val_loss: 0.6967 - val_accuracy: 0.4571
Epoch 2/4000
1/1 [==============================] - 0s 25ms/step - loss: 0.7030 - accuracy: 0.4333 - val_loss: 0.6858 - val_accuracy: 0.7286
Epoch 3/4000
1/1 [==============================] - 0s 25ms/step - loss: 0.6865 - accuracy: 0.6000 - val_loss: 0.6754 - val_accuracy: 0.8143
Epoch 4/4000
1/1 [==============================] - 0s 24ms/step - loss: 0.6705 - accuracy: 0.9000 - val_loss: 0.6653 - val_accuracy: 0.7429
Epoch 5/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.6550 - accuracy: 0.9333 - val_loss: 0.6557 - val_accuracy: 0.7286
Epoch 6/4000
1/1 [==============================] - 0s 25ms/step - loss: 0.6400 - accuracy: 0.9333 - val_loss: 0.6464 - val_accuracy: 0.7286
Epoch 7/4000
1/1 [==============================] - 0s 21ms/step - loss: 0.6254 - accuracy: 0.9333 - val_loss: 0.6375 - val_accuracy: 0.7286
Epoch 8/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.6113 - accuracy: 0.9000 - val_loss: 0.6289 - val_accuracy: 0.7286
Epoch 9/4000
1/1 [==============================] - 0s 22ms/step - loss: 0.5975 - accuracy: 0.9000 - val_loss: 0.6206 - val_accuracy: 0.7143
Epoch 10/4000
1/1 [==============================] - 0s 23ms/step - loss: 0.5841 - accuracy: 0.9000 - val_loss: 0.6127 - val_accuracy: 0.7143
Epoch 11/4000
1/1 [==============================] - 0s 21ms/step - loss: 0.5711 - accuracy: 0.8667 - val_loss: 0.6050 - val_accuracy: 0.7143
```

```
Epoch 3990/4000
1/1 [==============================] - 0s 21ms/step - loss: 2.0657e-04 - accuracy: 1.0000 - val_loss: 0.4963 - val_accuracy: 0.9143
Epoch 3991/4000
1/1 [==============================] - 0s 28ms/step - loss: 2.0639e-04 - accuracy: 1.0000 - val_loss: 0.4963 - val_accuracy: 0.9143
Epoch 3992/4000
1/1 [==============================] - 0s 29ms/step - loss: 2.0623e-04 - accuracy: 1.0000 - val_loss: 0.4964 - val_accuracy: 0.9143
Epoch 3993/4000
1/1 [==============================] - 0s 31ms/step - loss: 2.0606e-04 - accuracy: 1.0000 - val_loss: 0.4964 - val_accuracy: 0.9143
Epoch 3994/4000
1/1 [==============================] - 0s 27ms/step - loss: 2.0589e-04 - accuracy: 1.0000 - val_loss: 0.4965 - val_accuracy: 0.9143
Epoch 3995/4000
1/1 [==============================] - 0s 23ms/step - loss: 2.0572e-04 - accuracy: 1.0000 - val_loss: 0.4965 - val_accuracy: 0.9143
Epoch 3996/4000
1/1 [==============================] - 0s 22ms/step - loss: 2.0557e-04 - accuracy: 1.0000 - val_loss: 0.4966 - val_accuracy: 0.9143
Epoch 3997/4000
1/1 [==============================] - 0s 21ms/step - loss: 2.0540e-04 - accuracy: 1.0000 - val_loss: 0.4966 - val_accuracy: 0.9143
Epoch 3998/4000
1/1 [==============================] - 0s 22ms/step - loss: 2.0523e-04 - accuracy: 1.0000 - val_loss: 0.4967 - val_accuracy: 0.9143
Epoch 3999/4000
1/1 [==============================] - 0s 29ms/step - loss: 2.0507e-04 - accuracy: 1.0000 - val_loss: 0.4968 - val_accuracy: 0.9143
Epoch 4000/4000
1/1 [==============================] - 0s 29ms/step - loss: 2.0489e-04 - accuracy: 1.0000 - val_loss: 0.4968 - val_accuracy: 0.9143
```


1.0
0.8
0.6
0.4
0.2
0
500
1000
1500
2000
2500
3000
3500
4000
train
test

PRACTICAL 7

AIM: Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from sklearn.preprocessing import MinMaxScaler
dataset_train=pd.read_csv('Google_Stock_Price_Train.csv')
#print(dataset_train)
training_set=dataset_train.iloc[:,1:2].values
print(training_set)
sc=MinMaxScaler(feature_range=(0,1))
training_set_scaled=sc.fit_transform(training_set)
print(training_set_scaled)
X_train=[]
Y_train=[]
for i in range(60,1258):
    X_train.append(training_set_scaled[i-60:i,0])
    Y_train.append(training_set_scaled[i,0])
X_train,Y_train=np.array(X_train),np.array(Y_train)
print(X_train)
print('*****************************************')
print(Y_train)
X_train=np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
print('*****************************************')
print(X_train)
regressor=Sequential()
regressor.add(LSTM(units=50,return_sequences=True,input_shape=(X_train.shap
e[1],1)))
regressor.add(Dropout(0.2))
```

```
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))
regressor.add(Dense(units=1))
regressor.compile(optimizer='adam',loss='mean_squared_error')
regressor.fit(X_train,Y_train,epochs=100,batch_size=32)
dataset_test=pd.read_csv('Google_Stock_Price_Test.csv')
real_stock_price=dataset_test.iloc[:,1:2].values
dataset_total=pd.concat((dataset_train['Open'],dataset_test['Open']),axis=0
)
inputs=dataset_total[len(dataset_total)-len(dataset_test)-60:].values
inputs=inputs.reshape(-1,1)
inputs=sc.transform(inputs)
X_test=[]
for i in range(60,80):
    X_test.append(inputs[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
predicted_stock_price=regressor.predict(X_test)
predicted_stock_price=sc.inverse_transform(predicted_stock_price)
plt.plot(real_stock_price,color='red',label='real google stock price')
plt.plot(predicted_stock_price,color='blue',label='predicted stock price')
plt.xlabel('time')
plt.ylabel('google stock price')
plt.legend()
plt.show()
```

