Practical 1

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

Code:

```
import Crypto
import binascii
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
class Client:
    def __init__(self):
        random = Crypto.Random.new().read

        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

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

Shivam = Client()
print("\nPublic Key:",Shivam.identity)
```

Output:

```
C:\Users\schau\Music\blockchainprac>python client.py

Public Key: 30819f300d06092a864886f70d010101050003818d0030818902818100d83e162ba97c76a643921c6e88b65f322d9f832915d86bbb5e
26e450ccd752e23e9731c482be85c8cfdd8f7703a7488c1da4017a24bb55516b13dd509ef8bb1cbeac7684277ef1cbaa8566c0ab833d1d5798d6ef8c
7023b5613e99cf35fc9d510162245f66309b2f2558cbed60700e38eaeda5b75936fcc8a17f6d9abd5e9dd10203010001
```

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

Code:

```
#1B.- A transaction class to send and receive money and test it.
import Crypto
import binascii
import datetime
import collections
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA
class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)
```

```
    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')
class Transaction:
    def __init__(self, sender, receiver, value):
        self.sender = sender
        self.receiver = receiver
        self.value = value
        self.time = datetime.datetime.now()
    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
        return collections.OrderedDict({
            'sender': identity,
            'receiver': self.receiver,
            'value': self.value,
            'time': self.time
        })
    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')
Shivam = Client()
print("-"*50)
print("Shivam Key")
print(Shivam.identity)
Rahul = Client()
print("-"*50)
print("Rahul Key")
print(Rahul.identity)

t = Transaction(Rahul, Shivam.identity, 10.0)
print("-"*50)
print("Transaction Sign")
signature = t.sign_transaction()
print(signature)
print("-"*50)
```

Output:

```
C:\Users\schau\Music\blockchainprac>python Transaction.py
--------------------------------------------------
Shivam Key
30819f300d06092a864886f70d010101050003818d0030818902818100c5073ce12c1464deb55375a356ee196482945d8407daca1b3e9db72ea204b
f4c8c0243cd0063a20aad48d5c2cadb337f82833ab95bd9929afa11ed15f764be8b9700dfaf70cf1db463d480f39674915da55f85a418ea2a71ea62e
757d63c19fea62ae8f68818d2733005d8fd5da8c259a6f19902153aa1ab7e65fedae5ea2bb0203010001
--------------------------------------------------
Rahul Key
30819f300d06092a864886f70d010101050003818d0030818902818100bfd3a54ae3e1e9a520c3bae762811d2c355f8d6b19ef69fd33fb9c4785b9e0
9f48fee35924053cec700db1f30f4dc5b44f7140795d76c8b035f3c14921977173b70e58d9785b10ca592cdf45e5a3db5c20e520061ec15f85c6bb32
501ef36dad153f33f6cb729a1b2e7cc970beec1a6ce4465bab52f339796c0dd5183a93205f0203010001
--------------------------------------------------
Transaction Sign
24762ca62fd3c481b8eb80d1d1f9500e98ae94d8d5c75537021ea0cf570acadfe55a94064ddcc648527ab794d3b329fbb793e6ca45d98012142855777
16b5d2568b7133c11dbb23ad8a0adfb2f057b4f7a80baecb343bf86c87398a03c254b63e9d913fbb57db56ddc1c1c4fed9a36f30a3694b5753ee5791
1ef993b22fdb5cba
```

C. Create multiple transactions and display them.

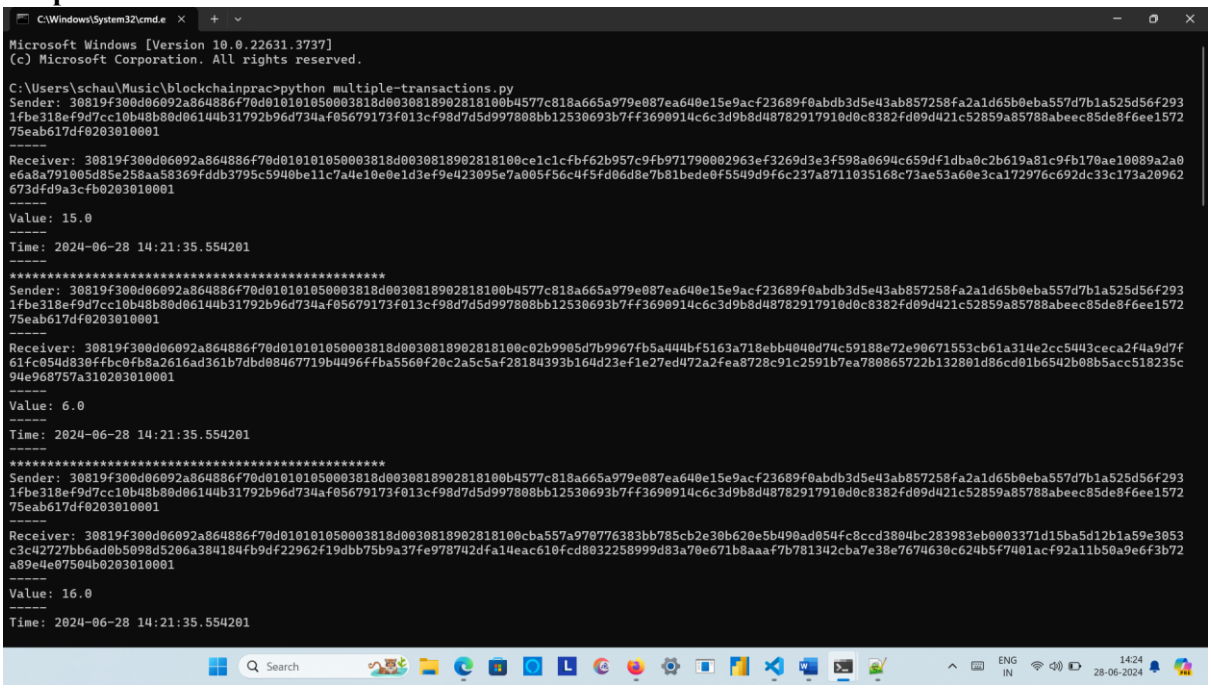
Code:

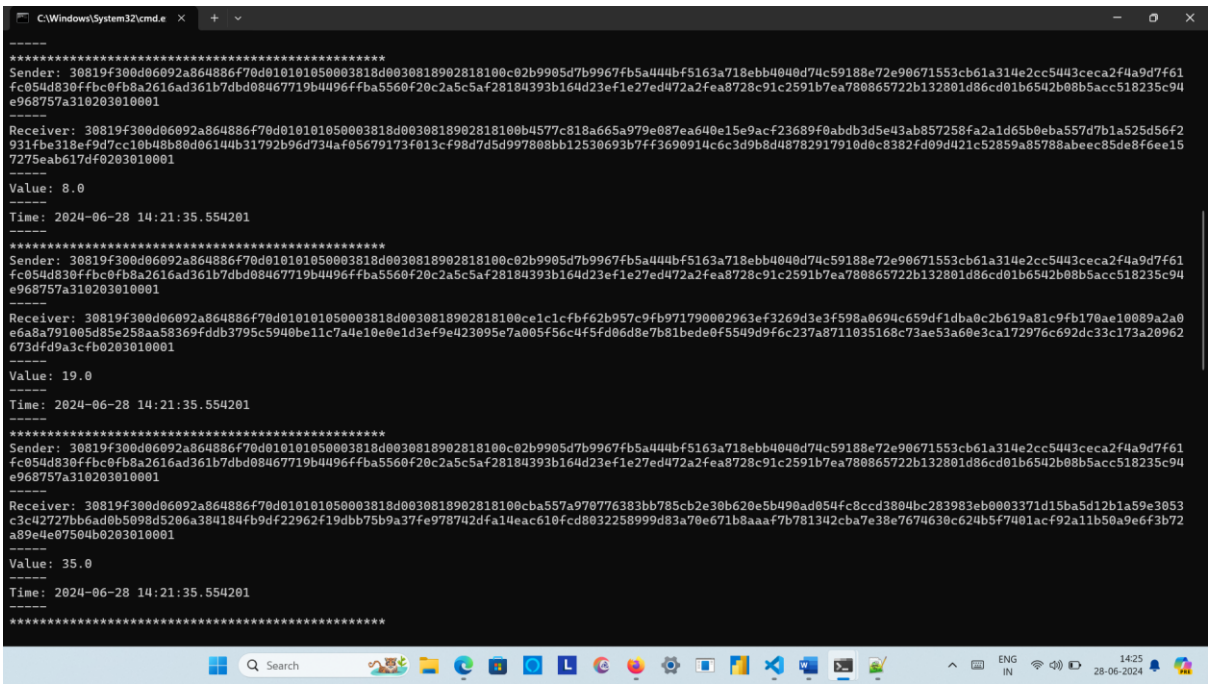
```
import Crypto
import binascii
import datetime
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
import hashlib
from hashlib import sha256
class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)
    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')
# Demo = Client()
# print(Demo.identity)
import datetime
import collections
import binascii
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA
class Transaction:
  def __init__(self, sender, receiver, value):
      self.sender = sender
      self.receiver = receiver
      self.value = value
      self.time = datetime.datetime.now()
  def to_dict(self):
      if self.sender == "Genesis":
          identity = "Genesis"
      else:
          identity = self.sender.identity
      return collections.OrderedDict({
          'sender': identity,
          'receiver': self.receiver,
          'value': self.value,
          'time': self.time
      })
  def sign_transaction(self):
      private_key = self.sender._private_key
      signer = PKCS1_v1_5.new(private_key)
      h = SHA.new(str(self.to_dict()).encode('utf8'))
      return binascii.hexlify(signer.sign(h)).decode('ascii')
```

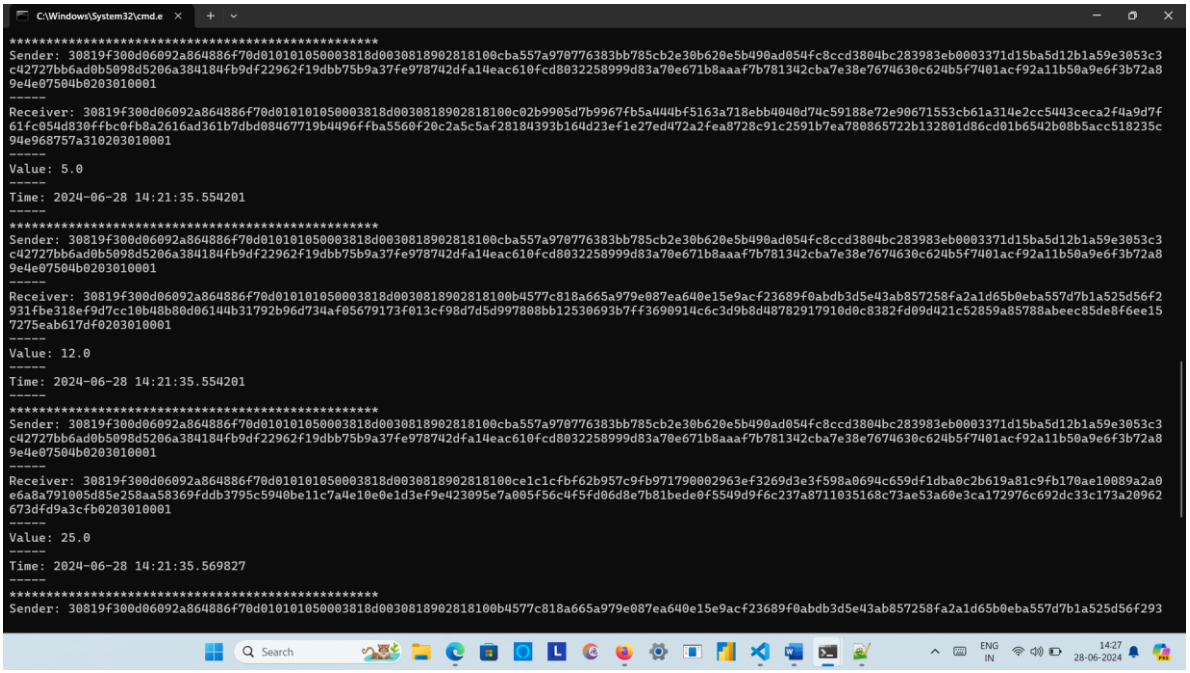
```
import hashlib
def sha256(message):
  return hashlib.sha256(message.encode('ascii')).hexdigest
def mine(message, difficulty=1):
  assert difficulty >=1
  prefix = '1' * difficulty
  for i in range(1000):
    digest = sha256(str(hash(message)) + str(i))
  if digest.startwith(prefix):
    print("after" + str(i) + "iteration found nonce:" + digest)
    return digest
class Block:
  def __init__(self):
    self.verified_transactions = []
    self.previous_block_hash = ""
    self.Nonce = ""
  last_block_hash = ""
  def display_transaction(transaction):
      dict = transaction.to_dict()
      print("Sender: " + dict['sender'])
      print('-----')
      print("Receiver: " + dict['receiver'])  # Corrected typo
      print('-----')
      print("Value: " + str(dict['value']))
      print('-----')
      print("Time: " + str(dict['time']))
      print('-----')
  TPCoins = []
  def dump_blockchain(self):
    print("Number of blocks in chain" + str(len(self)))
    for x in range(len(Block.TPCoins)):
      block_temp = Block.TPCoins[x]
      print("block #" + str(x))
      for transaction in block_temp.verified_transactions:
        Block.display_transaction(transaction)
        print("-------")
  last_transaction_index = 0
  transactions = []
  Ninad = Client()
  ks = Client()
  vighnesh = Client()
  sairaj = Client()
  t1 = Transaction(Ninad , ks.identity , 15.0)
  t1.sign_transaction()
  transactions.append(t1)
  t2 = Transaction(Ninad , vighnesh.identity , 6.0)
  t2.sign_transaction()
  transactions.append(t2)
  t3 = Transaction(Ninad , sairaj.identity , 16.0)
  t3.sign_transaction()
  transactions.append(t3)
```

```
t4 = Transaction(vighnesh , Ninad.identity , 8.0)
t4.sign_transaction()
transactions.append(t4)
t5 = Transaction(vighnesh , ks.identity , 19.0)
t5.sign_transaction()
transactions.append(t5)
t6 = Transaction(vighnesh , sairaj.identity , 35.0)
t6.sign_transaction()
transactions.append(t6)
t7 = Transaction(sairaj , vighnesh.identity , 5.0)
t7.sign_transaction()
transactions.append(t7)
t8 = Transaction(sairaj , Ninad.identity , 12.0)
t8.sign_transaction()
transactions.append(t8)
t9 = Transaction(sairaj , ks.identity , 25.0)
t9.sign_transaction()
transactions.append(t9)
t10 = Transaction(Ninad , ks.identity , 1.0)
t10.sign_transaction()
transactions.append(t10)
for transaction in transactions:
  display_transaction(transaction)
  print("*"*50)
```

Output:







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

Code:

