INDEX

Practical - 1

1.A.

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

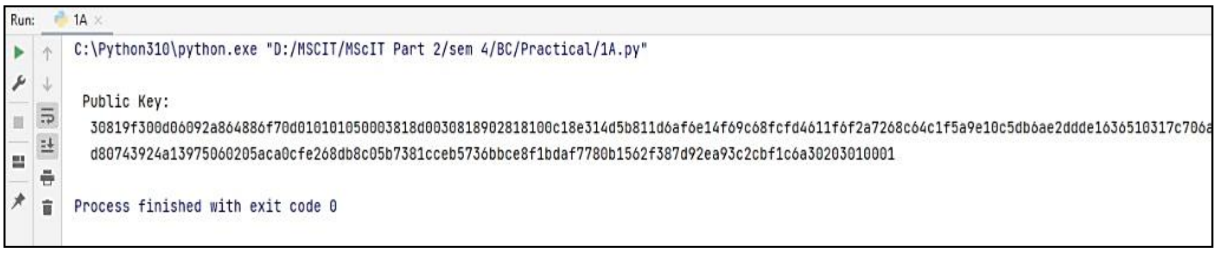
Theory:

RSA in blockchain refers to the utilization of the RSA cryptographic algorithm within a blockchain network. It involves using RSA for digital signatures, key management, and encryption purposes. RSA can provide security, authenticity, and privacy features to enhance the overall robustness of a blockchain system.

Code:

```
import binascii
import Crypto
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")


Dinesh = Client()
print("\n Public Key:",Dinesh.identity)
```

Output:



```
C:\Python310\python.exe "D:/MSCIT/MScIT Part 2/sem 4/BC/Practical/1A.py"

 Public Key:
  30819f300d06092a864886f70d010101050003818d0030818902818100c18e314d5b811d6af6e14f69c68fcfd4611f6f2a7268c64c1f5a9e10c5db6ae2ddde1636510317c706a
  d80743924a13975060205aca0cfe268db8c05b7381cceb5736bbce8f1bdaf7780b1562f387d92ea93c2cbf1c6a30203010001

Process finished with exit code 0
```

1.B.

Aim: A transaction class to send and receive money and test it.

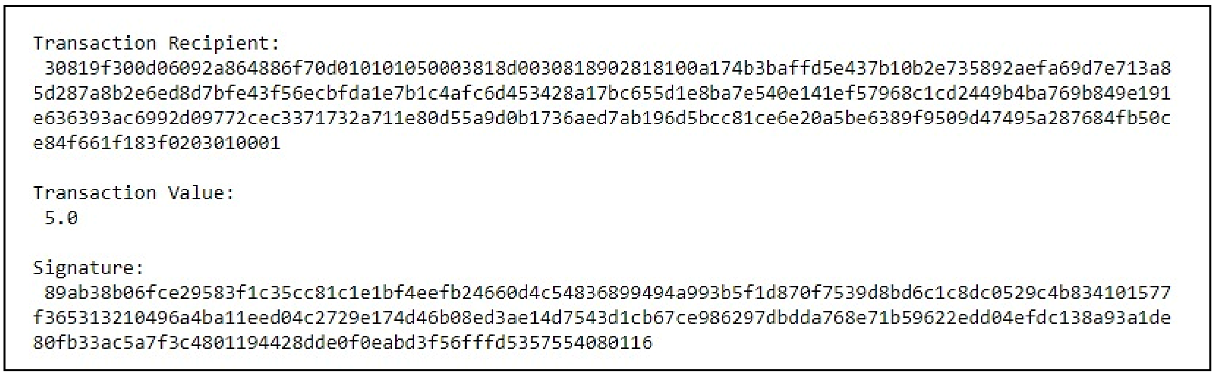
Theory:

In a blockchain, a transaction class represents a unit of data that encapsulates information about a specific transaction within the network. It typically includes details such as the sender, recipient, amount, and any additional metadata relevant to the transaction. Transactions are the fundamental building blocks of a blockchain, representing the transfer of assets or data between participants. They are validated, recorded, and stored in blocks, forming an immutable and transparent transaction history on the blockchain.

Code:

```
import collections
import datetime

from Crypto.Hash import SHA
from A import *

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):
        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('-----')

Dinesh = Client()
Ramesh = Client()


t = Transaction(Dinesh, Ramesh.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:
 30819f300d06092a864886f70d010101050003818d0030818902818100a174b3baffd5e437b10b2e735892aefa69d7e713a8
5d287a8b2e6ed8d7bfe43f56ecbfda1e7b1c4afc6d453428a17bc655d1e8ba7e540e141ef57968c1cd2449b4ba769b849e191
e636393ac6992d09772cec3371732a711e80d55a9d0b1736aed7ab196d5bcc81ce6e20a5be6389f9509d47495a287684fb50c
e84f661f183f0203010001

Transaction Value:
 5.0

Signature:
 89ab38b06fce29583f1c35cc81c1e1bf4eefb24660d4c54836899494a993b5f1d870f7539d8bd6c1c8dc0529c4b834101577
f365313210496a4ba11eed04c2729e174d46b08ed3ae14d7543d1cb67ce986297dbdda768e71b59622edd04efdc138a93a1de
80fb33ac5a7f3c4801194428dde0f0eabd3f56fffd5357554080116
```

1.C.

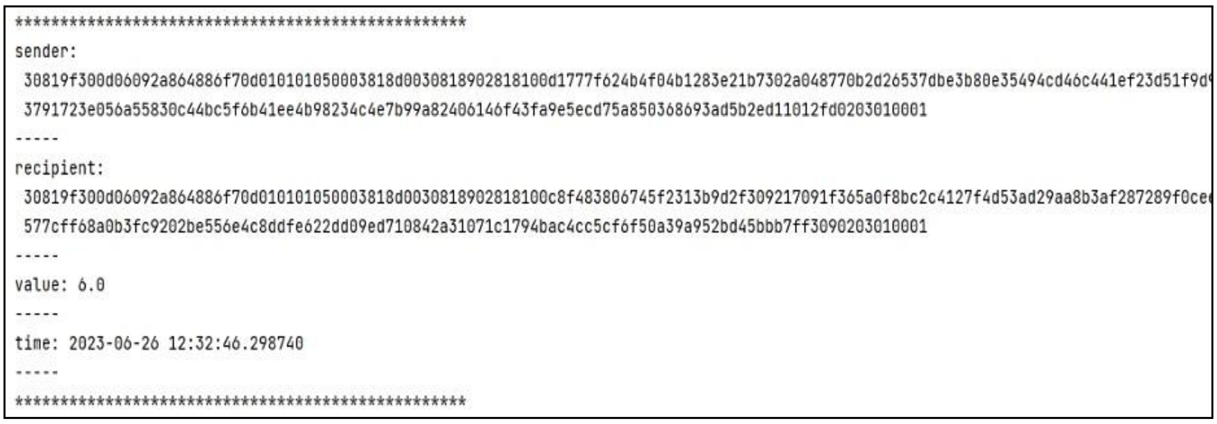
Aim: Create multiple transactions and display them.

Code:

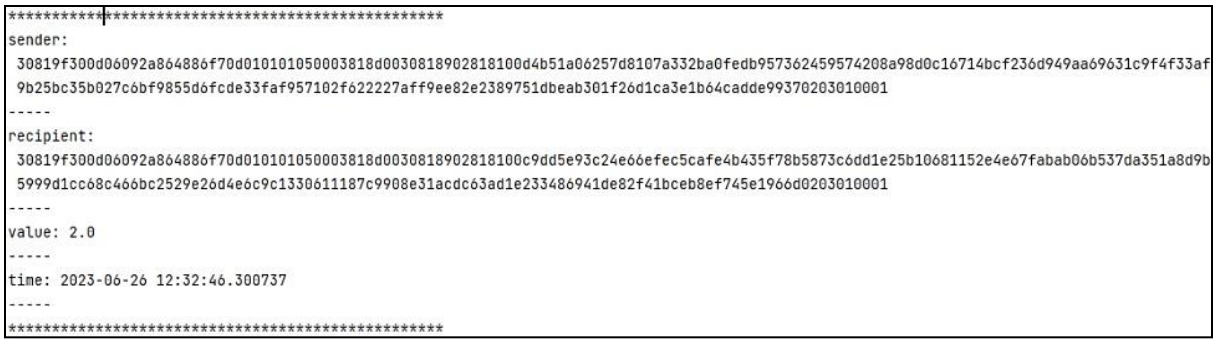
```
from B import *

Dinesh = Client()
Ramesh = Client()

t = Transaction(Dinesh, Ramesh.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)

Dinesh = Client()
Ramesh = Client()
Seema = Client()
Vijay = Client()

t1 = Transaction(Dinesh, Ramesh.identity, 15.0)
t1.sign_transaction()
transactions = [t1]
t2 = Transaction(Dinesh, Seema.identity, 6.0)
```

```
t2.sign_transaction()
transactions.append(t2)
t3 = Transaction(Ramesh, Vijay.identity, 2.0)
t3.sign_transaction()
transactions.append(t3)
t4 = Transaction(Seema, Ramesh.identity, 4.0)
t4.sign_transaction()
transactions.append(t4)
for transaction in transactions:
    Transaction.display_transaction(transaction)
    print("*"*50)
```