OUTPUT:

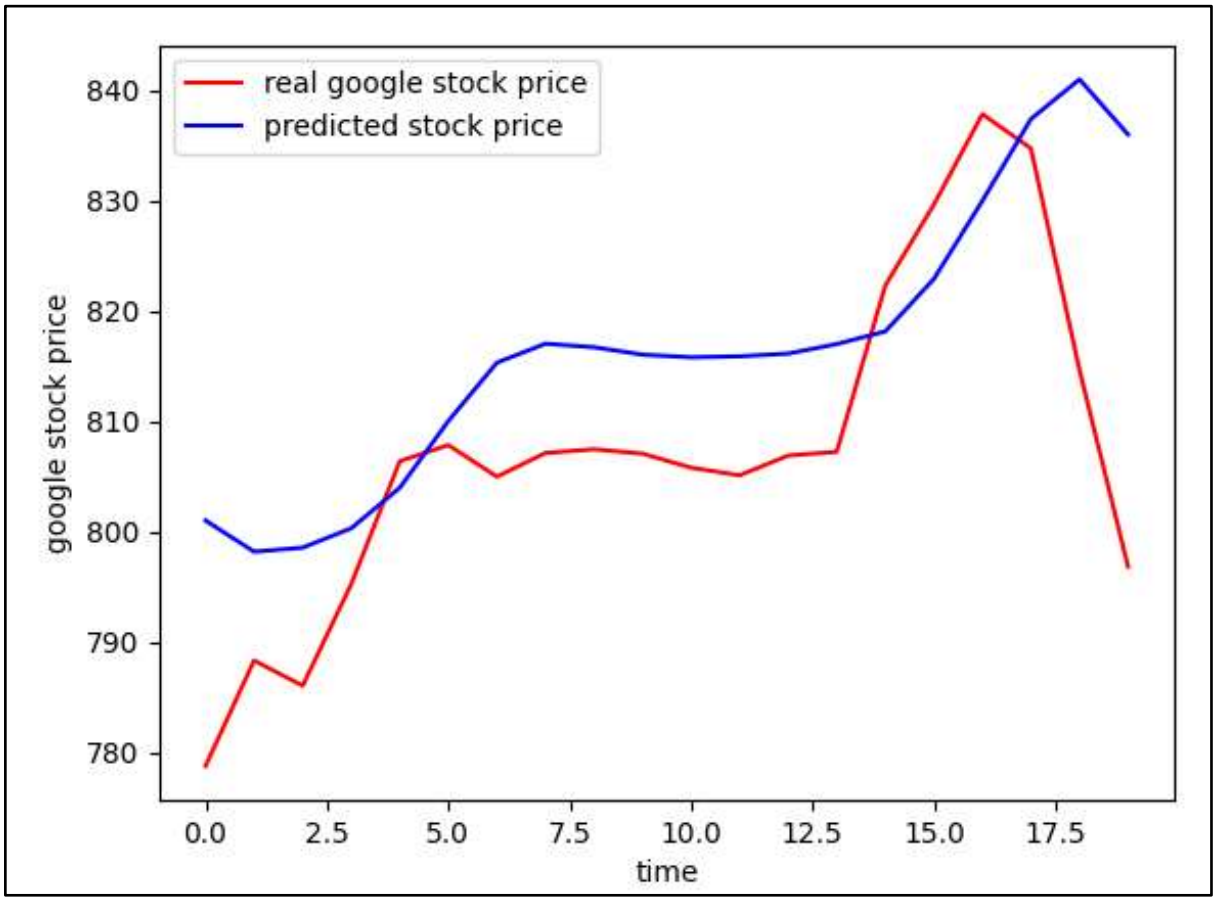
```
[[[0.08581368]
  [0.09701243]
  [0.09433366]
  ...
  [0.07846566]
  [0.08034452]
  [0.08497656]]

 [[0.09701243]
  [0.09433366]
  [0.09156187]
  ...
  [0.08034452]
  [0.08497656]
  [0.08627874]]

 [[0.09433366]
  [0.09156187]
  [0.07984225]
  ...
  [0.08497656]
  [0.08627874]
  [0.08471612]]

 ...
```

```
 [[0.92106928]
  [0.92438053]
  [0.93048218]
  ...
  [0.95475854]
  [0.95204256]
  [0.95163331]]

 [[0.92438053]
  [0.93048218]
  [0.9299055 ]
  ...
  [0.95204256]
  [0.95163331]
  [0.95725128]]

 [[0.93048218]
  [0.9299055 ]
  [0.93113327]
  ...
  [0.95163331]
  [0.95725128]
  [0.93796041]]]
```

```
[[325.25]
 [331.27]
 [329.83]
 ...
 [793.7 ]
 [783.33]
 [782.75]]
[[0.08581368]
 [0.09701243]
 [0.09433366]
 ...
 [0.95725128]
 [0.93796041]
 [0.93688146]]
[[0.08581368 0.09701243 0.09433366 ... 0.07846566 0.08034452 0.08497656]
 [0.09701243 0.09433366 0.09156187 ... 0.08034452 0.08497656 0.08627874]
 [0.09433366 0.09156187 0.07984225 ... 0.08497656 0.08627874 0.08471612]
 ...
 [0.92106928 0.92438053 0.93048218 ... 0.95475854 0.95204256 0.95163331]
 [0.92438053 0.93048218 0.9299055  ... 0.95204256 0.95163331 0.95725128]