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

```
from Crypto.Signature import PKCS1_v1_5
import hashlib
from hashlib import sha256
class Client:
    def __init__(self):

        random = Crypto.Random.new().read
        # Creating new public key and private key
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')
class Transaction:
  def __init__(self, sender, receiver, value):
      self.sender = sender
      self.receiver = receiver
      self.value = value
      self.time = datetime.datetime.now()

  def to_dict(self):
      if self.sender == "Genesis":
          identity = "Genesis"
      else:
          identity = self.sender.identity

      return collections.OrderedDict({
          'sender': identity,
          'receiver': self.receiver,
          'value': self.value,
          'time': self.time
      })

  def sign_transaction(self):
      private_key = self.sender._private_key
      signer = PKCS1_v1_5.new(private_key)
      h = SHA.new(str(self.to_dict()).encode('utf8'))
      return binascii.hexlify(signer.sign(h)).decode('ascii')
class Block:
  def __init__(self):
    self.verified_transactions = []
    self.previous_block_hash = ""
    self.Nonce = ""

  last_block_hash = ""

  def display_transaction(transaction):
      dict = transaction.to_dict()
      print("Sender: " + dict['sender'])
```

```
        print('-----')
        print("Receiver: " + dict['receiver'])  # Corrected typo
        print('-----')
        print("Value: " + str(dict['value']))
        print('-----')
        print("Time: " + str(dict['time']))
        print('-----')
Ninad = Client()
t0 = Transaction(
    "Genesis",
    Ninad.identity,
    500.0
)
block0 = Block()
block0.previous_block_hash = None
Nonce = None
block0.verified_transactions.append(t0)
digest = hash(block0)
last_block_hash = digest
TPCoins = []
def dump_blockchain(self):
  print("Number of blocks in chain: "+ str(len(self)))
  for x in range(len(TPCoins)):
    block_temp = TPCoins[x]
    print("block #" + str(x))
  for transaction in block_temp.verified_transactions:
    Block.display_transaction(transaction)
    print('-'*20)
  print("="*30)
TPCoins.append(block0)
dump_blockchain(TPCoins)
```

Output:

```
C:\Users\schau\Music\blockchainprac>python genesis-block
Number of blocks in chain: 1
block #0
Sender: Genesis
-----
Receiver: 30819f300d06092a864886f70d010101050003818d0030818902818100a3cf9b6461c5d987fd8359780024f3e289ed9ca3c6ac1c0738df
5ed41339419e79a8bffc25f981a9f8d9741ab00826a309f8112a51748538070a753e4149aec434bf3069eddeabba55d96204a4118c853f509467b100
1e7e2c2fa6a6d0e672a1814da8151cd78b7a34f6ccf8d7b6118dad6d777ef74956cc2b715c60049b8d2d0203010001
-----
Value: 500.0
-----
Time: 2024-06-28 14:33:52.075285
```

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):
    prefix = "1" * difficulty
    for i in range(1000):
        digest = hashlib.sha256(f"{message} {i}".encode()).hexdigest()
```

```
        if digest.startswith(prefix):
            print(f"after {str(i)} iterations found nonce: {digest}")
            break
mine("test message", 2)
```

Output:

```
C:\Users\schau\Music\blockchainprac>python Mining-function.py
after 52 iterations found nonce: 11f31955281b4afe3e769d508bb32cbac4f2f61689487710965aa2c609dff4bf
```

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" )
```

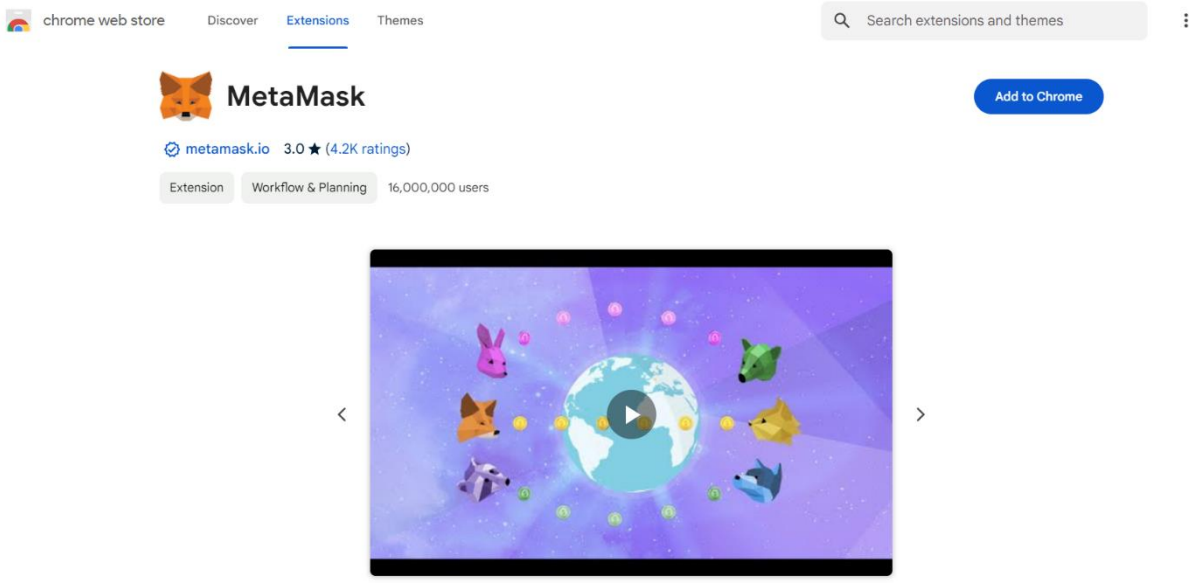
Output:

```
C:\Users\schau\Music\blockchainprac>python Add-Block.py
Timestamp: 2024-06-28 09:13:42.792874+00:00
Data: First block
Previous Hash: 0
Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9

Timestamp: 2024-06-28 09:13:42.792874+00:00
Data: Second block
Previous Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9
Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd

Timestamp: 2024-06-28 09:13:42.792874+00:00
Data: Third block
Previous Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd
Hash: 06e369fbfbe5362a8115a5c6f3e2d3ec7292cc4272052dcc3280898e3206208d
```

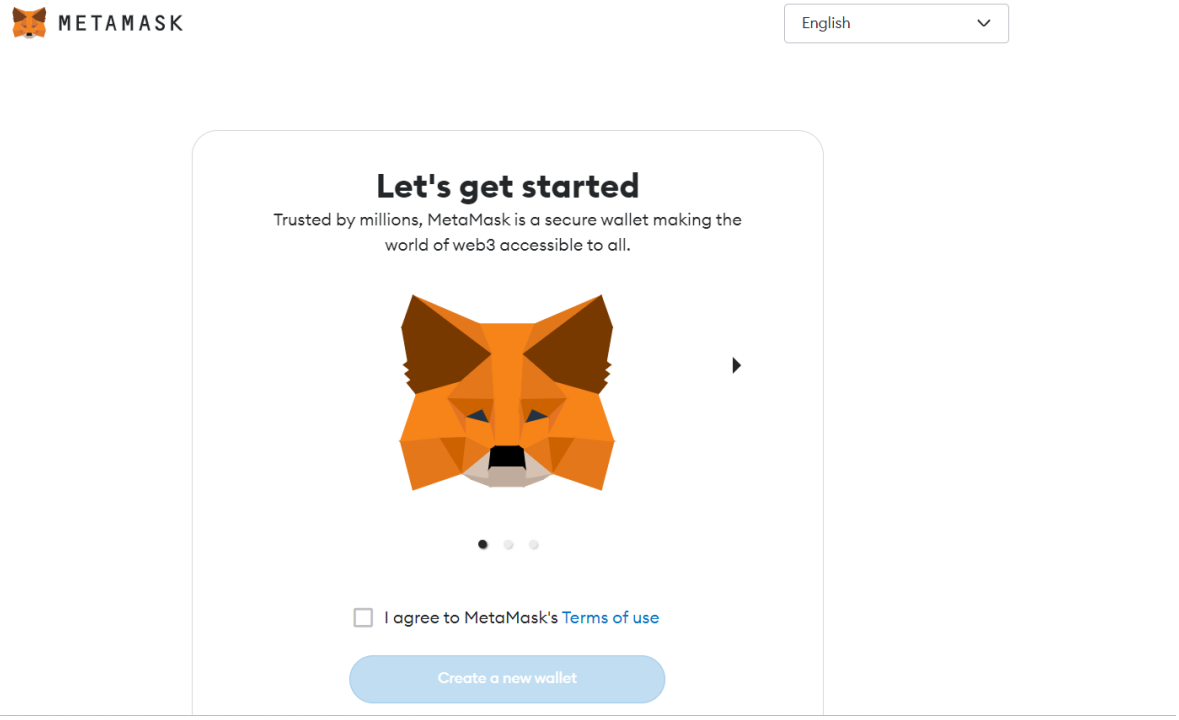
Practical 2

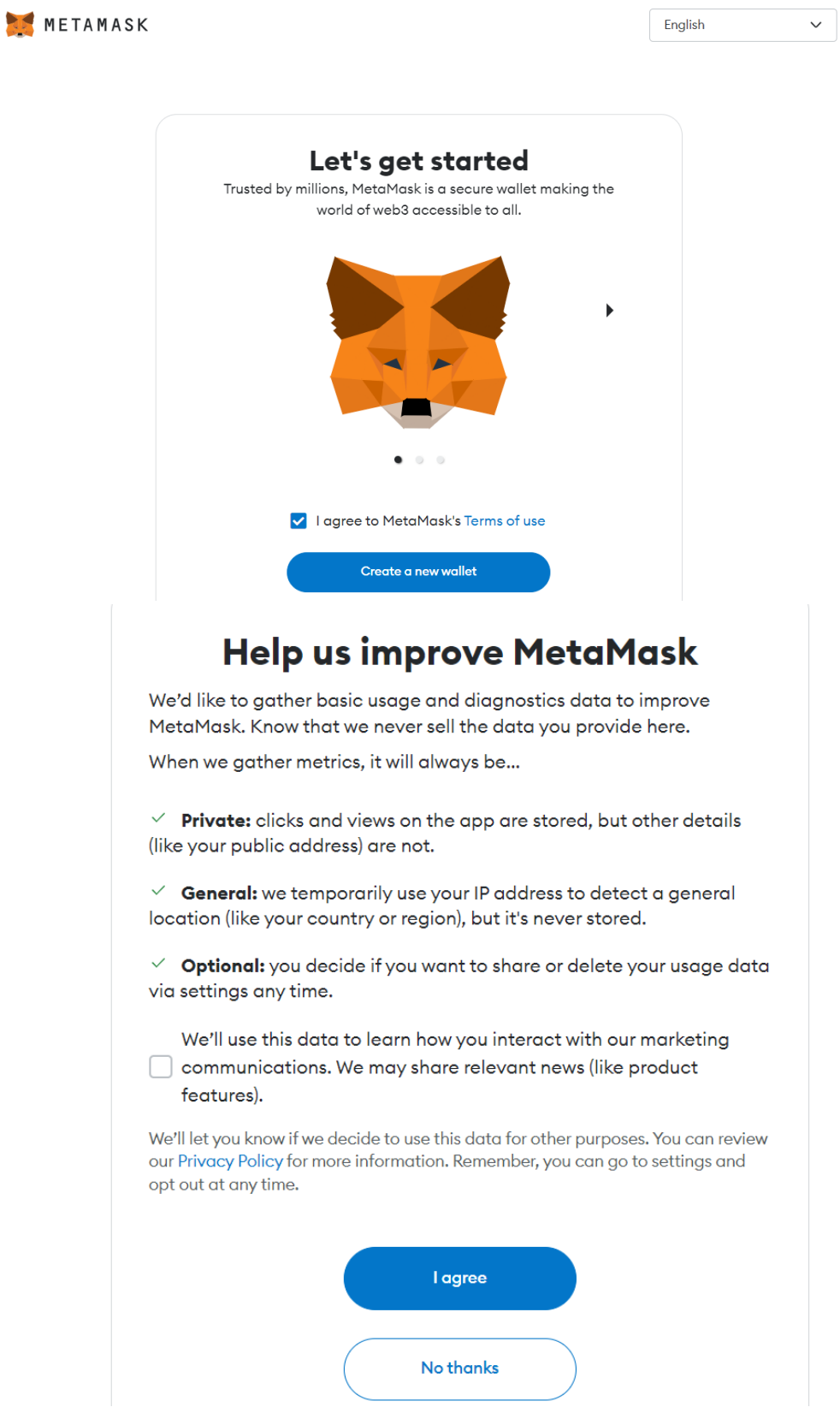
Aim: Install and configure Go Ethereum and the Mist browser. Develop and test a sample application.(MetaMask & Remix)

Step 1-> Install MetaMask extension for chrome from Chrome Web Store.

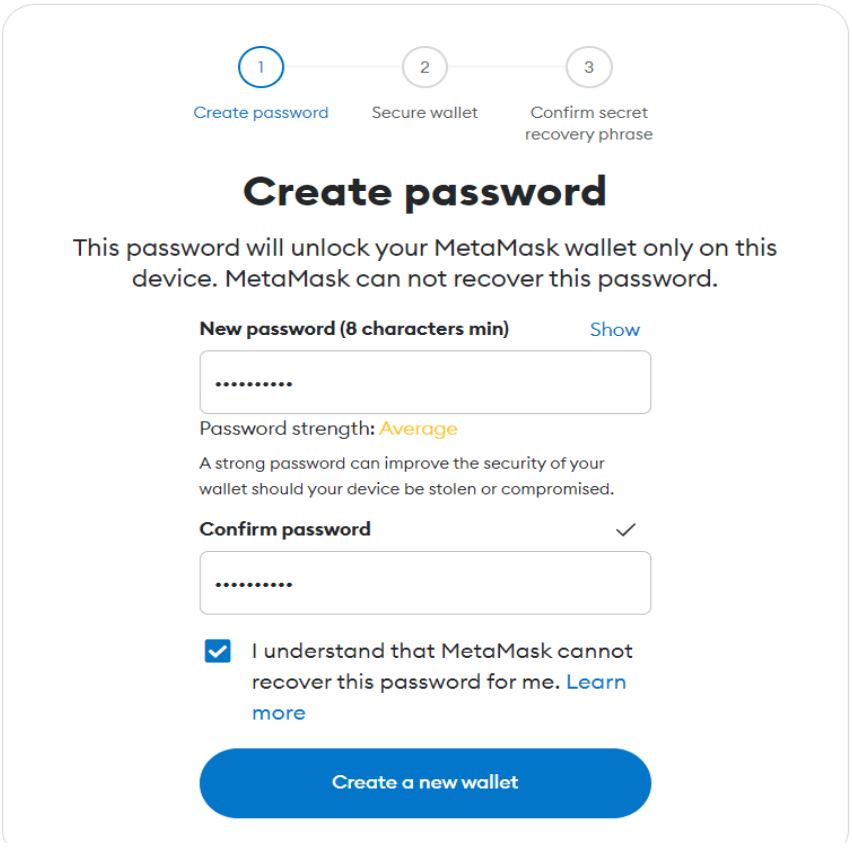


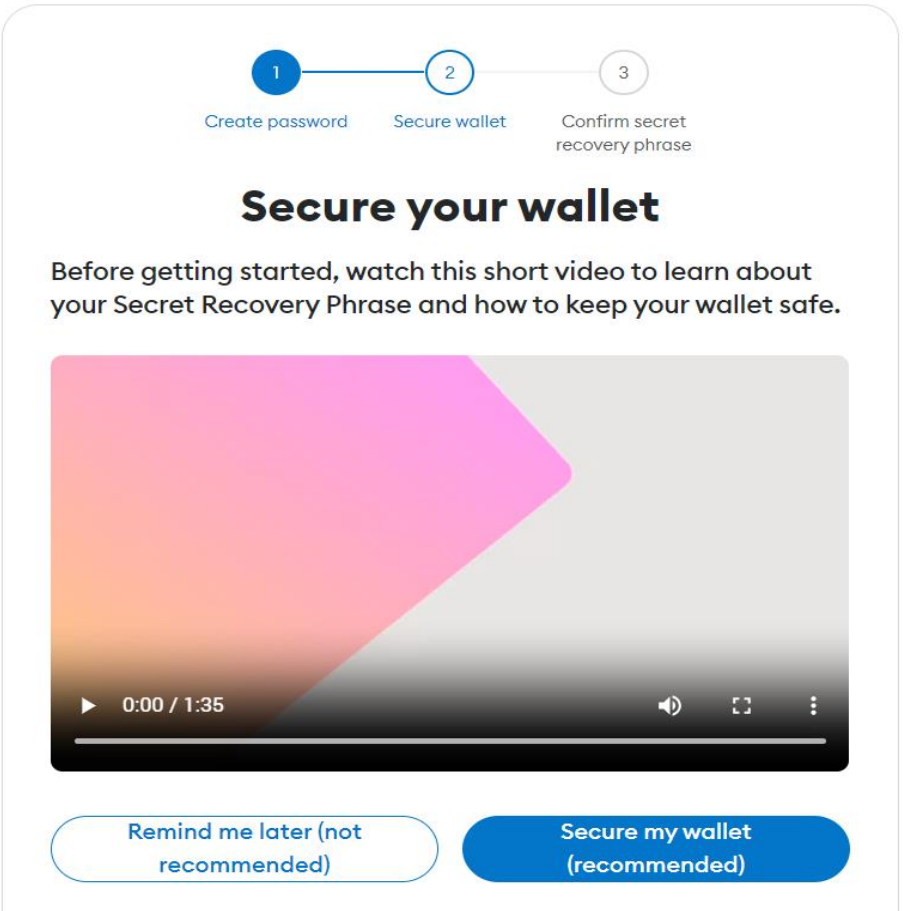
Step 2-> Click on Metamask Extension in Extensions. Below page will open in a new tab. Click on Create a New Wallet. Click on I agree.