Output:



```
**************************************************
sender:
 30819f300d06092a864886f70d010101050003818d0030818902818100d1777f624b4f04b1283e21b7302a048770b2d26537dbe3b80e35494cd46c441ef23d51f9d
 3791723e056a55830c44bc5f6b41ee4b98234c4e7b99a82406146f43fa9e5ecd75a850368693ad5b2ed11012fd0203010001
-----
recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100c8f483806745f2313b9d2f309217091f365a0f8bc2c4127f4d53ad29aa8b3af287289f0ce
 577cff68a0b3fc9202be556e4c8ddfe622dd09ed710842a31071c1794bac4cc5cf6f50a39a952bd45bbb7ff3090203010001
-----
value: 6.0
-----
time: 2023-06-26 12:32:46.298740
-----
**************************************************
```



```
**************************************************
sender:
 30819f300d06092a864886f70d010101050003818d0030818902818100d4b51a06257d8107a332ba0fedb957362459574208a98d0c16714bcf236d949aa69631c9f4f33af
 9b25bc35b027c6bf9855d6fcde33faf957102f622227aff9ee82e2389751dbeab301f26d1ca3e1b64cadde99370203010001
-----
recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100c9dd5e93c24e66efec5cafe4b435f78b5873c6dd1e25b10681152e4e67fabab06b537da351a8d9b
 5999d1cc68c466bc2529e26d4e6c9c1330611187c9908e31acdc63ad1e233486941de82f41bceb8ef745e1966d0203010001
-----
value: 2.0
-----
time: 2023-06-26 12:32:46.300737
-----
**************************************************
```

1.D.

Aim: Create a blockchain, a genesis block and execute it.

Theory:

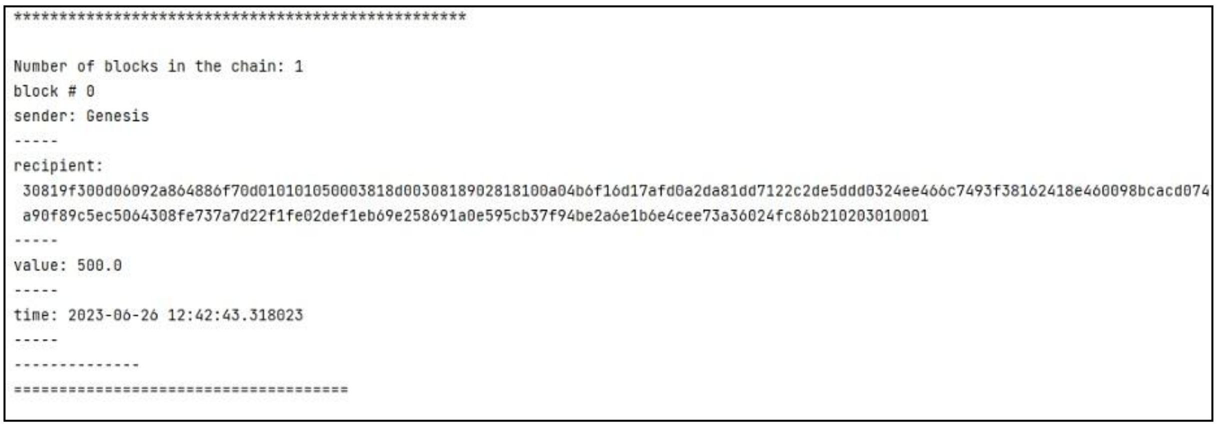
A blockchain genesis block, or simply the genesis block, is the very first block in a blockchain network. It serves as the foundation from which subsequent blocks are added. The genesis block is typically hard-coded or pre-defined by the blockchain's creator or initial participants.

Code:

```
from C import *
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("=====================================")

Dinesh = Client()
t0 = Transaction("Genesis", Dinesh.identity, 500.0)

block0 = Block(Dinesh)
block0.previous_block_hash = ""
NONCE = None
```

```
block0.verified_transactions.append(t0)
digest = hash(block0)
last_block_hash = digest

TPCoins = []
TPCoins.append (block0)
dump_blockchain(TPCoins)
```

Output:



```
**************************************************

Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient:
 30819f300d06092a864886f70d010101050003818d0030818902818100a04b6f16d17afd0a2da81dd7122c2de5ddd0324ee466c7493f38162418e460098bcacd074
 a90f89c5ec5064308fe737a7d22f1fe02def1eb69e258691a0e595cb37f94be2a6e1b6e4cee73a36024fc86b210203010001
-----
value: 500.0
-----
time: 2023-06-26 12:42:43.318023
-----
--------------
=====================================
```

1.E.

Aim: Create a mining function and test it.

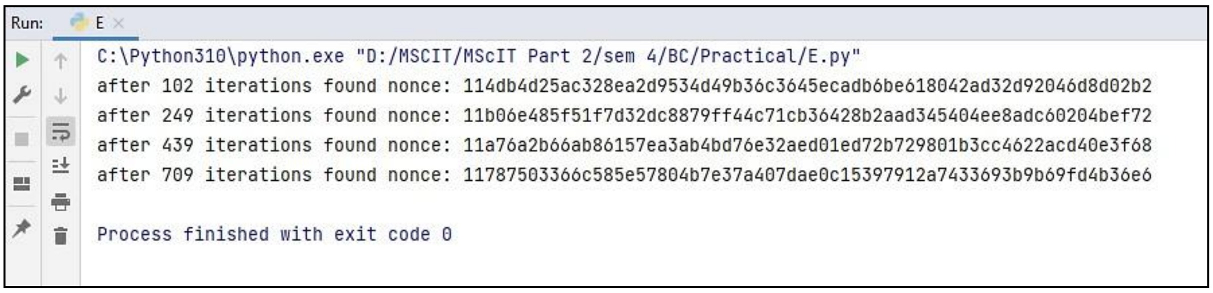
Theory:

the term "mining function" typically refers to the specific algorithm or computational process used by miners to solve complex mathematical problems or puzzles. The mining function is designed to be resource-intensive, requiring significant computational power and energy consumption.

Code:

```
from D import *
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 print(digest)

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

Output:



```
C:\Python310\python.exe "D:/MSCIT/MScIT Part 2/sem 4/BC/Practical/E.py"
after 102 iterations found nonce: 114db4d25ac328ea2d9534d49b36c3645ecadb6be618042ad32d92046d8d02b2
after 249 iterations found nonce: 11b06e485f51f7d32dc8879ff44c71cb36428b2aad345404ee8adc60204bef72
after 439 iterations found nonce: 11a76a2b66ab86157ea3ab4bd76e32aed01ed72b729801b3cc4622acd40e3f68
after 709 iterations found nonce: 11787503366c585e57804b7e37a407dae0c15397912a7433693b9b69fd4b36e6

Process finished with exit code 0
```

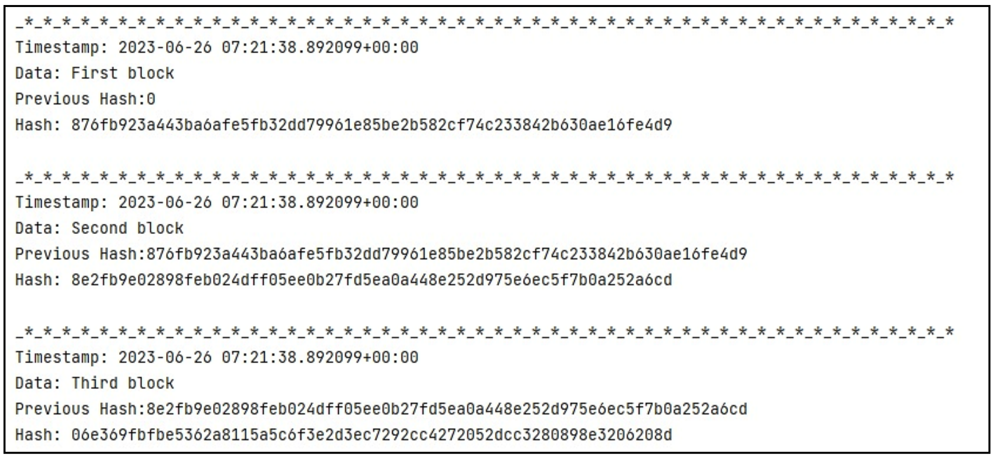
1.F.

Aim: Add blocks to the miner and dump the blockchain.

Code:

```
from E import *

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("_*"*50)
    print(f"Timestamp: {block.timestamp}\nData: {block.data}\nPrevious
Hash:{block.previous_hash}\nHash: {block.hash}\n")
```

Output:



```
_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
Timestamp: 2023-06-26 07:21:38.892099+00:00
Data: First block
Previous Hash:0
Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9

_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
Timestamp: 2023-06-26 07:21:38.892099+00:00
Data: Second block
Previous Hash:876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9
Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd

_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
Timestamp: 2023-06-26 07:21:38.892099+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.