 [0.93048218 0.9299055  0.93113327 ... 0.95163331 0.95725128 0.93796041]]
**********************************************
[0.08627874 0.08471612 0.07454052 ... 0.95725128 0.93796041 0.93688146]
**********************************************
```

```
Epoch 1/100
38/38 [==============================] - 6s 48ms/step - loss: 0.0517
Epoch 2/100
38/38 [==============================] - 2s 48ms/step - loss: 0.0068
Epoch 3/100
38/38 [==============================] - 2s 48ms/step - loss: 0.0058
Epoch 4/100
38/38 [==============================] - 2s 48ms/step - loss: 0.0050
Epoch 5/100
38/38 [==============================] - 2s 47ms/step - loss: 0.0046
Epoch 6/100
38/38 [==============================] - 2s 47ms/step - loss: 0.0048
```

```
38/38 [==============================] - 2s 48ms/step - loss: 0.0016
Epoch 97/100
38/38 [==============================] - 2s 49ms/step - loss: 0.0015
Epoch 98/100
38/38 [==============================] - 2s 48ms/step - loss: 0.0016
Epoch 99/100
38/38 [==============================] - 2s 49ms/step - loss: 0.0015
Epoch 100/100
38/38 [==============================] - 2s 48ms/step - loss: 0.0015
1/1 [==============================] - 1s 983ms/step
```