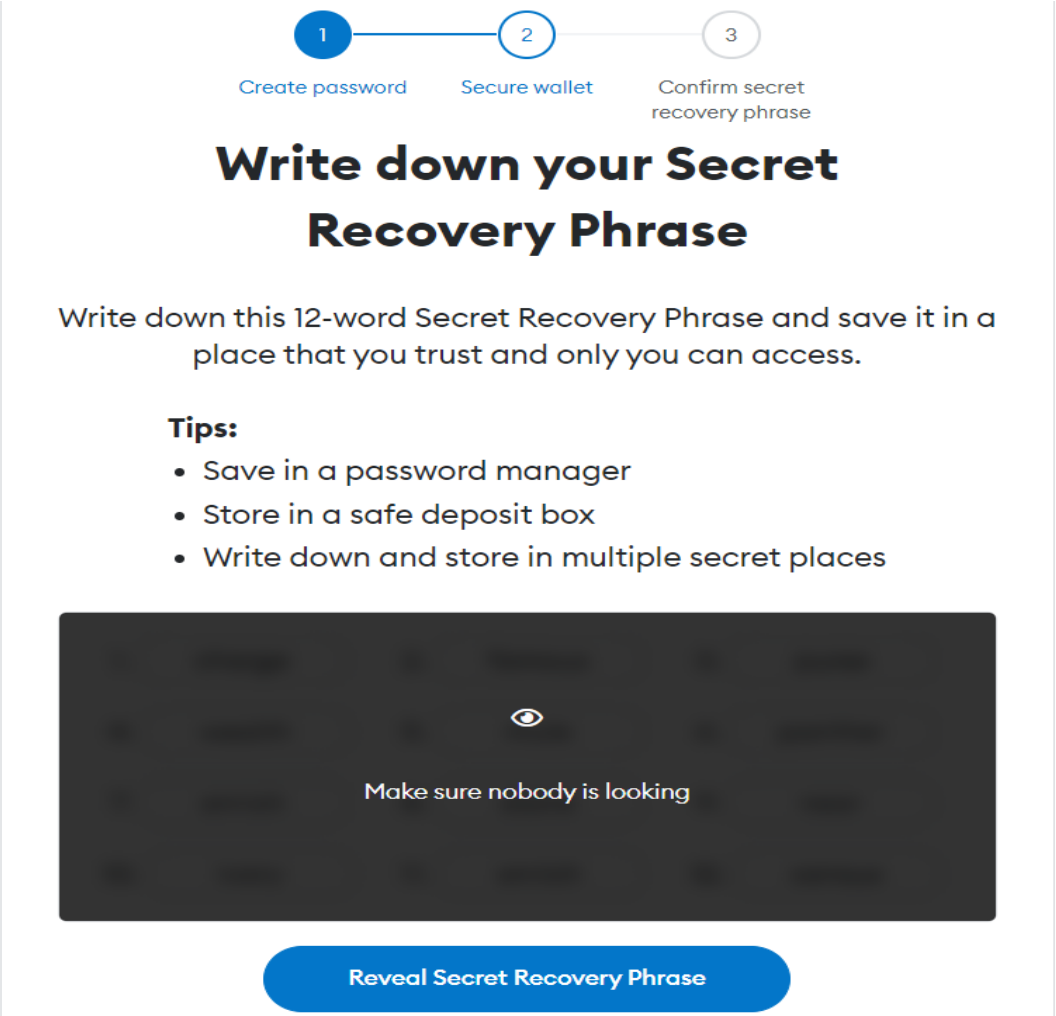


Step 3-> Create a password. This password can be used only on the device it was created on. Create a Strong password and click on Create a new Wallet button.

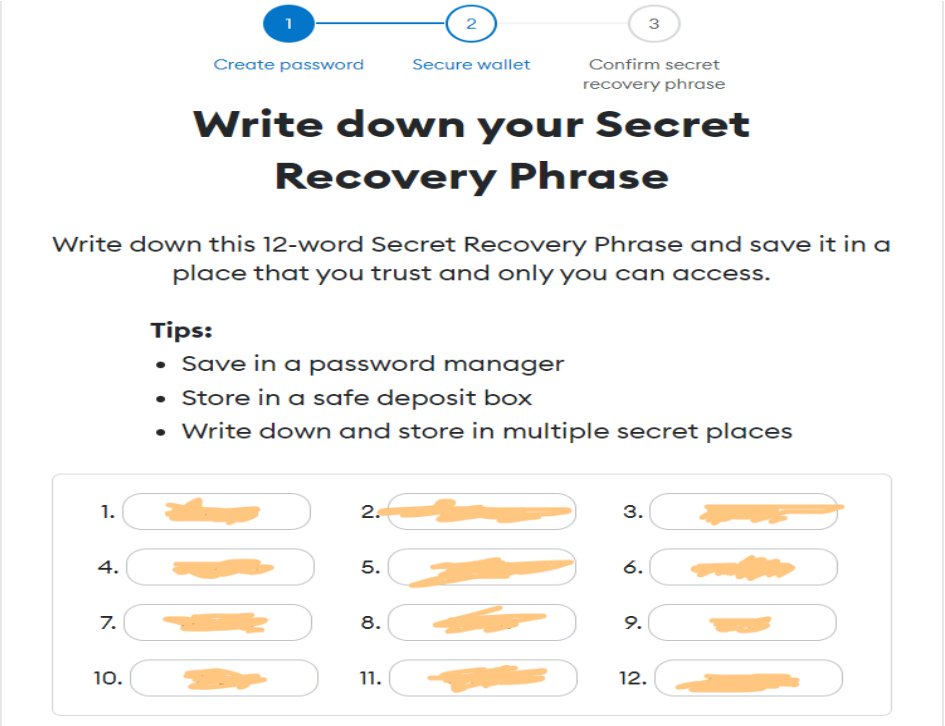




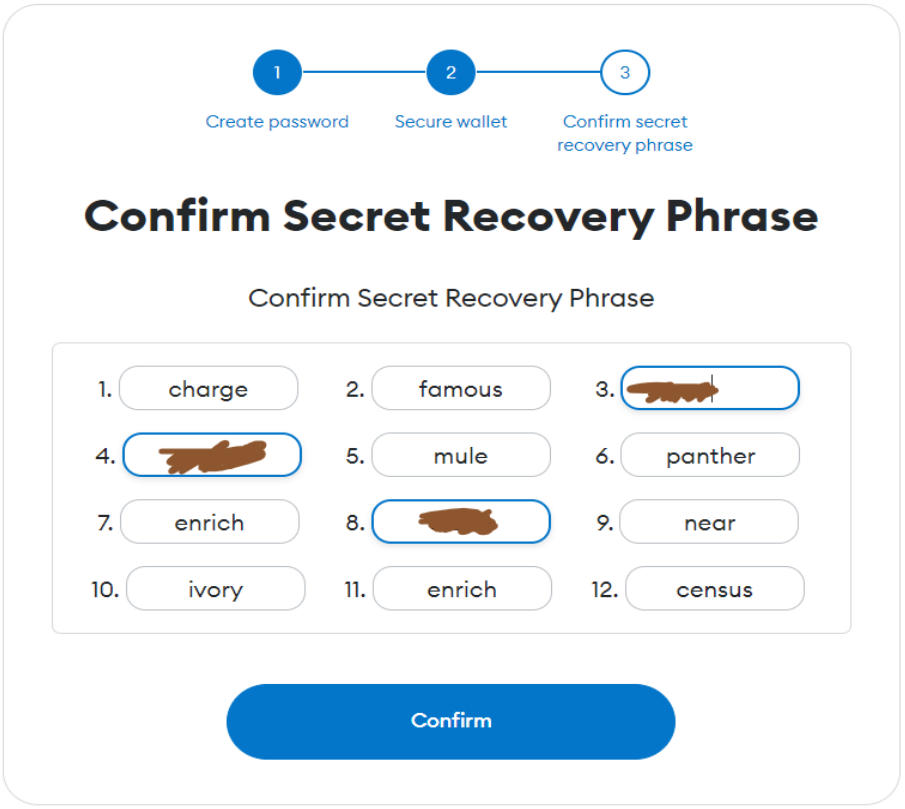
Step 4-> Click on Secure my wallet button, following window will appear.



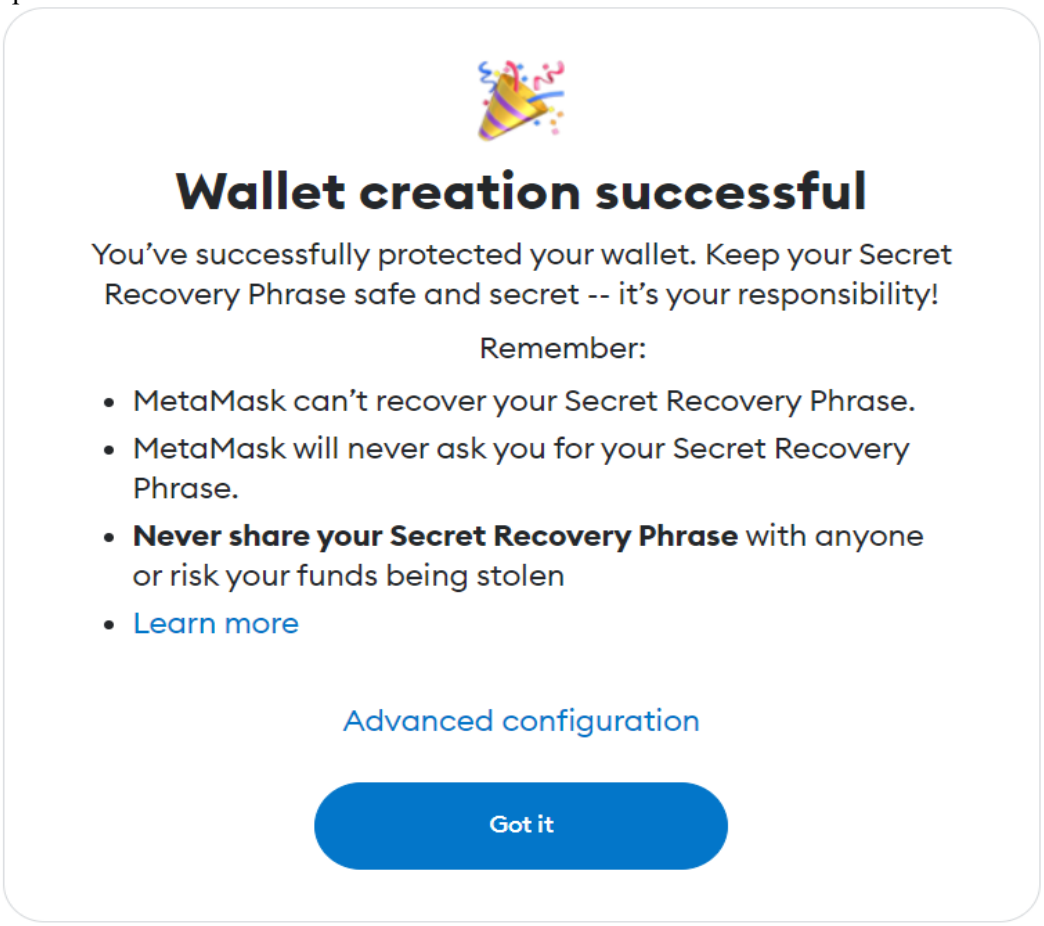
Step 5-> Click on Reveal Secret Recovery Phrase button and save the words in the same sequence.



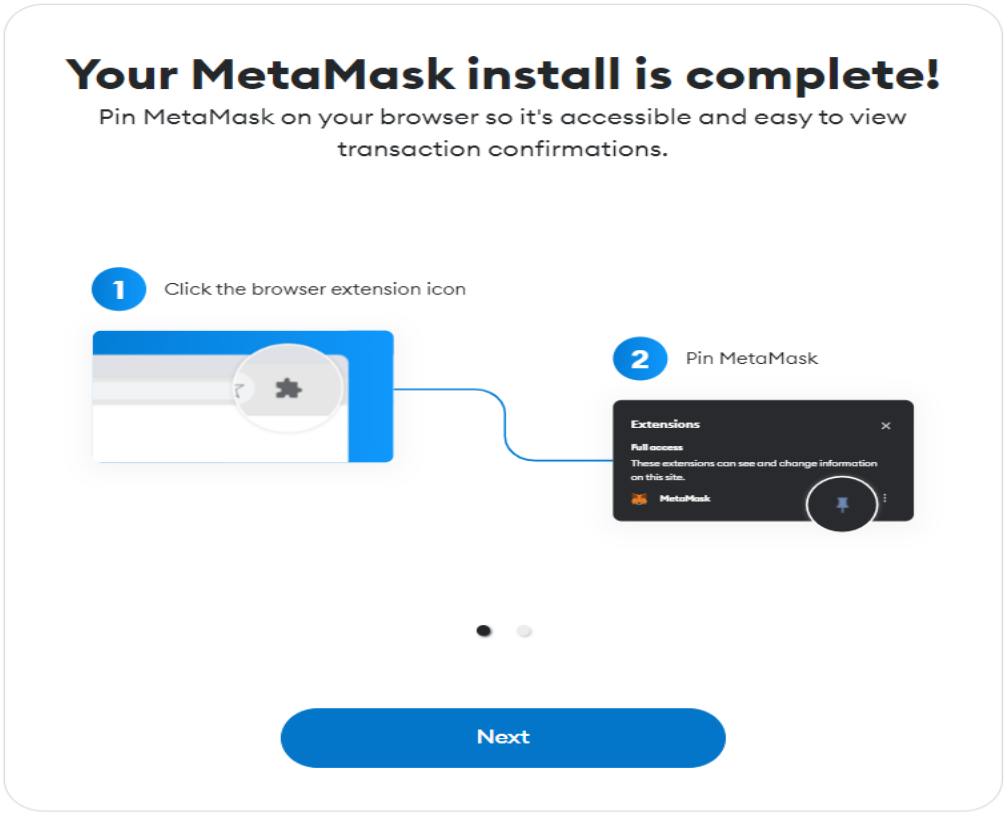
Step 6-> Enter the respective words in the empty positions and click Confirm.