Theory:

Go Ethereum (Geth) and Mist Browser are two software components that are commonly associated with the Ethereum blockchain platform.

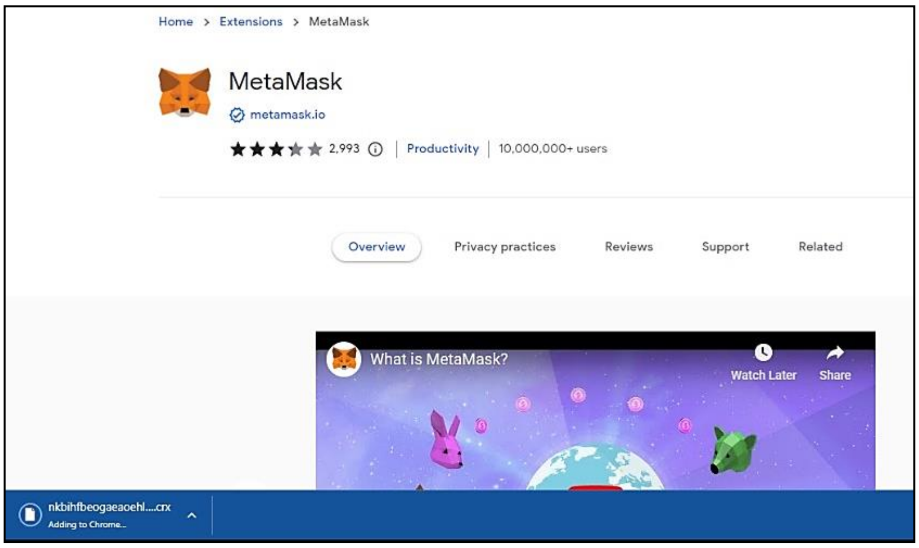
Go Ethereum (Geth):

Geth is an implementation of the Ethereum protocol written in the Go programming language. It serves as a command-line interface (CLI) tool and a full node client for the Ethereum network. Geth allows users to interact with the Ethereum blockchain, create accounts, deploy smart contracts, and execute transactions. It provides features for mining, syncing with the blockchain, and participating in network consensus. Geth is widely used by developers, miners, and users who want to interact directly with the Ethereum network.

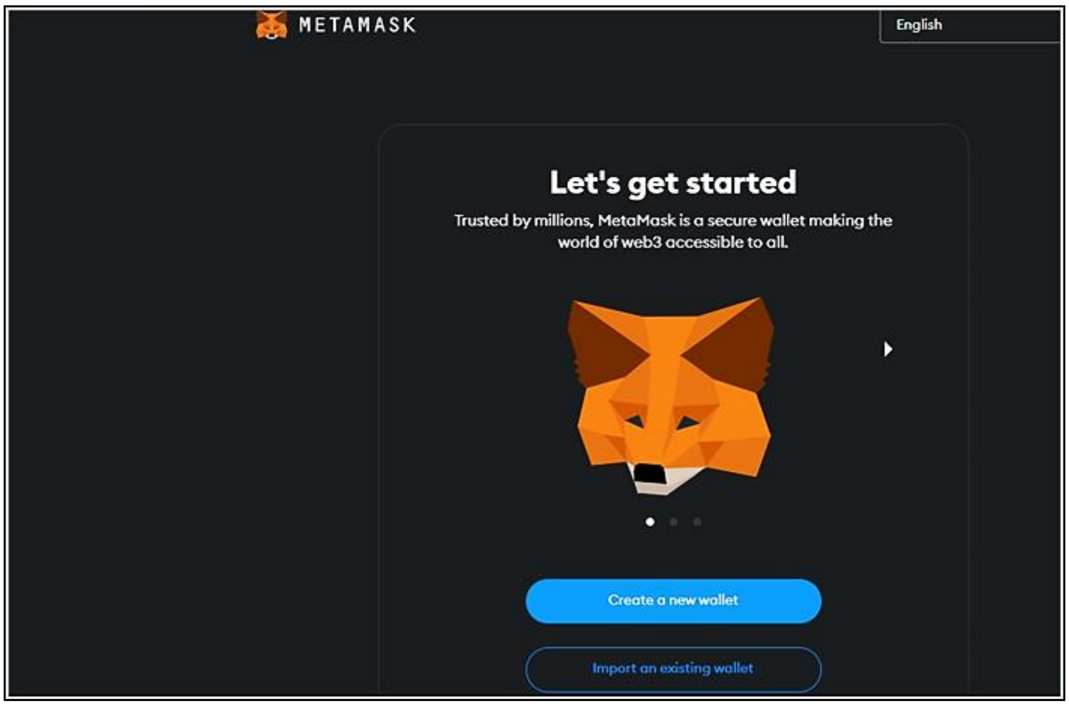
Mist Browser:

Mist is an Ethereum decentralized application (dApp) browser, providing a graphical user interface (GUI) for accessing and interacting with dApps on the Ethereum network. It offers a user-friendly environment for managing Ethereum accounts, exploring smart contracts, and interacting with decentralized applications. Mist Browser integrates with the Geth client to connect to the Ethereum network, enabling users to browse and interact with the decentralized web.

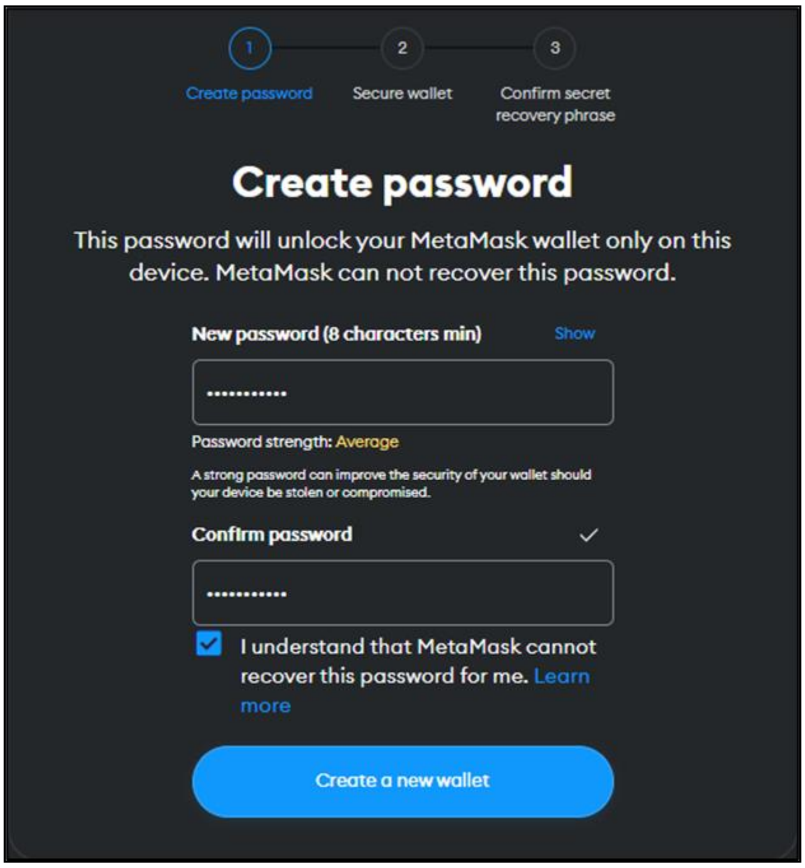
Step 1 : Install MetaMask extension for chrome from Chrome Web Store



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


METAMASK
English
Let's get started
Trusted by millions, MetaMask is a secure wallet making the world of web3 accessible to all.
Create a new wallet
Import an existing wallet

Step 3: Create a password. This password can be used only on the device itwas created on. Create a Strong password and click on Create a new Wallet button



1 Create password — 2 Secure wallet — 3 Confirm secret recovery phrase

Create password

This password will unlock your MetaMask wallet only on this device. MetaMask can not recover this password.