PRACTICAL 8

AIM: Performing encoding and decoding of images using deep autoencoder.

CODE:

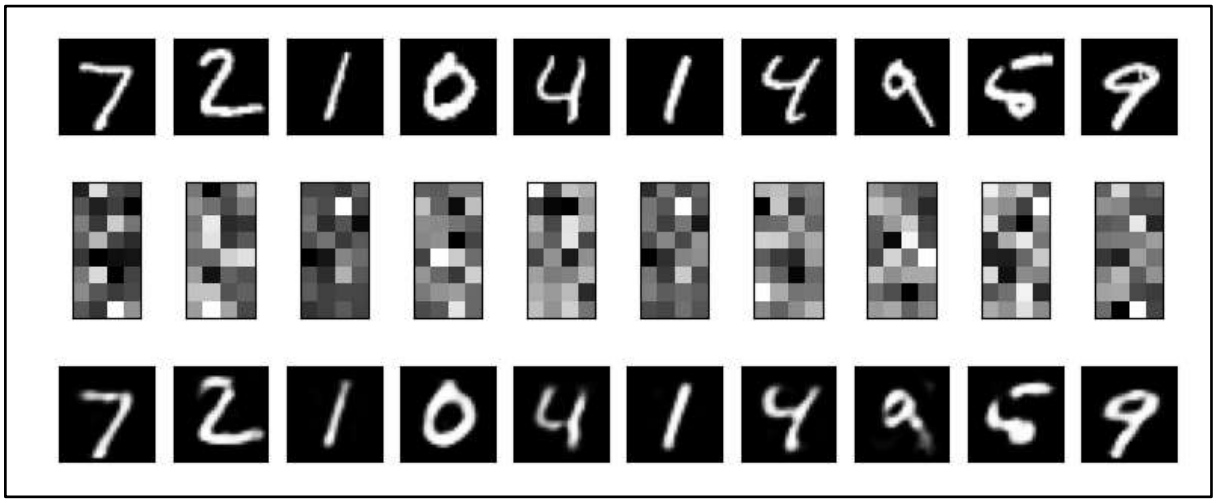
```
import keras
from keras import layers
from keras.datasets import mnist
import numpy as np
encoding_dim=32
#this is our input image
input_img=keras.Input(shape=(784,))
#"encoded" is the encoded representation of the input
encoded=layers.Dense(encoding_dim, activation='relu')(input_img)
#"decoded" is the lossy reconstruction of the input
decoded=layers.Dense(784, activation='sigmoid')(encoded)
#creating autoencoder model
autoencoder=keras.Model(input_img,decoded)
#create the encoder model
encoder=keras.Model(input_img,encoded)
encoded_input=keras.Input(shape=(encoding_dim,))
#Retrive the last layer of the autoencoder model
decoder_layer=autoencoder.layers[-1]
#create the decoder model
decoder=keras.Model(encoded_input,decoder_layer(encoded_input))
autoencoder.compile(optimizer='adam',loss='binary_crossentropy')
#scale and make train and test dataset
(X_train,_),(X_test,_)=mnist.load_data()
X_train=X_train.astype('float32')/255.
X_test=X_test.astype('float32')/255.
X_train=X_train.reshape((len(X_train),np.prod(X_train.shape[1:])))
X_test=X_test.reshape((len(X_test),np.prod(X_test.shape[1:])))
print(X_train.shape)
print(X_test.shape)
#train autoencoder with training dataset
autoencoder.fit(X_train,X_train,  epochs=50,  batch_size=256,  shuffle=True,
validation_data=(X_test,X_test))
```

```
encoded_imgs=encoder.predict(X_test)
decoded_imgs=decoder.predict(encoded_imgs)
import matplotlib.pyplot as plt
n = 10 # How many digits we will display
plt.figure(figsize=(40, 4))
for i in range(10):
# display original
    ax = plt.subplot(3, 20, i + 1)
    plt.imshow(X_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    # display encoded image
    ax = plt.subplot(3, 20, i + 1 + 20)
    plt.imshow(encoded_imgs[i].reshape(8, 4))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    # display reconstruction
    ax = plt.subplot(3, 20, 2 * 20 + i + 1)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

OUTPUT:

```
11490434/11490434 [==============================] - 1s 0us/step
(60000, 784)
(10000, 784)
Epoch 1/50
235/235 [==============================] - 1s 4ms/step - loss: 0.2784 - val_loss: 0.1932
Epoch 2/50
235/235 [==============================] - 1s 3ms/step - loss: 0.1722 - val_loss: 0.1533
Epoch 3/50
235/235 [==============================] - 1s 3ms/step - loss: 0.1434 - val_loss: 0.1330
Epoch 4/50
235/235 [==============================] - 1s 3ms/step - loss: 0.1274 - val_loss: 0.1199
Epoch 5/50
235/235 [==============================] - 1s 3ms/step - loss: 0.1170 - val_loss: 0.1114
Epoch 6/50
235/235 [==============================] - 1s 3ms/step - loss: 0.1101 - val_loss: 0.1059
Epoch 7/50
235/235 [==============================] - 1s 3ms/step - loss: 0.1054 - val_loss: 0.1020
```

```
235/235 [==============================] - 1s 3ms/step - loss: 0.0927 - val_loss: 0.0916
Epoch 48/50
235/235 [==============================] - 1s 3ms/step - loss: 0.0927 - val_loss: 0.0916
Epoch 49/50
235/235 [==============================] - 1s 3ms/step - loss: 0.0927 - val_loss: 0.0916
Epoch 50/50
235/235 [==============================] - 1s 3ms/step - loss: 0.0927 - val_loss: 0.0916
313/313 [==============================] - 0s 551us/step
313/313 [==============================] - 0s 601us/step
```



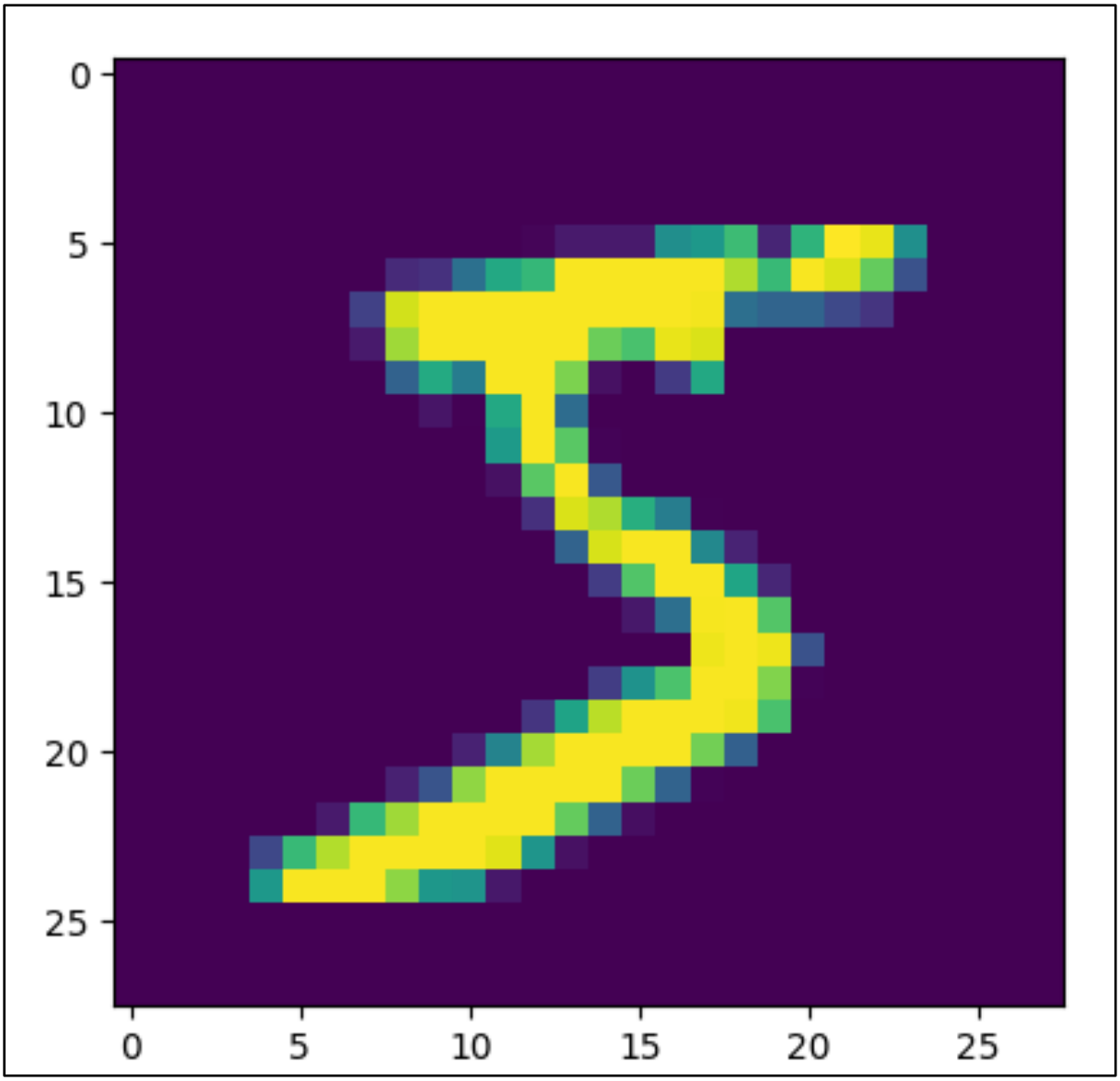
PRACTICAL 9

AIM: Implementation of convolutional neural network to predict numbers from number images.

CODE:

```