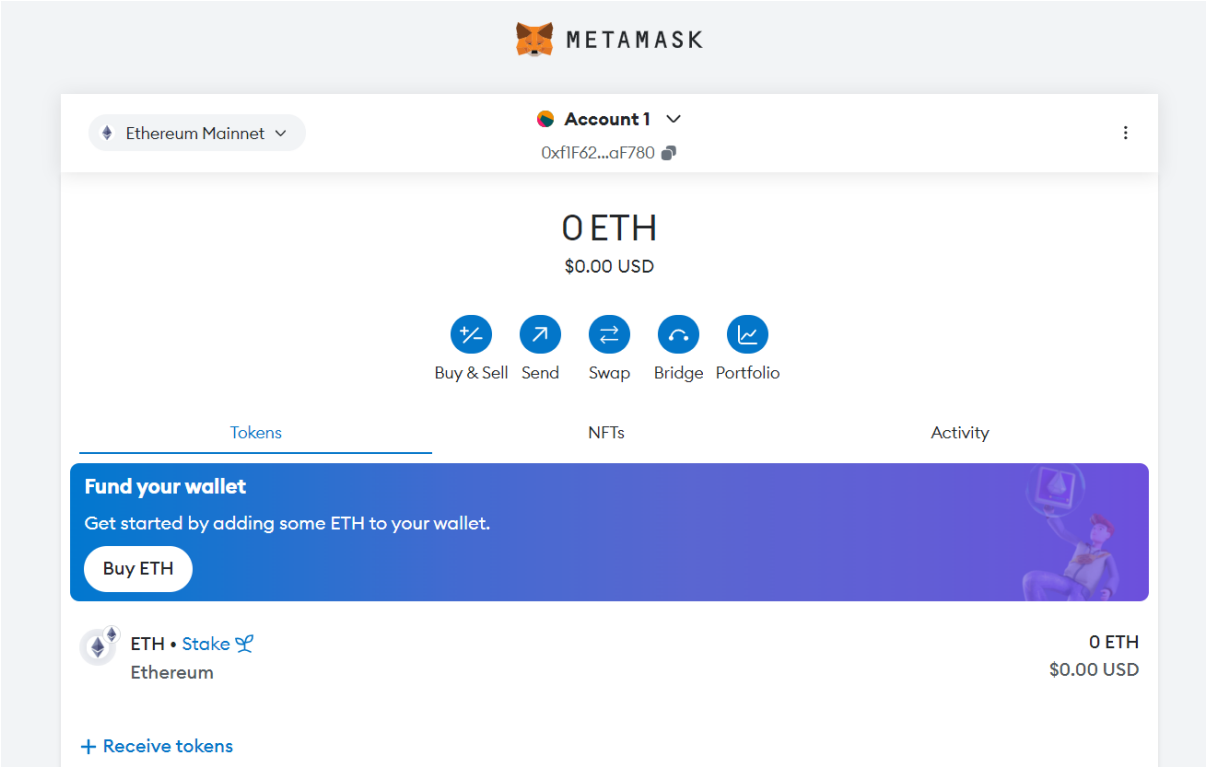
Step 7-> Click Got it!



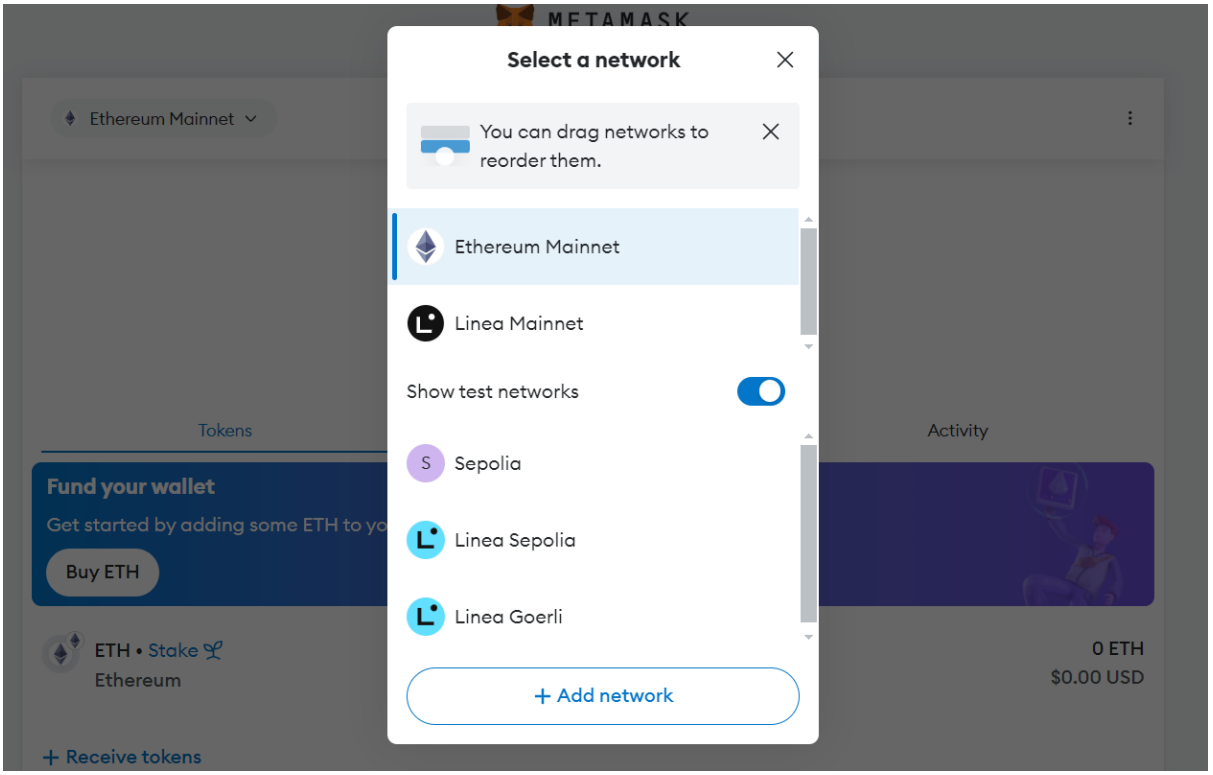
Step 8-> Click on Next



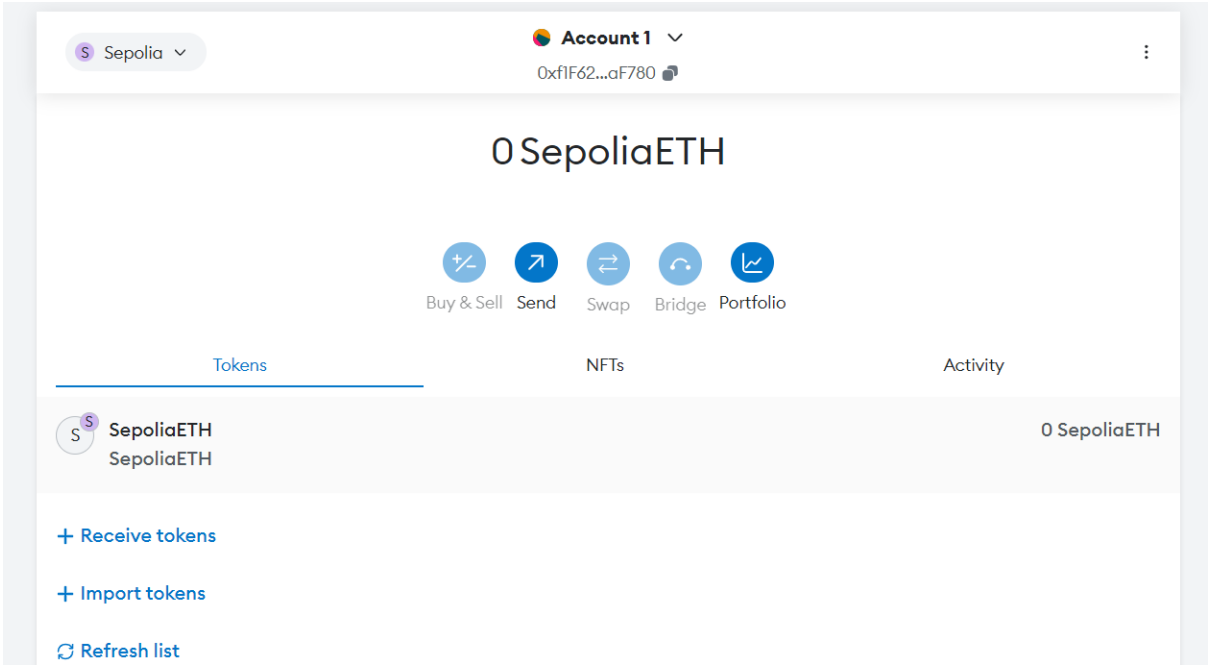
Step 9-> Following will be the Dashboard.



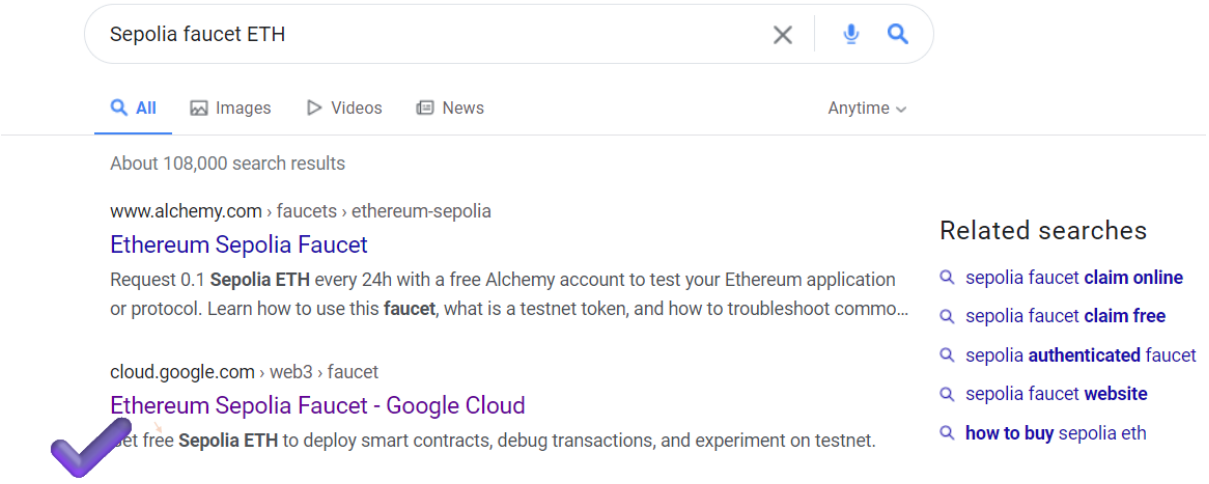
Step 10-> Click on Ethereum Mainnet button. Next click on Show/hide test networks.



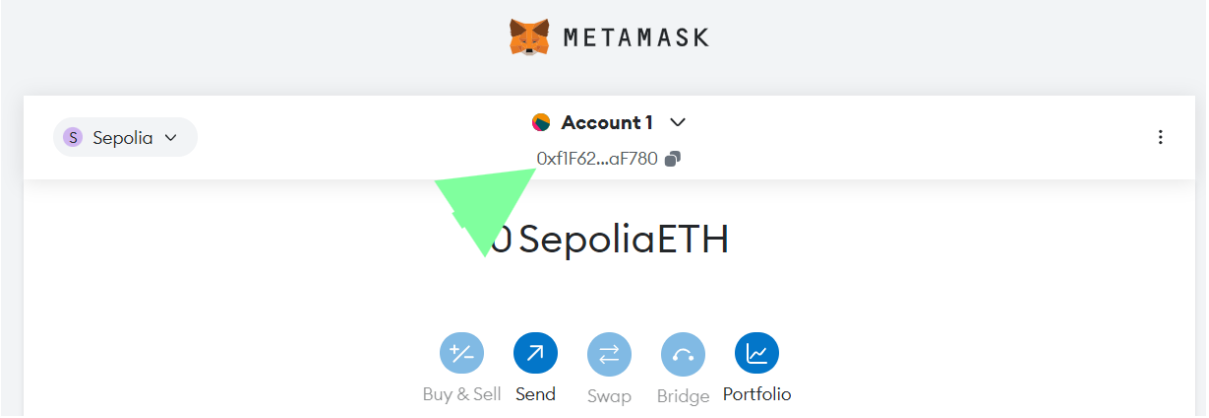
Step 11-> Check if tesnets are shown by clicking on Etherum Mainnet button. Click on Sepolia test network.



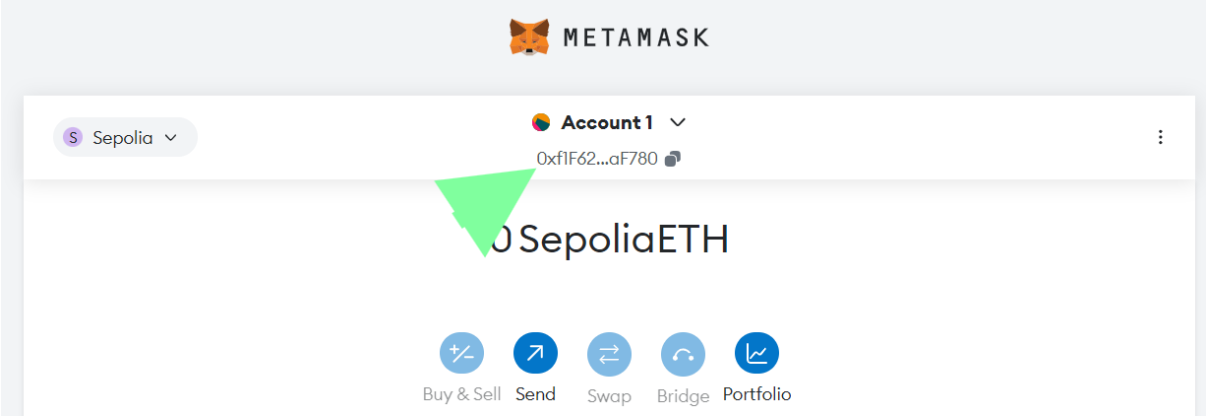
Step 12-> Go to https://sepoliafaucet.com/ and Click on Alchemy Login button.