New password (8 characters min) Show

•••••••••••

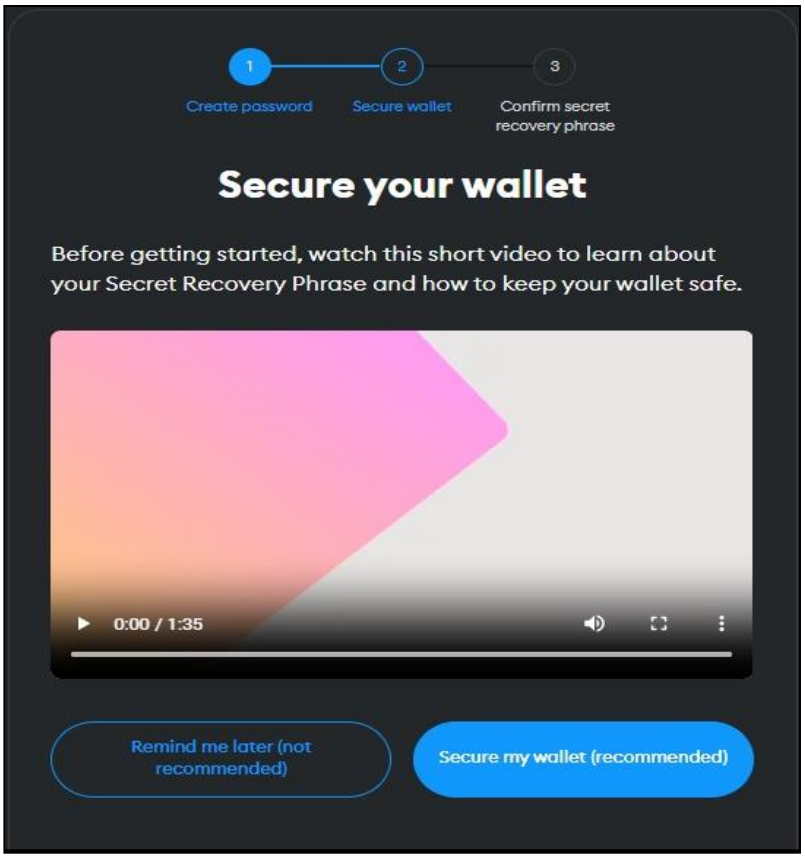
Password strength: Average

A strong password can improve the security of your wallet should your device be stolen or compromised.

Confirm password ✓

•••••••••••

☑ I understand that MetaMask cannot recover this password for me. Learn more

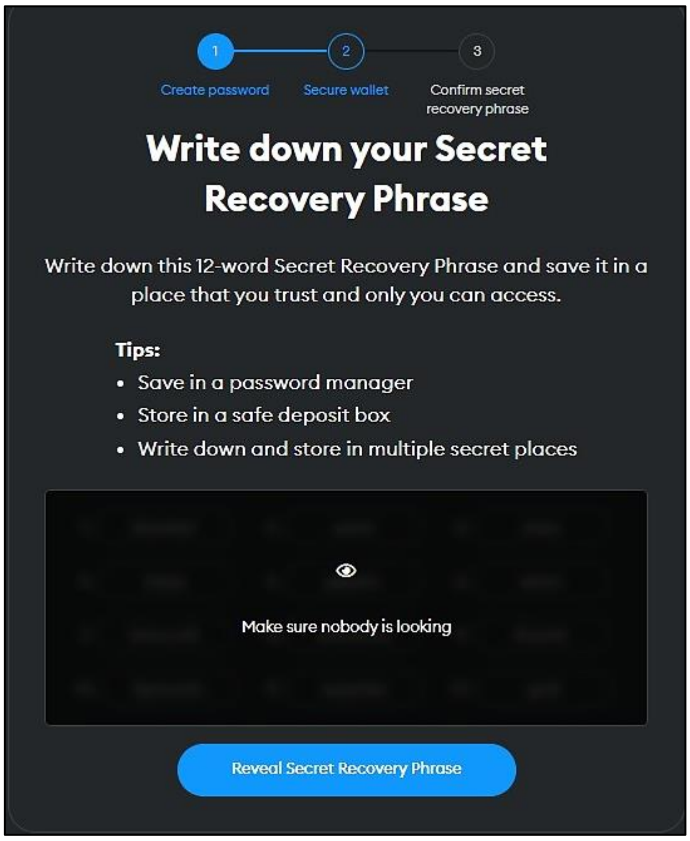

1
2
3
Create password
Secure wallet
Confirm secret recovery phrase

Secure your wallet

Before getting started, watch this short video to learn about your Secret Recovery Phrase and how to keep your wallet safe.

0:00 / 1:35
Remind me later (not recommended)
Secure my wallet (recommended)

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


1
2
3
Create password
Secure wallet
Confirm secret recovery phrase

Write down your Secret Recovery Phrase

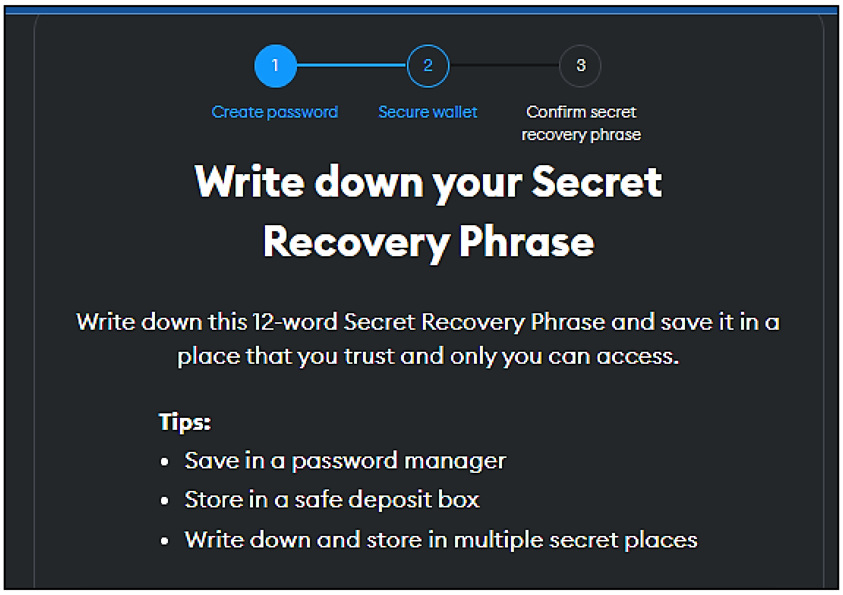
Write down this 12-word Secret Recovery Phrase and save it in a place that you trust and only you can access.

Tips:

- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places

Make sure nobody is looking
Reveal Secret Recovery Phrase

Step 5: Click on Reveal Secret Recovery Phrase button and save the words inthe same sequence



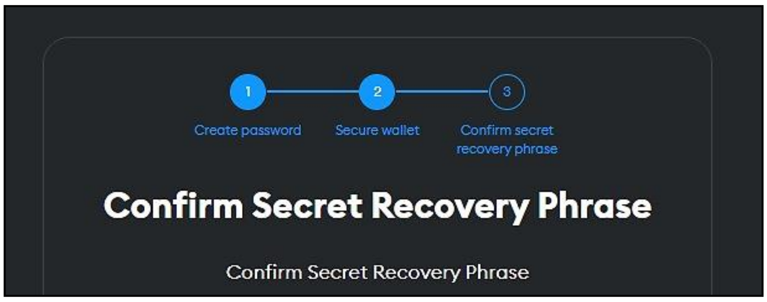
Write down your Secret Recovery Phrase

Write down this 12-word Secret Recovery Phrase and save it in a place that you trust and only you can access.

Tips:

- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places

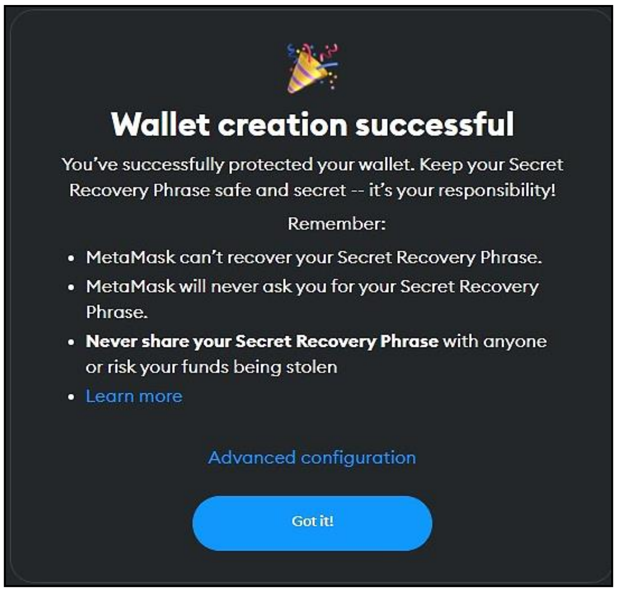
Step 6: Enter the respective words in the empty positions and click Confirm.



Confirm Secret Recovery Phrase

Confirm Secret Recovery Phrase

Step 7: Click Got it!