from keras.datasets import mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Conv2D,Flatten
import matplotlib.pyplot as plt
#download mnist data and split into train and test sets
(X_train,Y_train),(X_test,Y_test)=mnist.load_data()
#plot the first image in the dataset
plt.imshow(X_train[0])
plt.show()
print(X_train[0].shape)
X_train=X_train.reshape(60000,28,28,1)
X_test=X_test.reshape(10000,28,28,1)
Y_train=to_categorical(Y_train)
Y_test=to_categorical(Y_test)
Y_train[0]
print(Y_train[0])
model=Sequential()
#add model layers
#learn image features
model.add(Conv2D(64,kernel_size=3,activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=3,activation='relu'))
model.add(Flatten())
model.add(Dense(10,activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['ac
curacy'])
#train
model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=3)
print(model.predict(X_test[:4]))
#actual results for 1st 4 images in the test set
print(Y_test[:4])
```

OUTPUT:




```
(28, 28)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
Epoch 1/3
1875/1875 [==============================] - 54s 29ms/step - loss: 0.2302 - accuracy: 0.9516 - val_loss: 0.1051 - val_accuracy: 0.9669
Epoch 2/3
1875/1875 [==============================] - 48s 26ms/step - loss: 0.0699 - accuracy: 0.9786 - val_loss: 0.0744 - val_accuracy: 0.9782
Epoch 3/3
1875/1875 [==============================] - 48s 26ms/step - loss: 0.0501 - accuracy: 0.9841 - val_loss: 0.0953 - val_accuracy: 0.9729
1/1 [==============================] - 0s 54ms/step
```