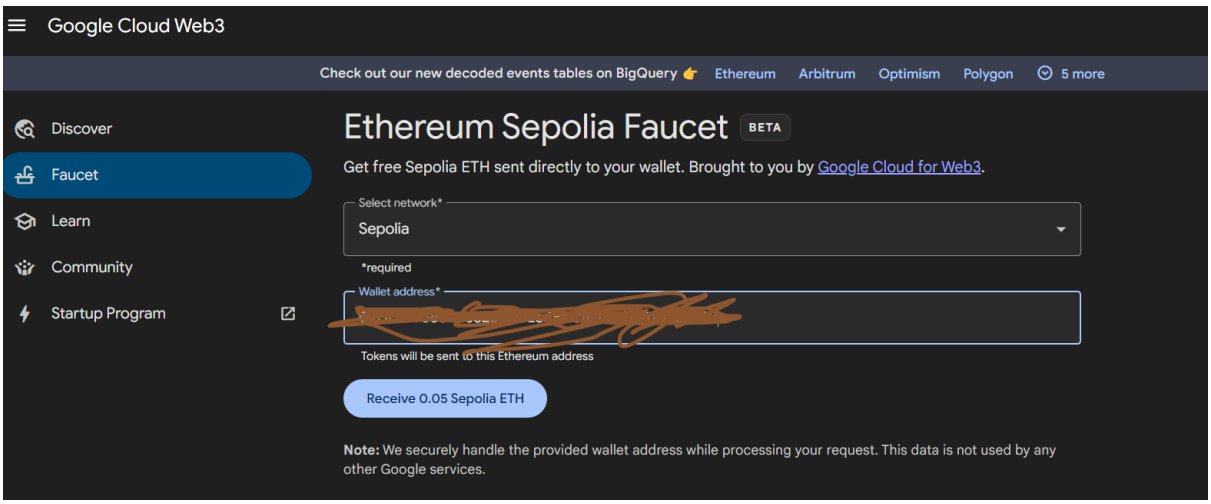


Step 13-> Now go to MetaMask and copy the account address.

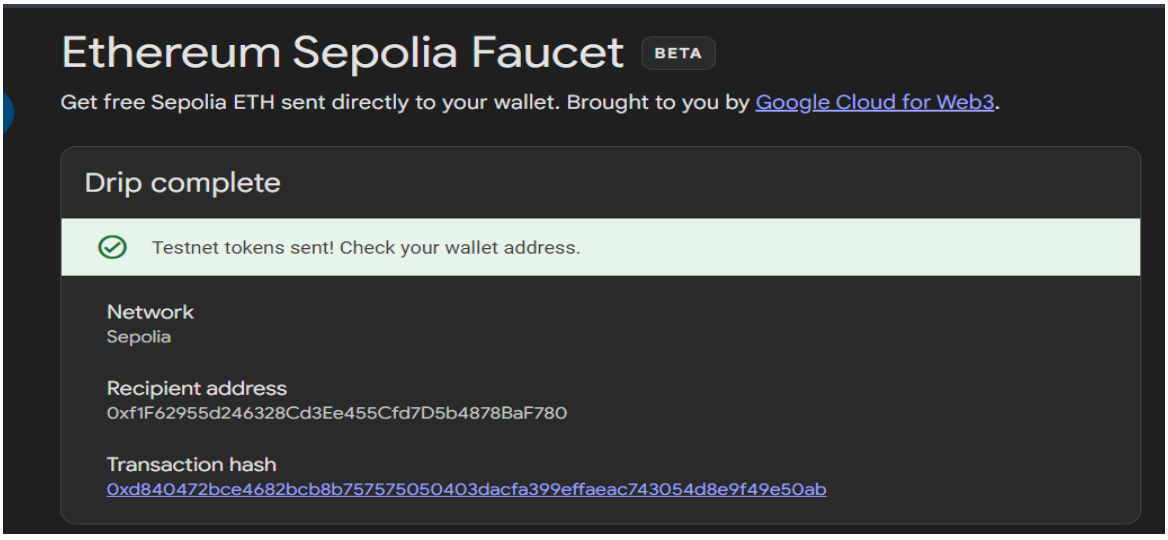


Step 14-> Paste the address and click on Send Me ETH.

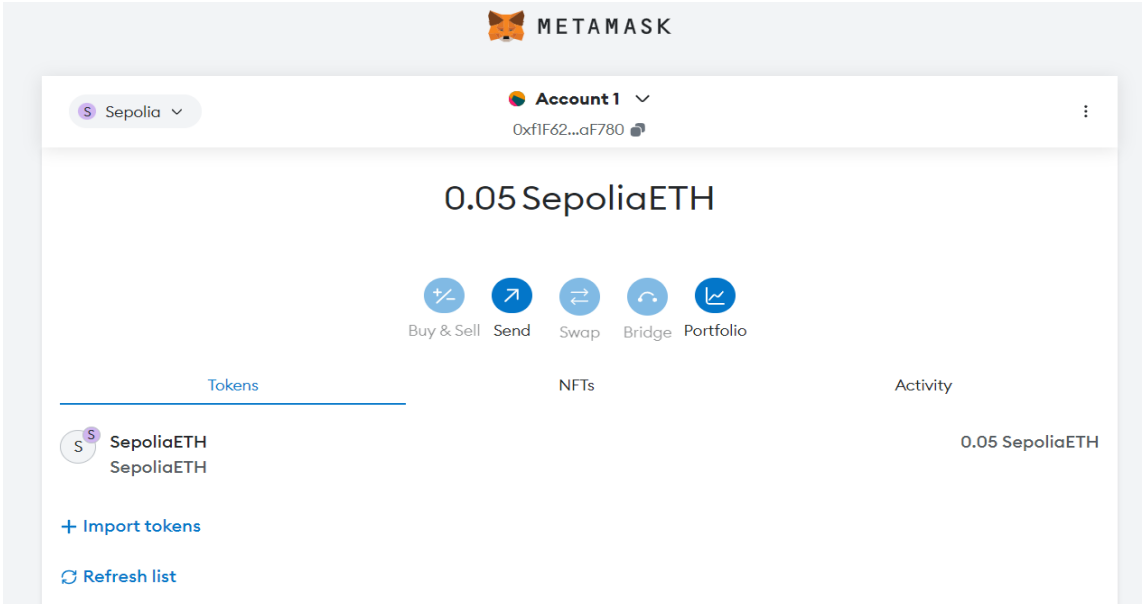




Step 16-> Your ETH transfer is successful.



Step 17-> Check your MetaMask account for Sepolia test network. 0.5 ETH will be added.



Practical 3

Aim: Implement and Demonstrate the use of the Following in Solidity.

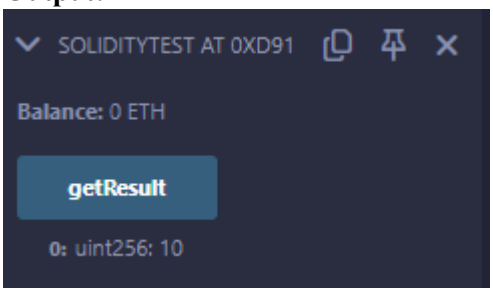
1. TO EXECUTE SOLIDITY SCRIPTS GO TO ->HTTPS://REMIX.ETHEREUM.ORG/
2. OPEN CONTRACTS FOLDER AND STARTING WRITING SCRIPTS. THE SCRIPTS ARE COMPILED USING SOLIDITY COMPILER.
3. THE FOLLOWING SCRIPTS WERE COMPILED USING 0.5.0+COMMIT.1D4F565A SOLIDITY COMPILER
4. DEPLOY THE SCRIPTS TO EXECUTE CODE

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

A. Variable:

```
1.  pragma solidity ^0.5.0;
2.  contract SolidityTest {
3.    uint storedData; // State variable
4.    constructor() public {
5.      storedData = 10;
6.    }
7.    function getResult() public view returns(uint){
8.      uint a = 1; // local variable
9.      uint b = 2;
10.     uint result = a + b;
11.     return storedData; //access the state variable
12.   }
13. }
```

Output:

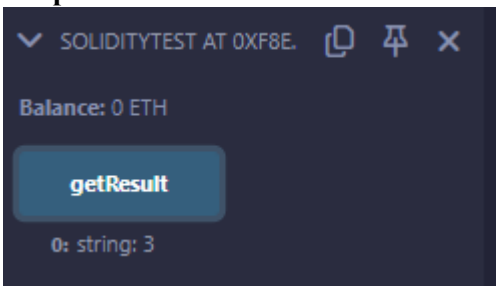


B. String:

```
pragma solidity ^0.5.0;

contract SolidityTest {
  constructor() public{
  }
  function getResult() public view returns(string memory){
    uint a = 1;
    uint b = 2;
    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:



C. Operators

```
pragma solidity ^0.5.0;
contract SolidityTest {
  uint16 public a = 20;
  uint16 public b = 10;
  uint256 public sum = a + b;
  uint256 public diff = a - b;
  uint256 public mul = a * b;
  uint256 public div = a / b;
  uint256 public mod = a % b;
  uint256 public dec = --b;
  uint256 public inc = ++a;
}
```

Output:

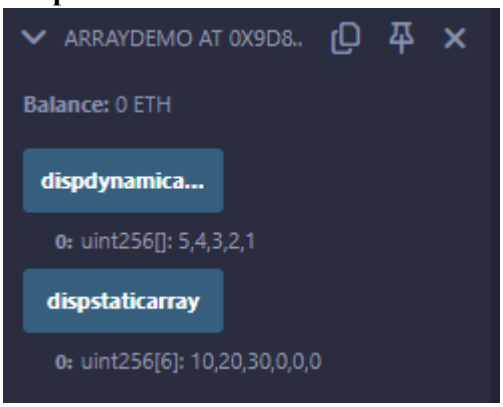
D. Array

Code:

```
pragma solidity ^0.5.0;

contract arraydemo {
   // Static Array
   uint[6] arr2 = [10, 20, 30];

   function dispstaticarray() public view returns (uint[6] memory) {
      return arr2;
   }

   // Dynamic Array
   uint x = 5;
   uint[] arr1;
```

```
  constructor() public {
    while (x > 0) {
      arr1.push(x);
      x = x - 1;
    }
  }

  function dispdynamicarray() public view returns (uint[] memory) {
    return arr1;
  }
}
```

Output:

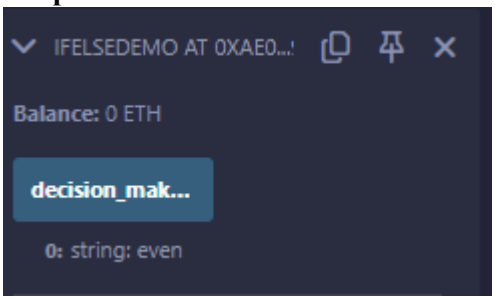


E. Decision Making:

Code:

```
pragma solidity ^0.5.0;

contract ifelsedemo {
  uint i = 10;

  function decision_making() public view returns (string memory) {
    if (i % 2 == 0) {
      return "even";
    } else {
      return "Odd";
    }
  }
}
```

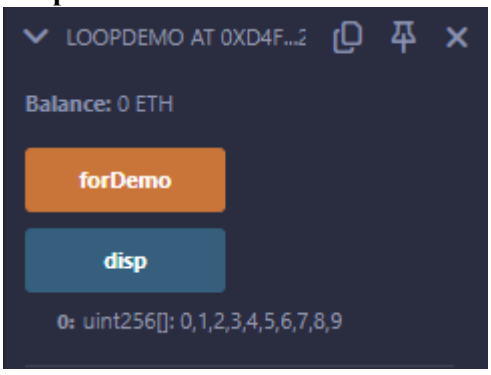
Output:



F. Loop:
Code:

```
pragma solidity ^0.5.0;

contract loopDemo {
   uint[] data;

   function forDemo() public returns (uint[] memory) {
      for (uint i = 0; i < 10; i++) {
         data.push(i);
      }
      return data;
   }

   function disp() public view returns (uint[] memory) {
      return data;
   }
}
```

Output:

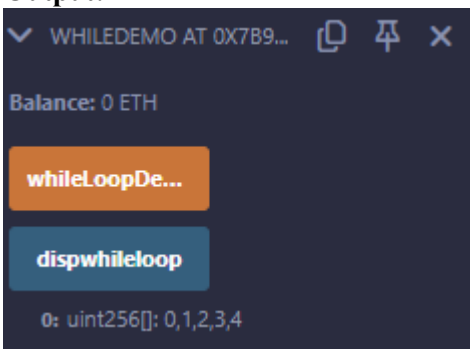


G. While Loop
Code:

```
pragma solidity ^0.5.0;

contract whiledemo {
   uint[] data;
```

```
    uint x = 0;

    function whileLoopDemo() public {
        while(x < 5) {
            data.push(x);
            x = x + 1;
        }
    }

    function dispwhileloop() public view returns(uint[] memory) {
        return data;
    }
}
```

Output:

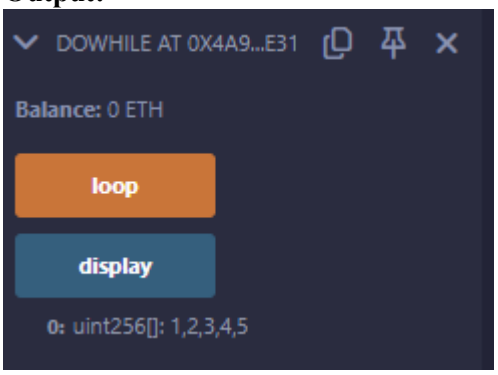


H. Do While Loop:

```
pragma solidity ^0.5.0;

// Creating a contract
contract DoWhile {
    // Declaring a dynamic array
    uint256[] data;
    // Declaring state variable
    uint8 j = 0;

    // Defining function to demonstrate 'Do-While loop'
    function loop() public returns (uint256[] memory) {
        do {
            j++;
            data.push(j);
        } while (j < 5);
        return data;
    }

    // Function to display the data array (view function)
    function display() public view returns(uint256[] memory) {
        return data;
```

```
    }
}
```

Output:



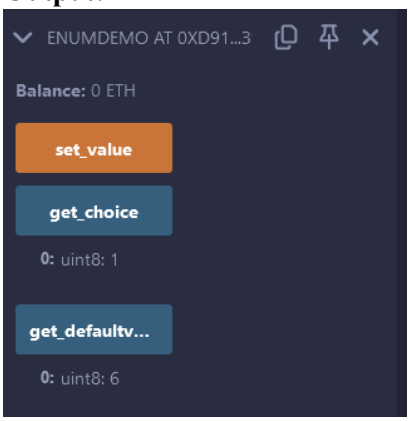
I. Enums:

Code:

```
pragma solidity ^0.5.0;

contract enumdemo {
    // Define the enum for week days
    enum week_days {
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday,
        Saturday,
        Sunday
    }

    // Declare variables to hold enum values
    week_days week;
    week_days choice;

    // Define a constant with a default enum value
    week_days constant default_value = week_days.Sunday;

    // Function to set a specific enum value
    function set_value() public {
        choice = week_days.Tuesday;
    }

    // Function to retrieve the current choice
    function get_choice() public view returns (week_days) {
        return choice;
    }
```