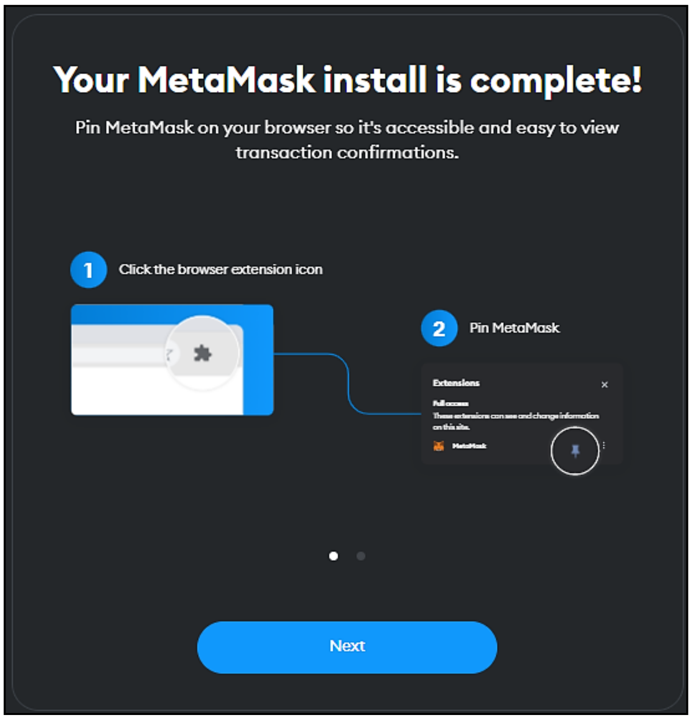
Wallet creation successful

You've successfully protected your wallet. Keep your Secret Recovery Phrase safe and secret -- it's your responsibility!

Remember:

- MetaMask can't recover your Secret Recovery Phrase.
- MetaMask will never ask you for your Secret Recovery Phrase.
- **Never share your Secret Recovery Phrase** with anyone or risk your funds being stolen
- Learn more

Step 8: Click on Next



Your MetaMask install is complete!

Pin MetaMask on your browser so it's accessible and easy to view transaction confirmations.

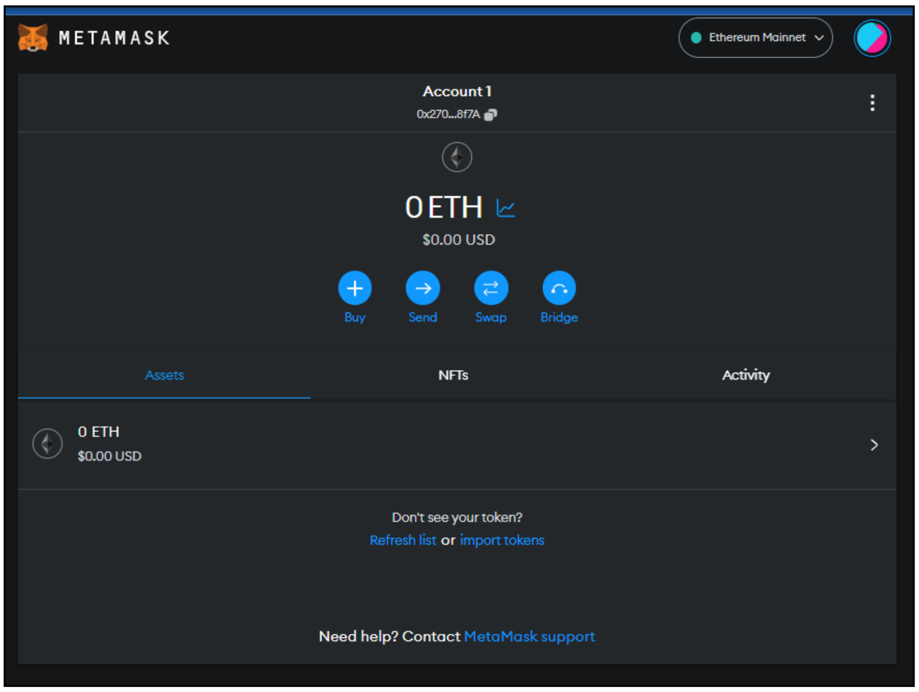
1 Click the browser extension icon

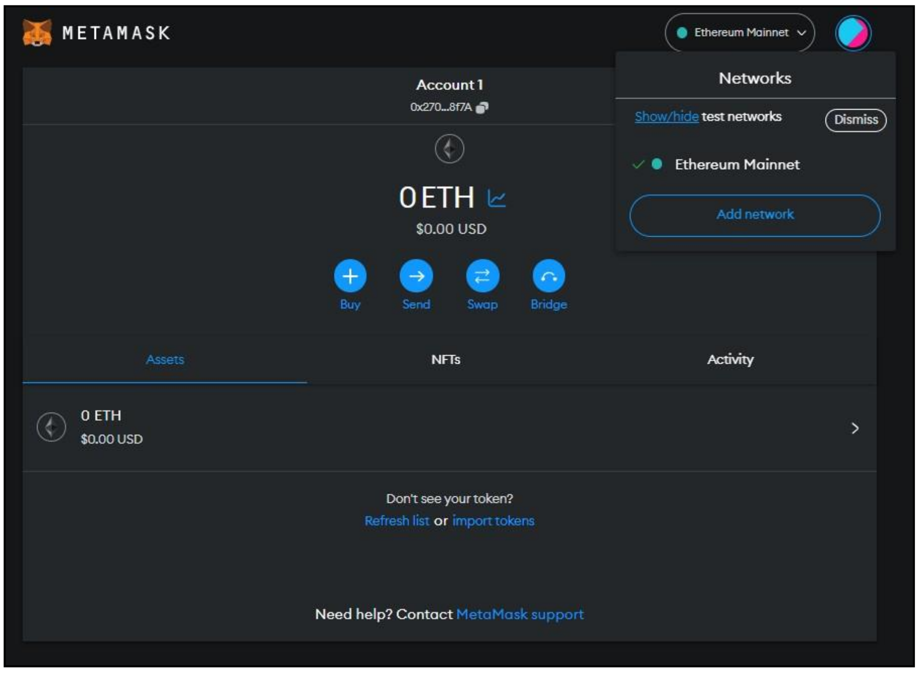
2 Pin MetaMask

Extensions
Full access
These extensions can see and change information on this site.
MetaMask
Next

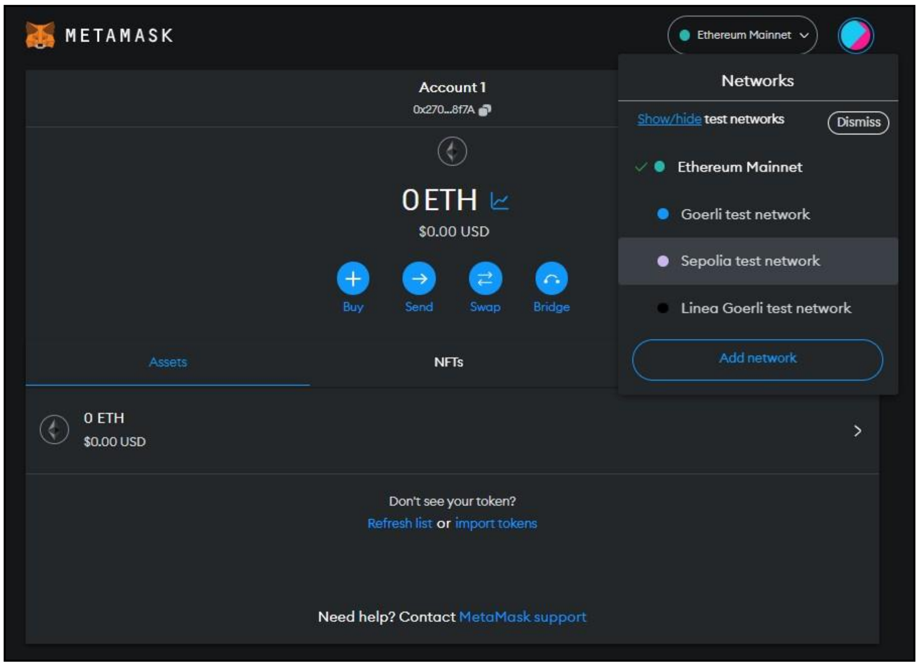
Step 9 : Following will be the Dashboard

Step 10: Click on Ethereum Mainnet button. Next click on Show/hide testnetworks.