```
[[1.28678295e-08 1.60042327e-11 4.08504593e-06 1.85614681e-05
  4.91933751e-11 1.62152708e-10 5.10676401e-14 9.99967337e-01
  8.30323734e-06 1.70347710e-06]
 [8.40842915e-07 2.47974286e-09 9.99998212e-01 2.44631337e-09
  5.72847195e-12 3.67103084e-11 9.84666372e-07 7.65504793e-13
  5.66962228e-08 5.52745458e-13]
 [2.66522898e-06 9.97940004e-01 3.96974838e-06 1.76509811e-08
  1.96918356e-03 2.62945696e-06 1.83274778e-06 5.80798678e-06
  7.38622475e-05 6.91403415e-08]
 [9.99975204e-01 2.32659465e-12 2.21851951e-05 8.21377133e-11
  1.66345146e-10 2.17179590e-08 3.32290639e-08 4.78619835e-12
  1.18910428e-08 2.57370039e-06]]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

PRACTICAL 10

AIM: Denoising of images using autoencoder.

CODE:

```
import keras
from keras.datasets import mnist
from keras import layers
import numpy as np
from keras.callbacks import TensorBoard
import matplotlib.pyplot as plt
(X_train,_),(X_test,_)=mnist.load_data()
X_train=X_train.astype('float32')/255.
X_test=X_test.astype('float32')/255.
X_train=np.reshape(X_train,(len(X_train),28,28,1))
X_test=np.reshape(X_test,(len(X_test),28,28,1))
noise_factor=0.5
X_train_noisy=X_train+noise_factor*np.random.normal(loc=0.0,scale=1.0,size=
X_train.shape)
X_test_noisy=X_test+noise_factor*np.random.normal(loc=0.0,scale=1.0,size=X_
test.shape)
X_train_noisy=np.clip(X_train_noisy,0.,1.)
X_test_noisy=np.clip(X_test_noisy,0.,1.)
n=10
plt.figure(figsize=(20,2))
for i in range(1,n+1):
 ax=plt.subplot(1,n,i)
 plt.imshow(X_test_noisy[i].reshape(28,28))
 plt.gray()
 ax.get_xaxis().set_visible(False)
 ax.get_yaxis().set_visible(False)
plt.show()
input_img=keras.Input(shape=(28,28,1))
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(input_img)
x=layers.MaxPooling2D((2,2),padding='same')(x)
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x)
encoded=layers.MaxPooling2D((2,2),padding='same')(x)
```

```
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(encoded)
x=layers.UpSampling2D((2,2))(x)
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x)
x=layers.UpSampling2D((2,2))(x)
decoded=layers.Conv2D(1,(3,3),activation='sigmoid',padding='same')(x)
autoencoder=keras.Model(input_img,decoded)
autoencoder.compile(optimizer='adam',loss='binary_crossentropy')
autoencoder.fit(X_train_noisy,X_train,
 epochs=3,
 batch_size=128,
 shuffle=True,
 validation_data=(X_test_noisy,X_test),
callbacks=[TensorBoard(log_dir='/tmo/tb',histogram_freq=0,write_graph=False
)])
predictions=autoencoder.predict(X_test_noisy)
m=10
plt.figure(figsize=(20,2))
for i in range(1,m+1):
 ax=plt.subplot(1,m,i)
 plt.imshow(predictions[i].reshape(28,28))
 plt.gray()
 ax.get_xaxis().set_visible(False)
 ax.get_yaxis().set_visible(False)
plt.show()
```

OUTPUT:

```
Epoch 1/3
469/469 [==============================] - 48s 101ms/step - loss: 0.1625 - val_loss: 0.1183
Epoch 2/3
469/469 [==============================] - 48s 101ms/step - loss: 0.1152 - val_loss: 0.1102
Epoch 3/3
469/469 [==============================] - 48s 102ms/step - loss: 0.1094 - val_loss: 0.1064
313/313 [==============================] - 2s 6ms/step
```