```
    // Function to retrieve the default enum value
    function get_defaultvalue() public view returns (week_days) {
        return default_value;
    }
}

// FIGURE 10 - ACCESSING ENUM VALUES
// Page 46
```

Output:



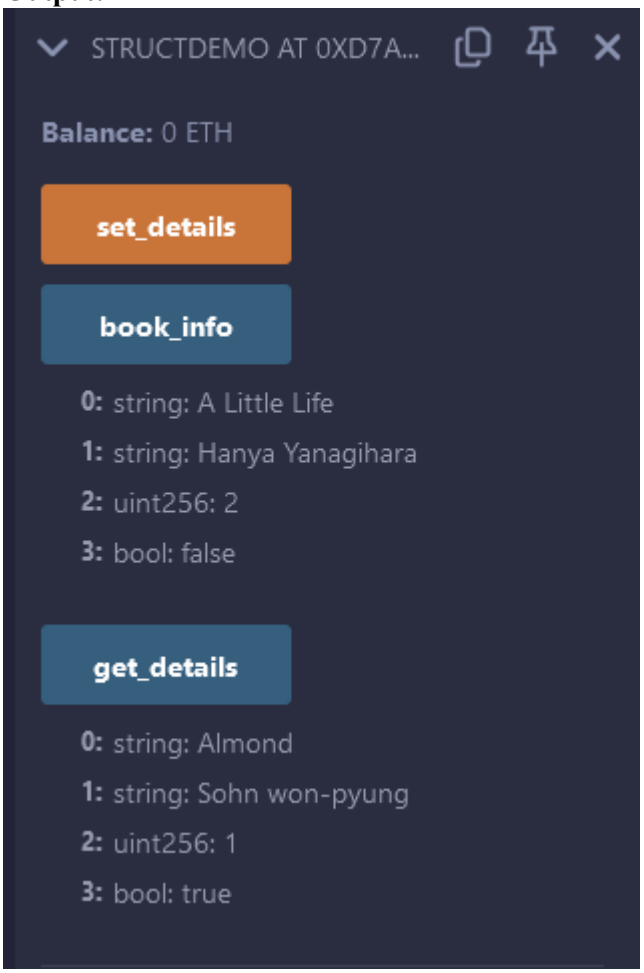
J. Structs

Code:

```
pragma solidity ^0.5.0;
contract structdemo {
struct Book {
string name;
string author;
uint256 id;
bool availability;
}
Book book2;
Book book1 = Book("A Little Life", "Hanya Yanagihara", 2, false);
function set_details() public {
book2 = Book("Almond", "Sohn won-pyung", 1, true);
}
function book_info()
public
view
returns (
string memory,
string memory,
uint256,
bool
)
{
return (book1.name, book1.author, book1.id, book1.availability);
```

```
}
function get_details()
public
view
returns (
string memory, string memory, uint256, bool
)
{
return (book2.name, book2.author, book2.id, book2.availability);
}
}
```

Output:



K. Mapping:
Code:

```
pragma solidity ^0.5.0;

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

  function updateBalance(uint256 newBalance) public {
```

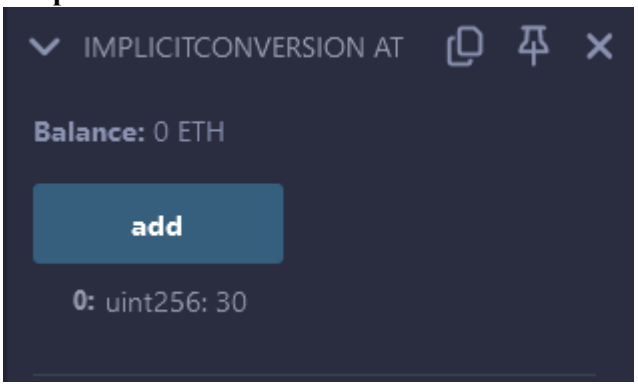
```
        balances[msg.sender] = newBalance;
    }
}

contract Updater {
    LedgerBalance ledgerBalance;

    constructor(address _ledgerBalanceAddress) public {
        ledgerBalance = LedgerBalance(_ledgerBalanceAddress);
    }

    function updateBalance(uint256 newBalance) public {
        ledgerBalance.updateBalance(newBalance);
    }

    function getBalance() public view returns (uint256) {
        return ledgerBalance.balances(address(this));
    }
}
```

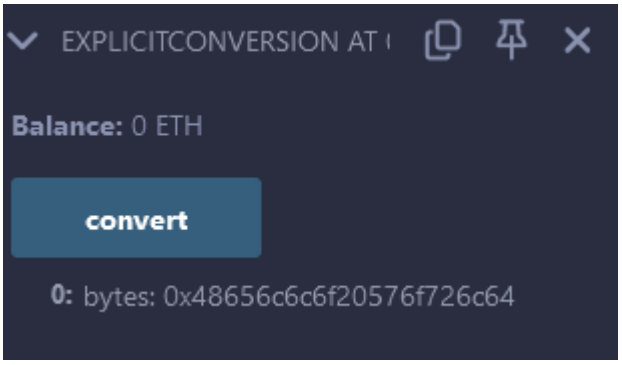
L. Conversation

Code:

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

contract ImplicitConversion {
    function add() public pure returns (uint256) {
        uint256 a = 10;
        uint256 b = 20;
        return a + b;
    }
}

contract ExplicitConversion {
    function convert() public pure returns (bytes memory) {
        string memory str = "Hello World";
        bytes memory b = bytes(str);
        return b;
    }
}
```

Output:





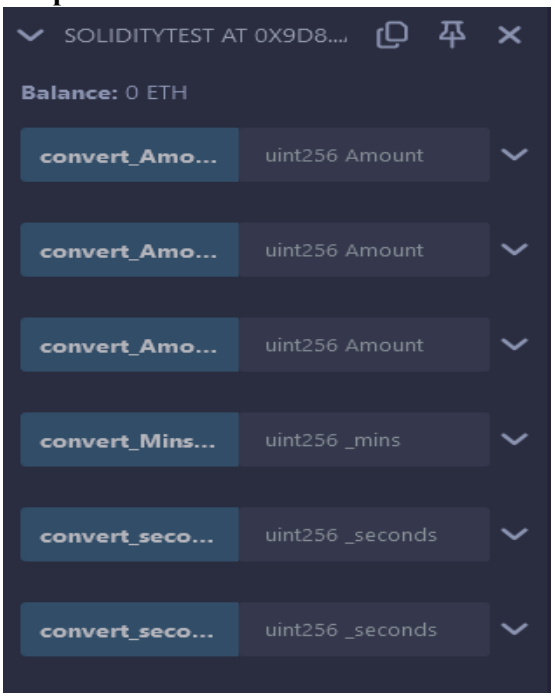
M. Ether Units

Code:

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

contract SolidityTest {

   function convert_Amount_to_Wei(uint256 Amount)
      public
      pure
      returns (uint256)
   {
      return Amount * 1 wei;
   }

   function convert_Amount_To_Ether(uint256 Amount)
      public
      pure
      returns (uint256)
   {
      return Amount * 1 ether;
   }

   function convert_Amount_To_Gwei(uint256 Amount)
      public
      pure
      returns (uint256)
```

```
  {
    return Amount * 1 gwei;
  }

  function convert_seconds_To_mins(uint256 _seconds)
    public
    pure
    returns (uint256)
  {
    return _seconds / 60;
  }

  function convert_seconds_To_Hours(uint256 _seconds)
    public
    pure
    returns (uint256)
  {
    return _seconds / 3600;
  }

  function convert_Mins_To_Seconds(uint256 _mins)
    public
    pure
    returns (uint256)
  {
    return _mins * 60;
  }
}
```

Output:



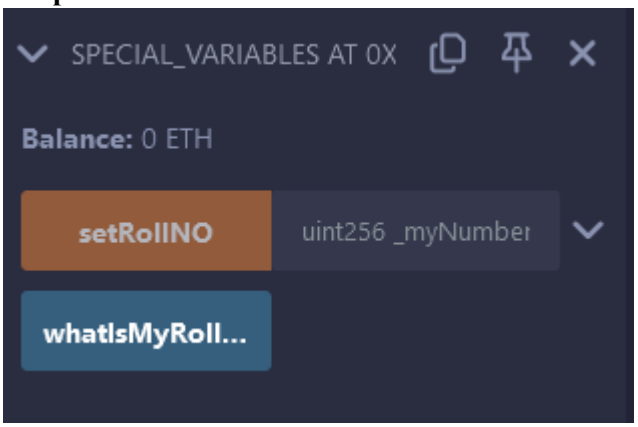
Provide the values:

N. Special Variables

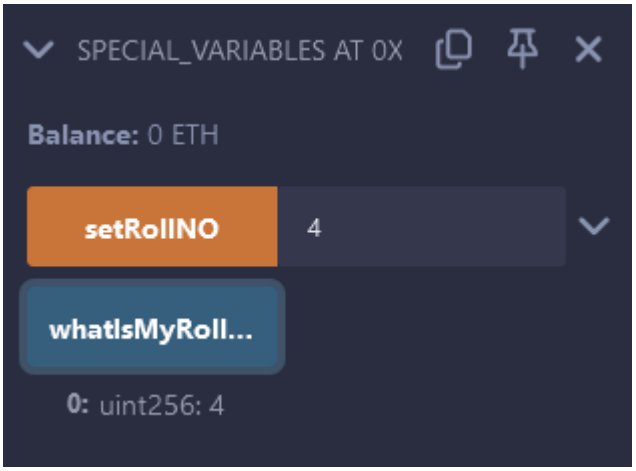
Code:
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Special_Variables {
    mapping(address => uint256) rollNo;

    function setRollNO(uint256 _myNumber) public {
        rollNo[msg.sender] = _myNumber;
    }

    function whatIsMyRollNumber() public view returns (uint256) {
```

```
    return rollNo[msg.sender];
  }
}
```

Output:



Provide the input:

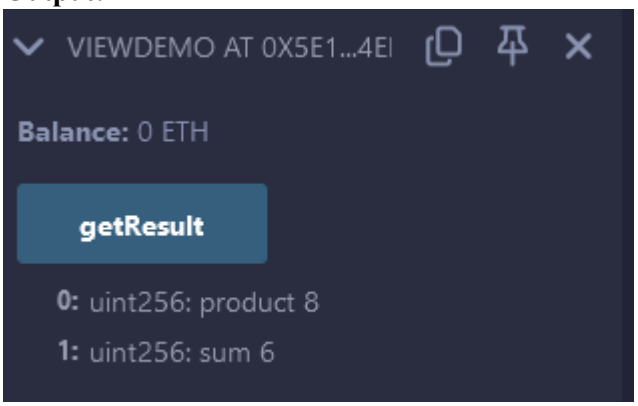


B) Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

1. View Function

```
pragma solidity ^0.5.0;

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

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

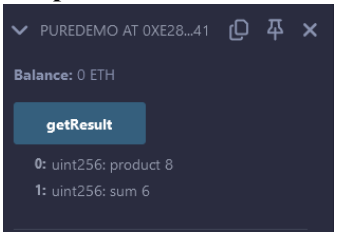
Output:



2. Pure Function:

```
pragma solidity ^0.5.0;

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

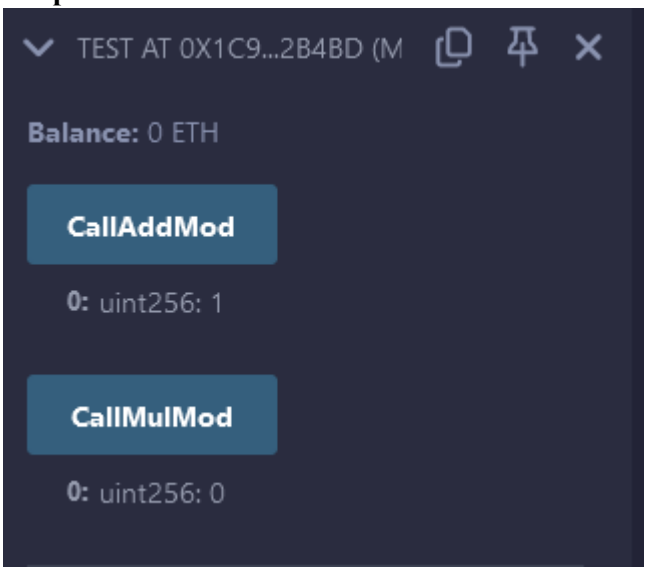
Output:



3. Mathematical Function:

```
pragma solidity ^0.5.0;

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

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

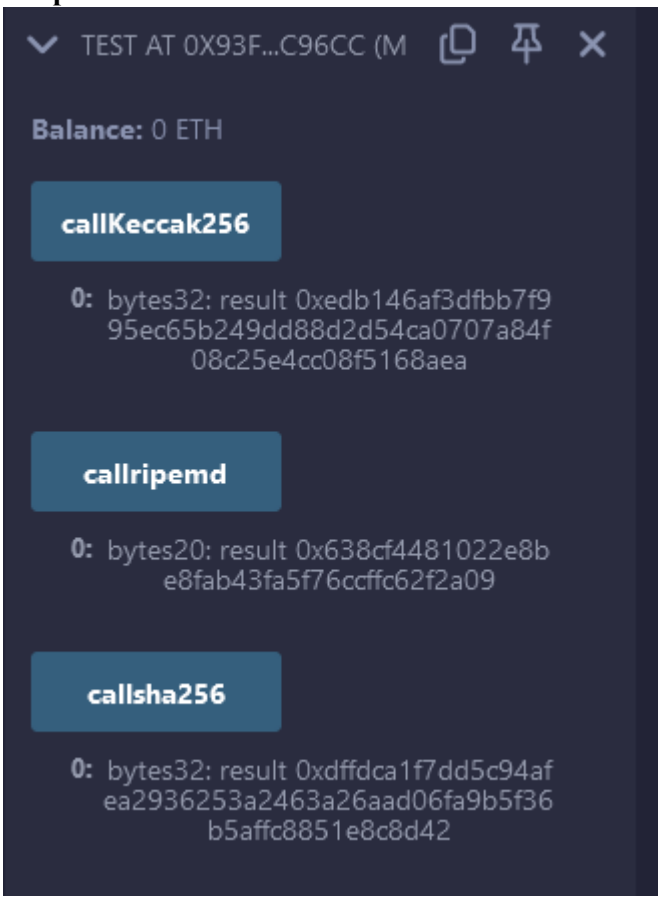
Output:



4.Cryptograhic:

```
pragma solidity ^0.5.0;

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

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

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

Output:

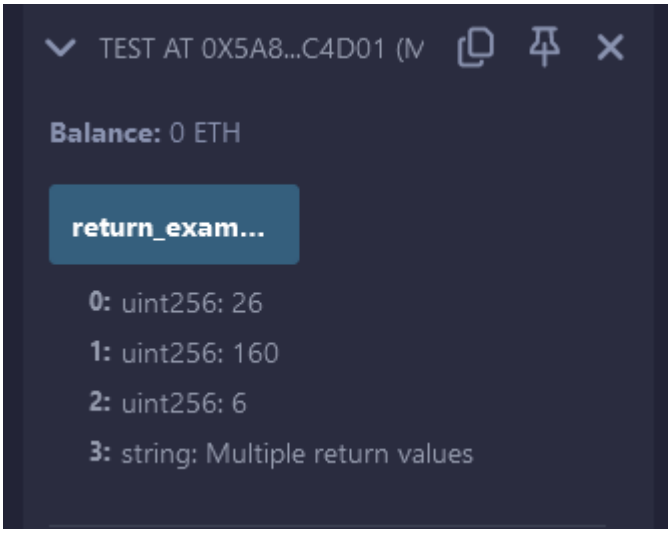


5. Functions:

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

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

```
}
```

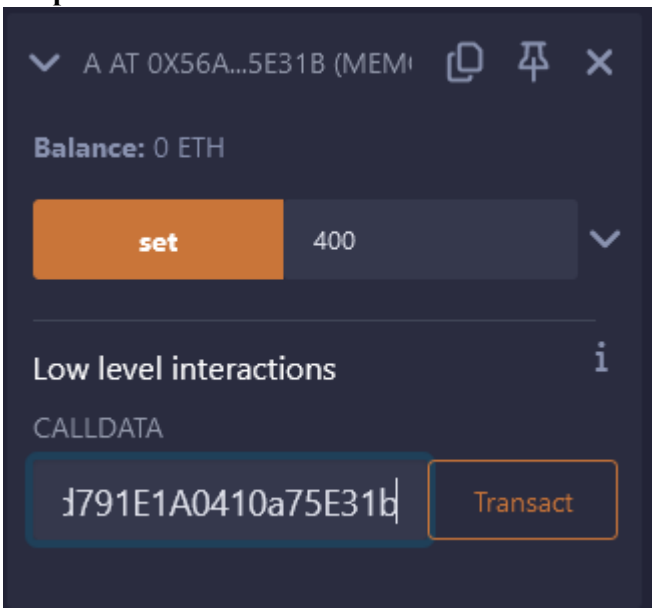
Output:



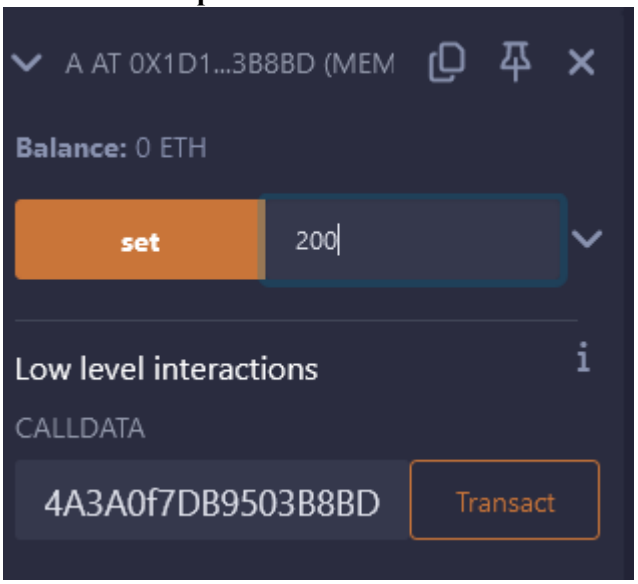
6. Fallback Function:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.12;

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:



Contract Example:



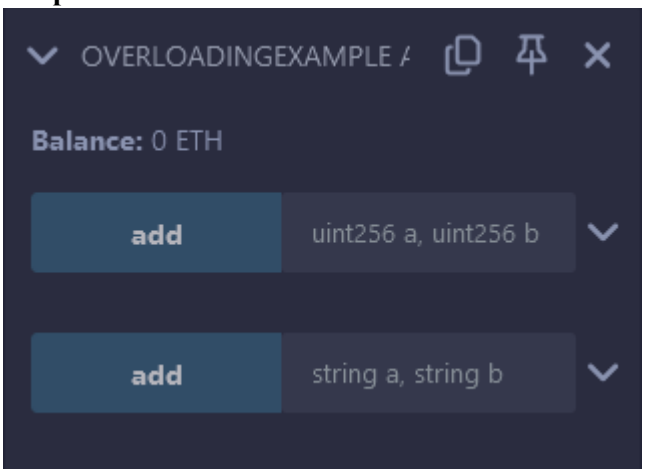
7. Function Overloading:

Code:

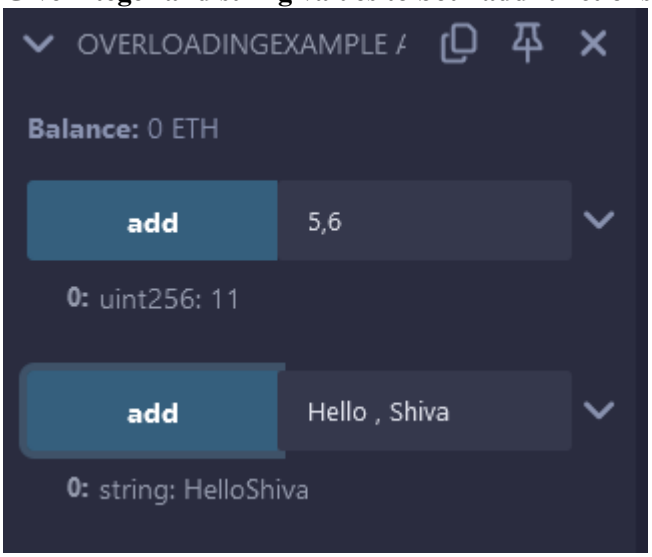
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract OverloadingExample {

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

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

```
}
```

Output:



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



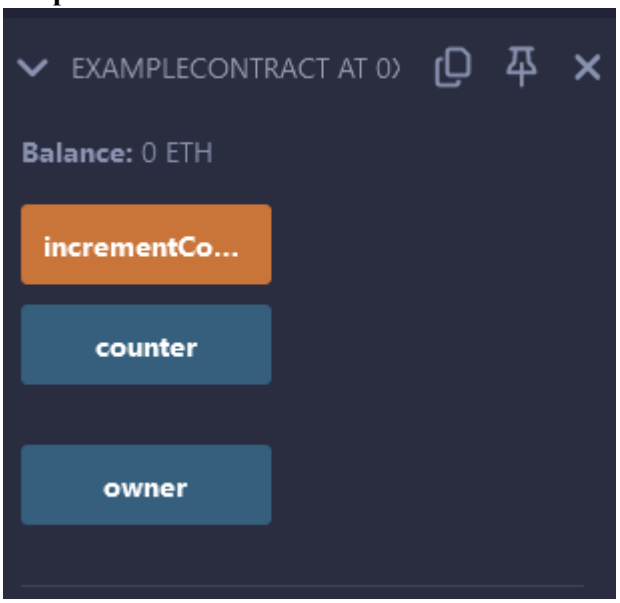
8. Function Modifiers:

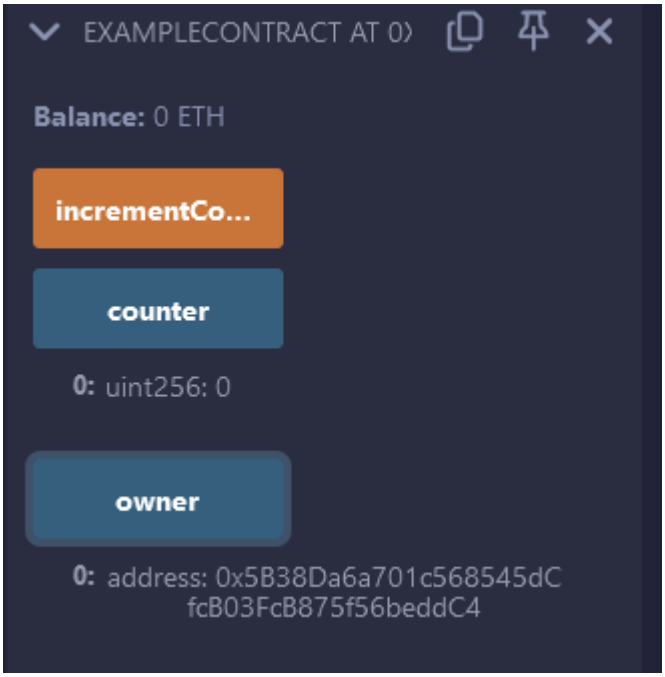
Code:

```
// 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 {
```

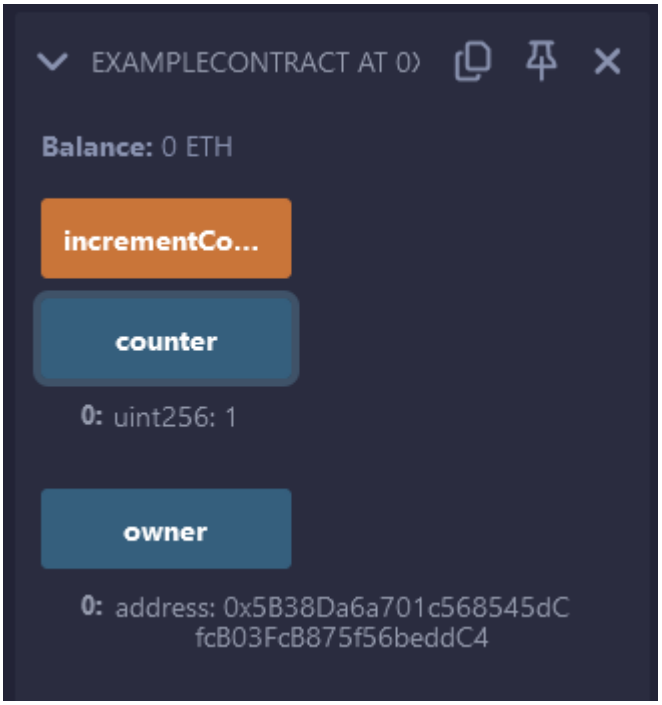
```
        counter++;
    }
}
```

Output:





As we click on Increment Button it will Increment One by One.



Practical 4

Aim: Implement and Demonstrate the use of then following in solidity.

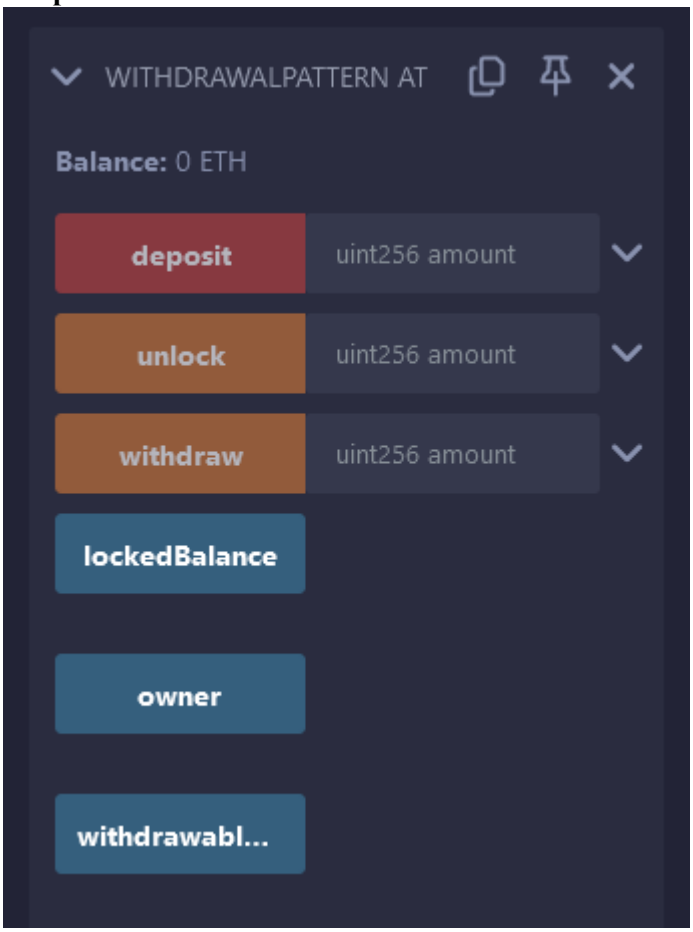
A. Withdrawal Pattern, Restricted Access.

Code:

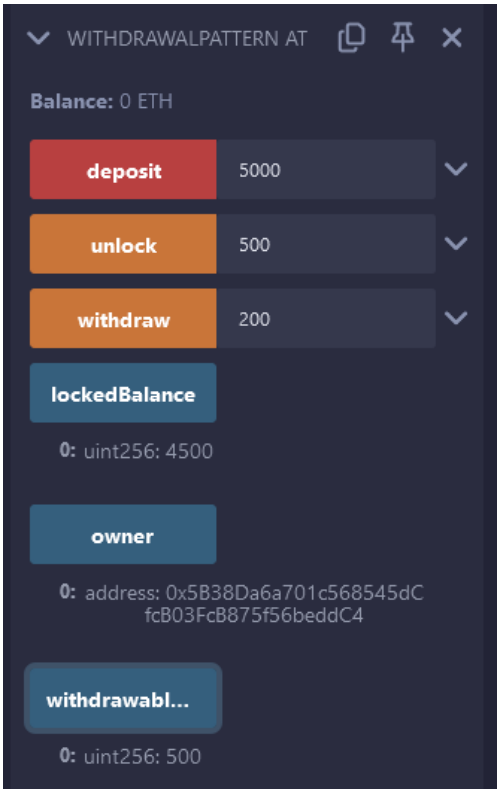
1. Withdrawal Pattern.

```
2.  // SPDX-License-Identifier: MIT
3.  pragma solidity 0.8.18;
4.
5.  contract WithdrawalPattern {
6.      address public owner;
7.      uint256 public lockedBalance;
8.      uint256 public withdrawableBalance;
9.
10.     constructor() {
11.         owner = msg.sender;
12.     }
13.
14.     modifier onlyOwner() {
15.         require(msg.sender == owner, "Only the owner can call this function");
16.         _;
17.     }
18.
19.     function deposit(uint256 amount) public payable {
20.         require(amount > 0, "Amount must be greater than zero");
21.         lockedBalance += amount;
22.     }
23.
24.     function withdraw(uint256 amount) public onlyOwner {
25.         require(amount <= withdrawableBalance, "Insufficient withdrawable balance");
26.         withdrawableBalance -= amount;
27.         payable(msg.sender).transfer(amount);
28.     }
29.
30.     function unlock(uint256 amount) public onlyOwner {
31.         require(amount <= lockedBalance, "Insufficient locked balance");
32.         lockedBalance -= amount;
33.         withdrawableBalance += amount;
34.     }
35. }
```

Output:



Input All the required details.