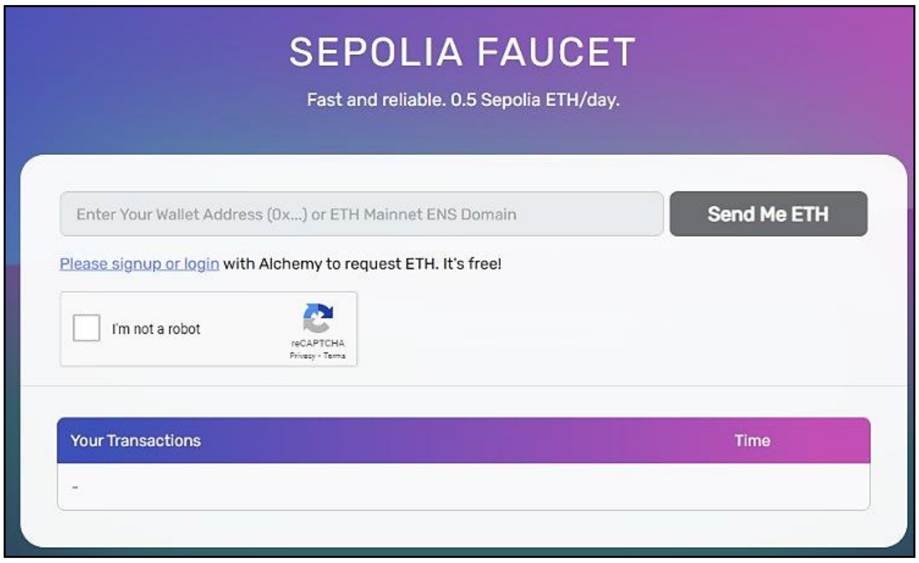

METAMASK
Ethereum Mainnet
Account 1
0x270...8f7A
0 ETH
$0.00 USD
Buy Send Swap Bridge
Assets NFTs Activity
0 ETH
$0.00 USD
Don't see your token?
Refresh list or import tokens
Need help? Contact MetaMask support


METAMASK
Ethereum Mainnet
Account 1
0x270...8f7A
0 ETH
$0.00 USD
Buy Send Swap Bridge
Assets NFTs Activity
0 ETH
$0.00 USD
Don't see your token?
Refresh list or import tokens
Need help? Contact MetaMask support
Networks
Show/hide test networks Dismiss
✓ Ethereum Mainnet
Add network

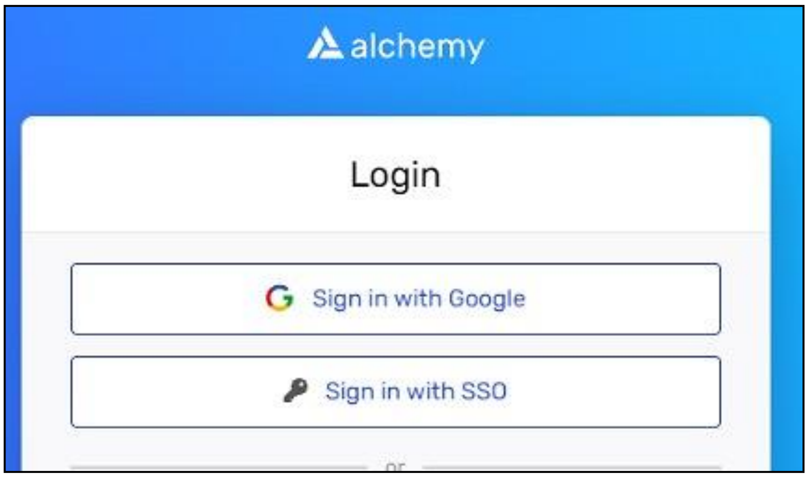
Step 11: Check if tesnets are shown by clicking on Etherum Mainnet button. Clickon Sepolia test network.



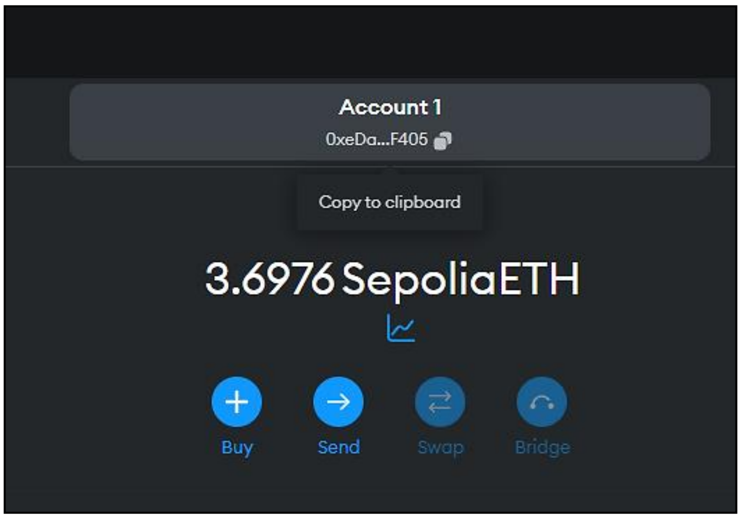
Step 12: Go to https://sepoliafaucet.com/ and Click on Alchemy Login button.


SEPOLIA FAUCET
Fast and reliable. 0.5 Sepolia ETH/day.
Enter Your Wallet Address (0x...) or ETH Mainnet ENS Domain
Send Me ETH
Please signup or login with Alchemy to request ETH. It's free!
I'm not a robot
reCAPTCHA
Privacy - Terms
Your Transactions
Time
-

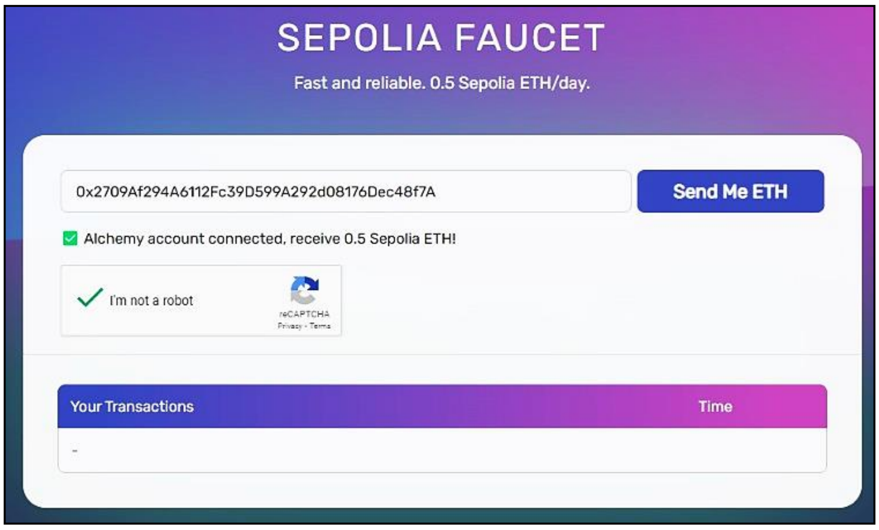
Step 13: Login to a gmail account in another browser tab and click on Sign in withGoogle



Step 14: Now go to MetaMask and copy the account address.



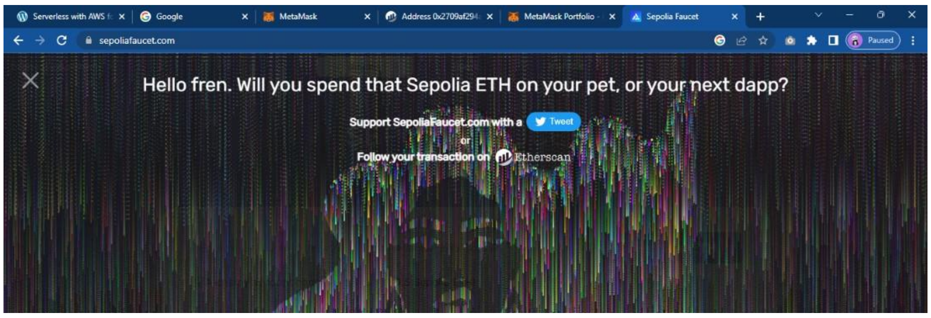
Step 15: Paste the address and click on Send Me ETH.



0x2709Af294A6112Fc39D599A292d08176Dec48f7A

Alchemy account connected, receive 0.5 Sepolia ETH!

Step 16: Your ETH transfer is succesfull. You should see a similar animation.

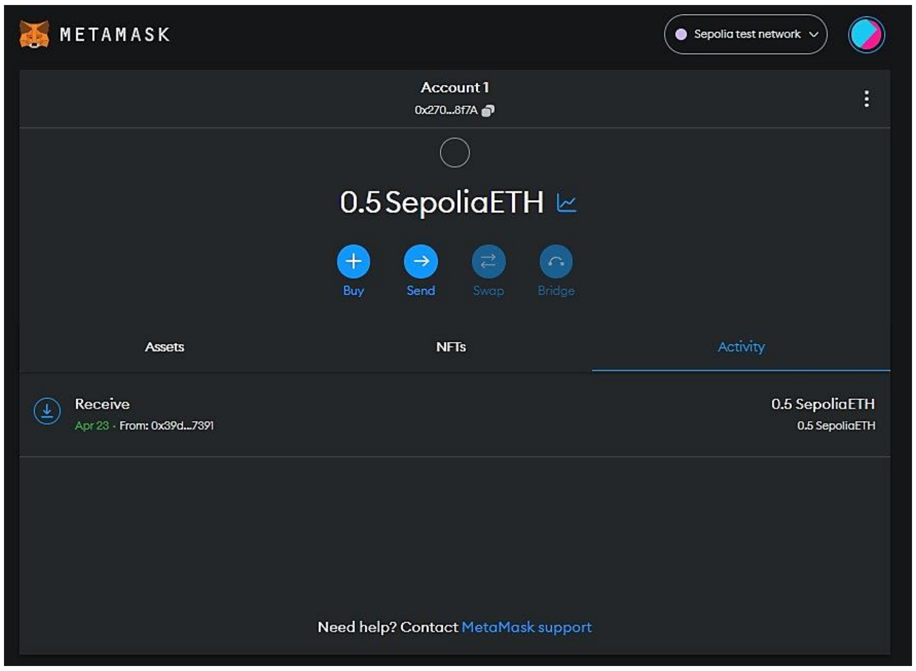

Hello fren. Will you spend that Sepolia ETH on your pet, or your next dapp?

Support SepoliaFaucet.com with a Tweet

or

Follow your transaction on Etherscan

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



0.5 SepoliaETH

Receive
Apr 23 · From: 0x39d...7391
0.5 SepoliaETH
0.5 SepoliaETH

Need help? Contact MetaMask support

Practical - 3

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

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

Theory:

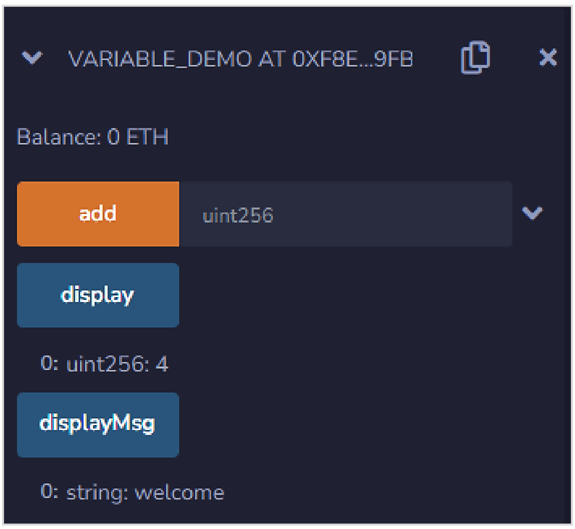
- **Variables:** Containers that store and manipulate data.
- **Operators:** Symbols that perform operations on variables and values.
- **Loops:** Structures that repeat a block of code until a certain condition is met.
- **Decision Making:** Constructs that allow the execution of different code blocks based on specified conditions.
- **Strings:** Data types used to represent and manipulate text.
- **Arrays:** Containers that store collections of elements of the same type.
- **Enums:** User-defined data types that represent a set of named values.
- **Structs:** Custom data types that allow you to define a structure with multiple properties.
- **Mappings:** Key-value data structures used to store and retrieve data efficiently.
- **Conversions:** Operations that transform variables from one type to another.
- **Ether Units:** Units of measurement used to represent and interact with Ethereum's native cryptocurrency, Ether.
- **Special Variables:** Pre-defined variables in Solidity that provide information or context about the contract's environment or execution state.

1.Variable

Code:

```
pragma solidity ^0.5.0;
contract variable_demo {
uint256 sum = 4; //state variable
uint256 x;
address a;
string s = "welcome";
function add(uint256) public {
uint256 y = 2; //local variable sum = sum+x+y:
sum = sum + x + y;
}
function display() public view returns (uint256) {
return sum;
}
function displayMsg() public view returns (string memory) {
return s;
}

}
```

Output:



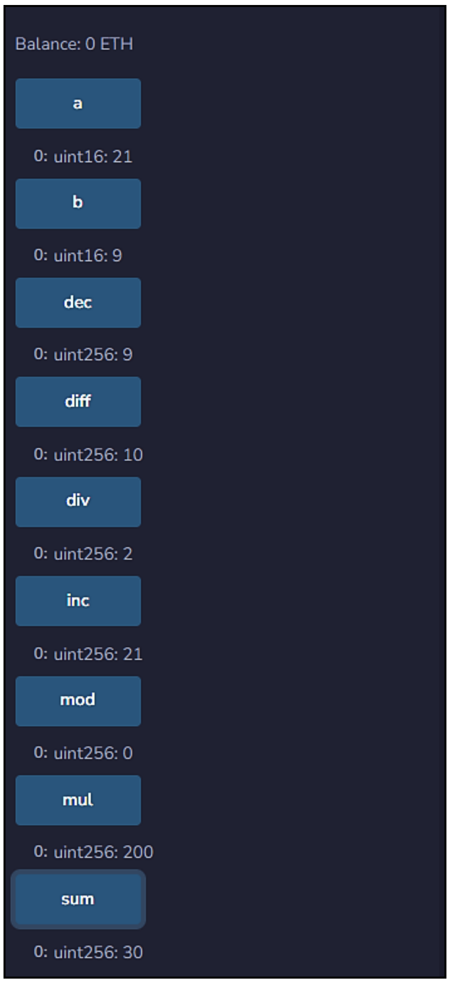
```
VARIABLE_DEMO AT 0XF8E...9FB
Balance: 0 ETH
add uint256
display
0: uint256: 4
displayMsg
0: string: welcome
```

2.Operators

Code:

```
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:


```
Balance: 0 ETH
a
0: uint16: 21
b
0: uint16: 9
dec
0: uint256: 9
diff
0: uint256: 10
div
0: uint256: 2
inc
0: uint256: 21
mod
0: uint256: 0
mul
0: uint256: 200
sum
0: uint256: 30
```

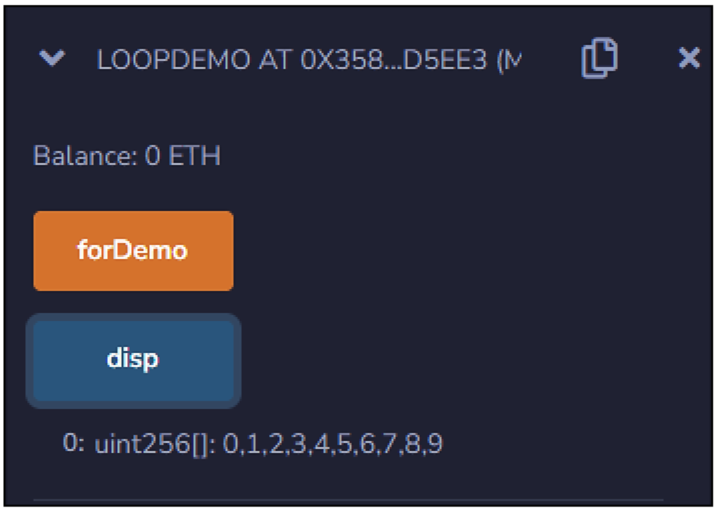
3.Loops

Code:

for Loop:

```
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:



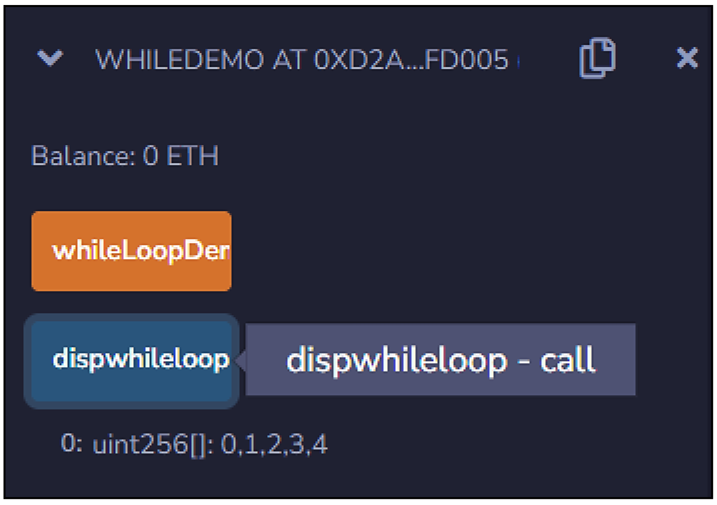
```
0: uint256[]: 0,1,2,3,4,5,6,7,8,9
```

3.a. 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:



```
Balance: 0 ETH
0: uint256[]: 0,1,2,3,4
```