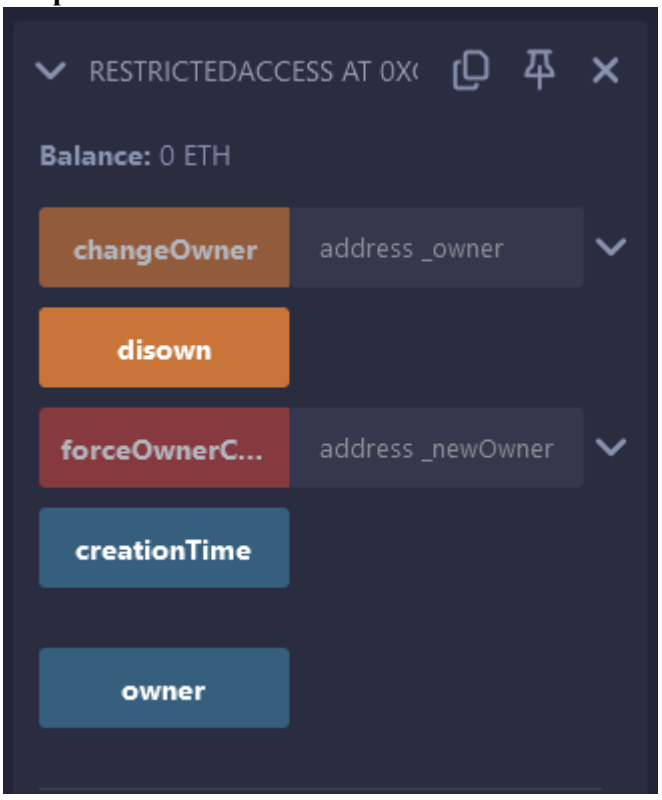


2. Restricted Access

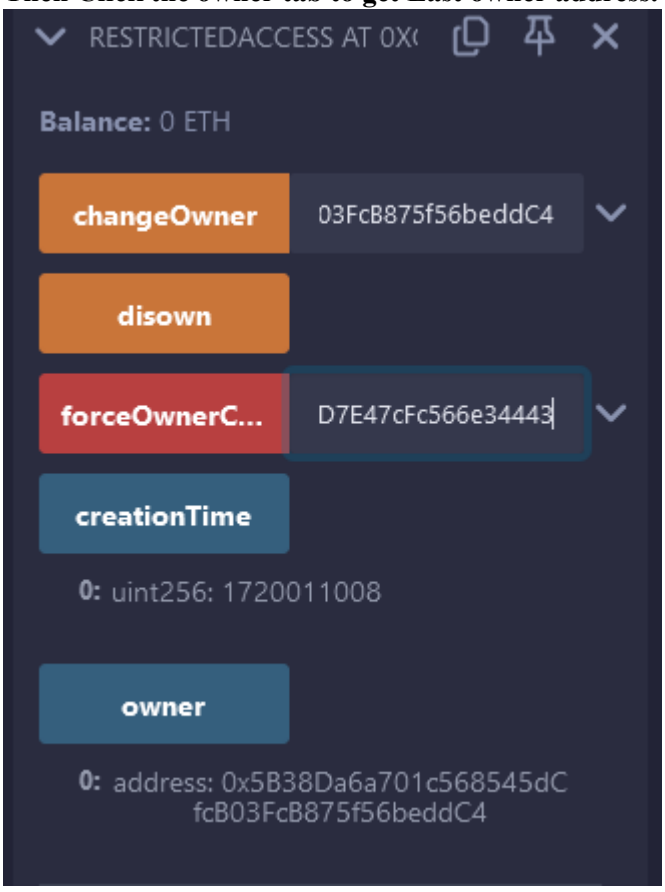
Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

contract RestrictedAccess {
    address public owner = msg.sender;
    uint256 public creationTime = block.timestamp;

    modifier onlyBy(address _account) {
        require(msg.sender == _account, "Sender not authorized!");
        _;
    }

    modifier onlyAfter(uint256 _time) {
        require(block.timestamp >= _time, "Function was called too early!");
        _;
    }

    modifier costs(uint256 _amount) {
        require(msg.value >= _amount, "Not enough Ether provided!");
        _;
    }

    function forceOwnerChange(address _newOwner)
        public
        payable
        costs(200 ether)
    {
        owner = _newOwner;
    }

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

    function disown() public onlyBy(owner) onlyAfter(creationTime + 3 weeks) {
        delete owner;
    }
}

/*Last owner address:
0:
address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4  */
```

Output:



In this we change the owner address and Input New Owner Address.
Then Click the owner tab to get Last owner address.



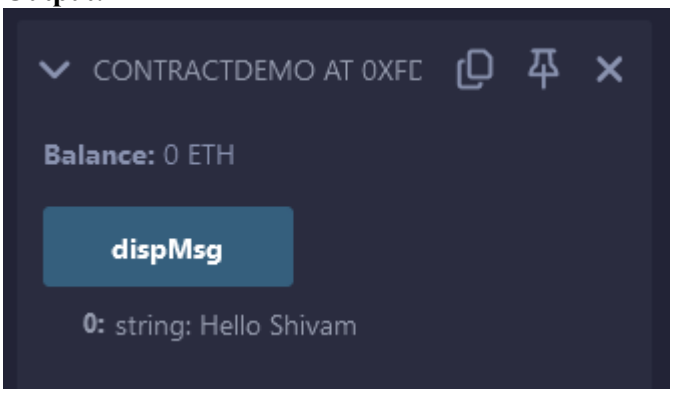
B. Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces

1. Contracts

Code:

```
pragma solidity ^0.5.0;

contract ContractDemo {
   string message = "Hello Shivam";

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

Output:



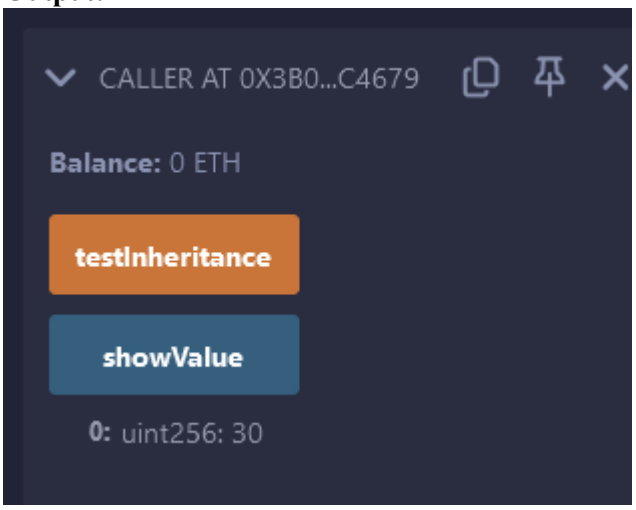
2. Inheritance

Code:

```
pragma solidity >=0.4.22 <0.6.0;

contract Parent {
   uint256 internal sum;

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

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

contract Caller {
   Child cc = new Child();
```

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

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

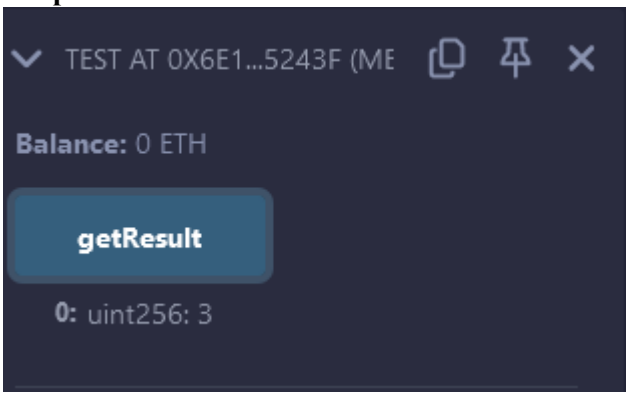
Output:



3. Abstract Access.
Code:

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

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

contract Test is Calculator {
    constructor() public {}

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

Output:

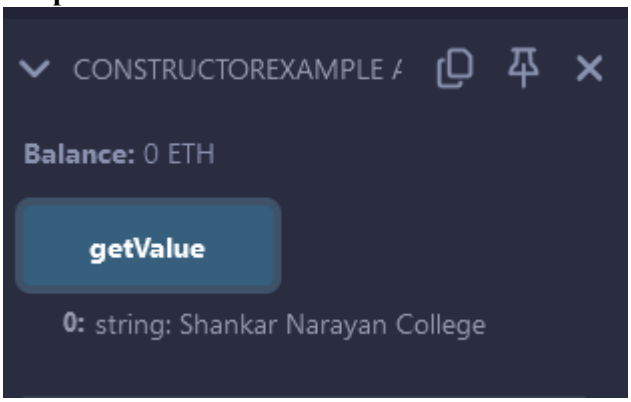


4. Constructor:

Code:

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

// Creating a contract
contract constructorExample {
   string str;

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

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

Output:



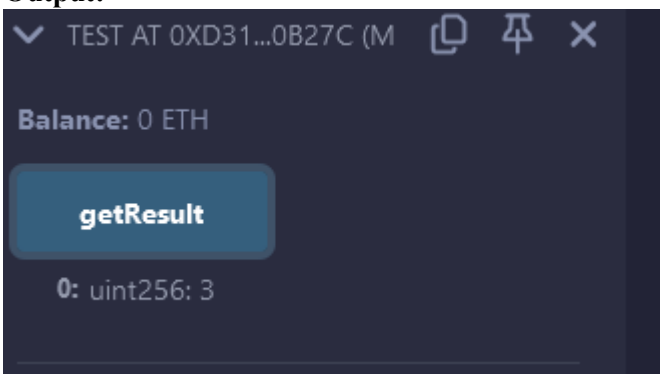
5.Interfaces:

Code:

```
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:



C. Libraries, Assembly, Events, Error handling.

1) Libraries:

a. My Math lab Libraries:

Code:

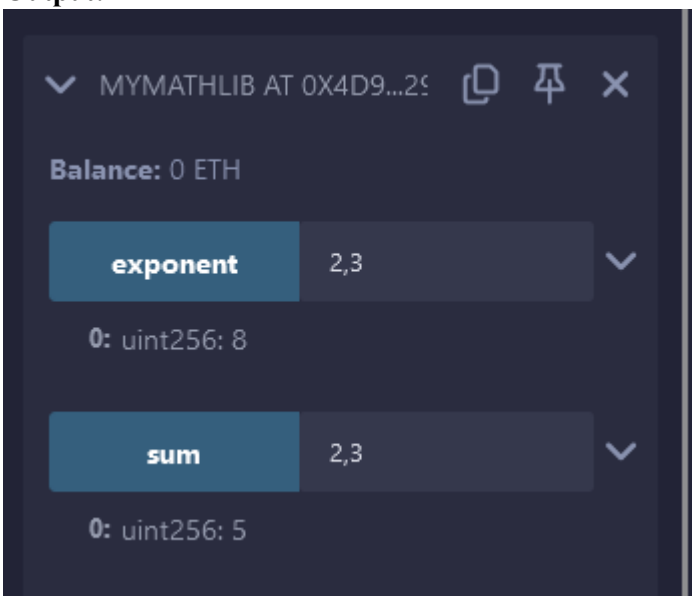
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

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

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

b. Using Libraries:

Code:

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

import "contracts/myLIB.sol";

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

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

Output:



2) Assembly:

Code:

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

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

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

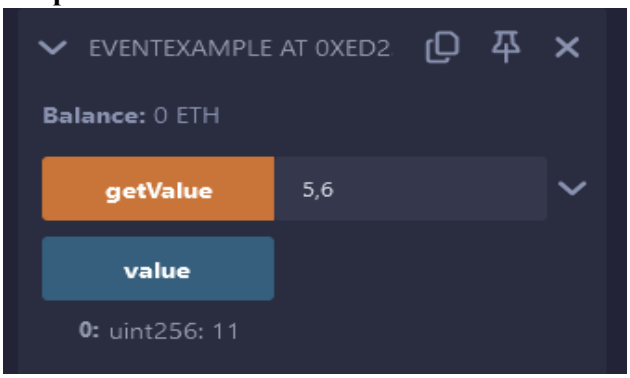
Output:

3) Events

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);  // Emitting the Increment event with the caller's address
      value = _a + _b;  // Updating the value state variable
   }
}
```

Output:

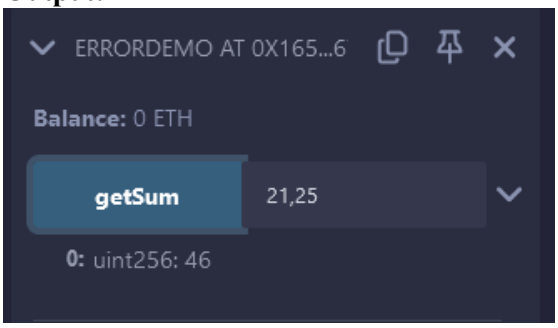


4)Error Handling:

Code:

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

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

Output:



Practical 5

Aim: Write a program to demonstrate the mining of ether.

Code:

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

Output:

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

Practical No: 6

Aim: Demonstrate the running of the blockchain node.

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

Genesis.json

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

CustomGenesis.json

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

Step 2-> Run command geth account new –datadir

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

testnet-blockchain.

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

Your new key was generated

Public address of the key:   0x77CB2BdBC0f1743bC73E92f1a8b1AB80BEDB35AE
Path of the secret key file: C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\key
store\UTC--2023-04-20T14-33-26.959134300Z--77cb2bdbc0f1743bc73e92f1a8b1ab80bedb35ae

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

Step 3-> Run command geth account new –datadir
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine

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

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

Step 4-> Run command geth --identity "localB" --http --http.port "8280" --http.corsdomain "*" --http.api "db,eth,net,web3" –datadir "C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine"

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

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

Step 5-> Run the command

miner.setEtherbase('0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0')

in the geth console

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

```
To exit, press ctrl-d or type exit
> INFO [04-20|20:29:45.021] Mapped network port                    proto=tcp extport=30303 intport=3030
NP IGDv1-IP1"

>
> miner.setEtherbase('0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0')
true
> miner.start()
INFO [04-20|20:34:45.673] Updated mining threads                 threads=4
INFO [04-20|20:34:45.674] Transaction pool price threshold updated price=1,000,000,000
null
> INFO [04-20|20:34:45.683] Commit new sealing work                number=1 sealhash=2e6f57..6db9c6 unc
=0 fees=0 elapsed=7.571ms
INFO [04-20|20:34:45.686] Commit new sealing work                number=1 sealhash=2e6f57..6db9c6 uncle
 fees=0 elapsed=9.940ms
INFO [04-20|20:34:47.975] Generating DAG in progress             epoch=0 percentage=0 elapsed=1.636s
INFO [04-20|20:34:49.873] Generating DAG in progress             epoch=0 percentage=1 elapsed=3.534s
```

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

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

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

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

Practical 7

Aim: Create your own blockchain and demonstrate its use.

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

Code:

```
// npm install crypto-js
const SHA256=require("crypto-js/sha256");
 class Block{
 constructor(index,timestamp,data,previousHash=""){
  this.index=index;
  this.timestamp=timestamp;
  this.data=data;
  this.previousHash=previousHash;
  this.hash=this.calculateHash();
 }
 calculateHash(){
  return SHA256(
   this.index+
    this.previousHash+
    this.timestamp+
    JSON.stringify(this.data)
  ).toString();
 }
 }
 class Blockchain{
 constructor(){
  this.chain=[this.createGenesisBlock()];
 }
 createGenesisBlock(){
  return new Block(0,"09/06/2024","GenesisBlock","0");
 }
 getLatestBlock(){
  return this.chain[this.chain.length-1];
 }
 addBlock(newBlock){
  newBlock.previousHash=this.getLatestBlock().hash;
   newBlock.hash=newBlock.calculateHash();
  this.chain.push(newBlock);
 }
 isChainValid(){
  for(leti=1;i<this.chain.length;i++){
   constcurrentBlock = this.chain[i];
   constpreviousBlock = this.chain[i-1];
   if(currentBlock.hash != currentBlock.calculateHash()){
    returnfalse;
   }
   if(currentBlock.previousHash != previousBlock.hash){
    return false;
```

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

Output:

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