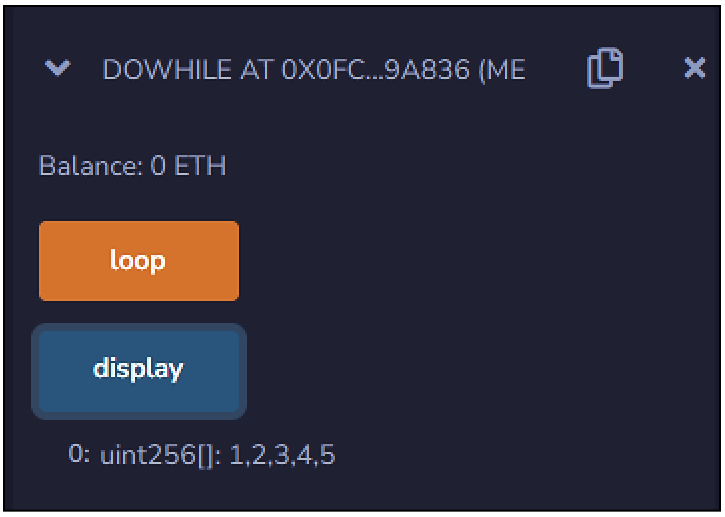
3.b. Do while loop:

Code:

```
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 display() public view returns(uint256[] memory){
return data;
}
}
```

Output:



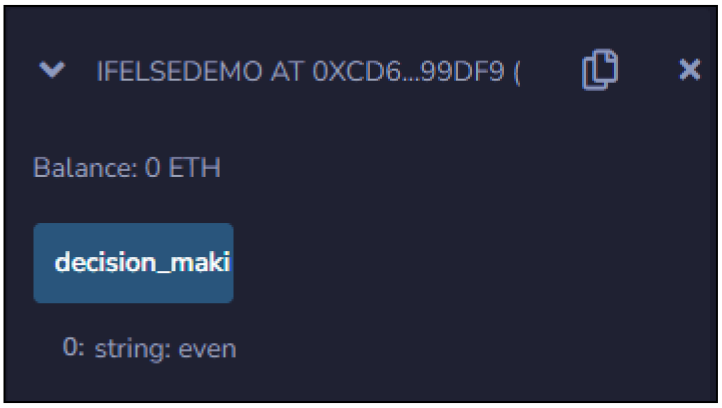
```
Balance: 0 ETH
0: uint256[]: 1,2,3,4,5
```

4.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:



```
IFELSEDEMO AT 0XCD6...99DF9 (
Balance: 0 ETH
decision_maki
0: string: even
```

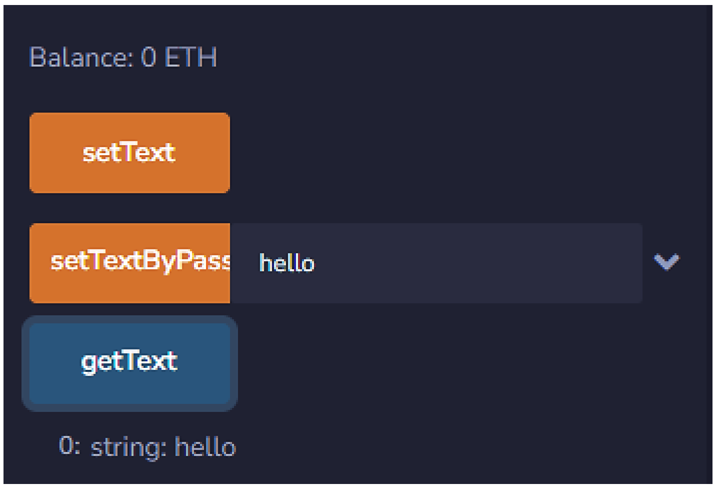
5.Strings

Code:

```
pragma solidity ^0.5.0;
contract LearningStrings {
string text;
function getText() public view returns (string memory) {
return text;
}
function setText() public {
text = "hello";
}
function setTextByPassing(string memory message) public {
```

```
text = message;
}
}
```

Output:



Balance: 0 ETH

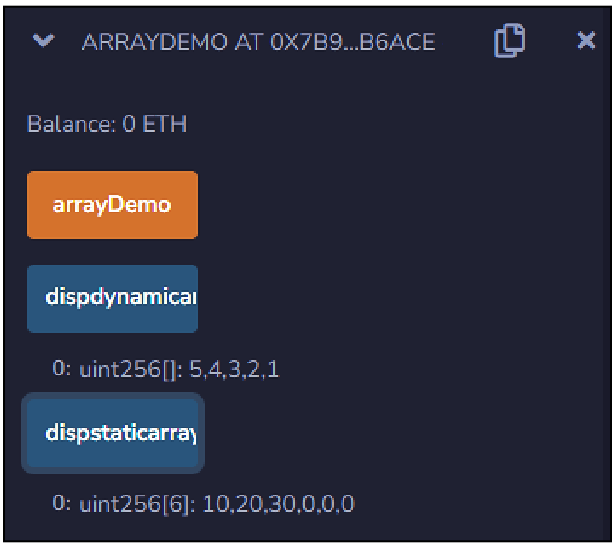
```
0: string: hello
```

6.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;
function arrayDemo() public
{
while(x>0)
{
arr1.push(x);
x=x-1;
}
}
function dispdynamicarray() public view returns(uint[] memory)
{
return arr1;
}
}
```

Output:



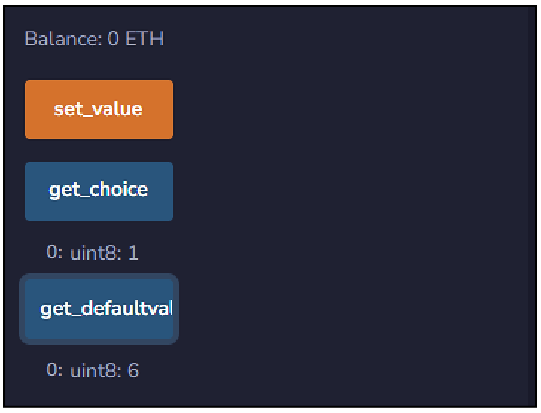
```
Balance: 0 ETH
0: uint256[]: 5,4,3,2,1
0: uint256[6]: 10,20,30,0,0,0
```

7.Enums

Code:

```
pragma solidity ^0.5.0;
contract enumdemo {
enum week_days {
Monday,
Tuesday,
Wednesday,
Thursday,
Friday,
Saturday,
Sunday}
week_days week;
week_days choice;
week_days constant default_value = week_days.Sunday;
function set_value() public {
choice = week_days.Tuesday;
}
function get_choice() public view returns (week_days) {
return choice;
}
function get_defaultvalue() public view returns (week_days) {
return default_value;
}
}
```

Output:



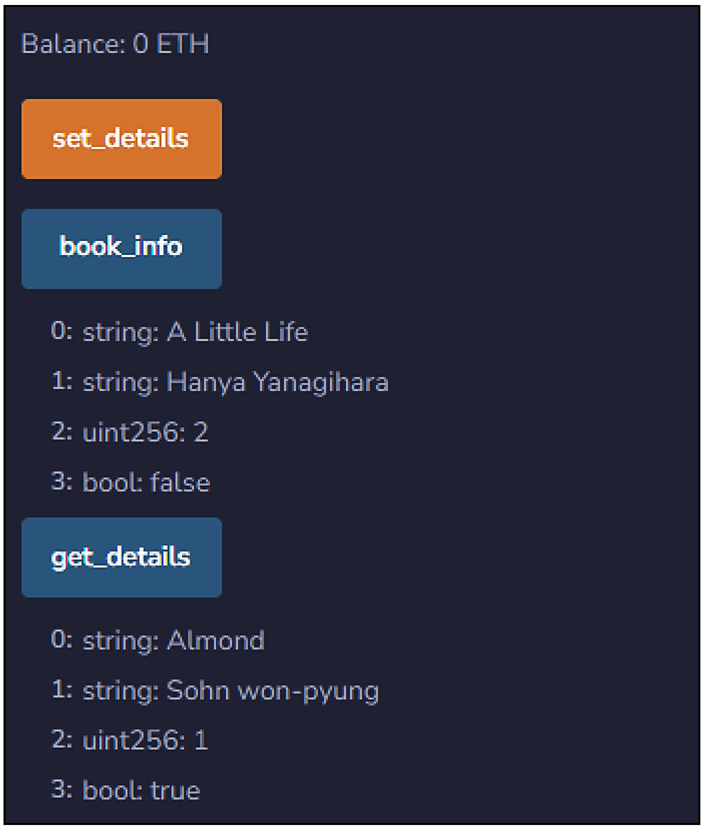
```
0: uint8: 1
0: uint8: 6
```

8.Struct

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:



```
Balance: 0 ETH

0: string: A Little Life
1: string: Hanya Yanagihara
2: uint256: 2
3: bool: false

0: string: Almond
1: string: Sohn won-pyung
2: uint256: 1
3: bool: true
```

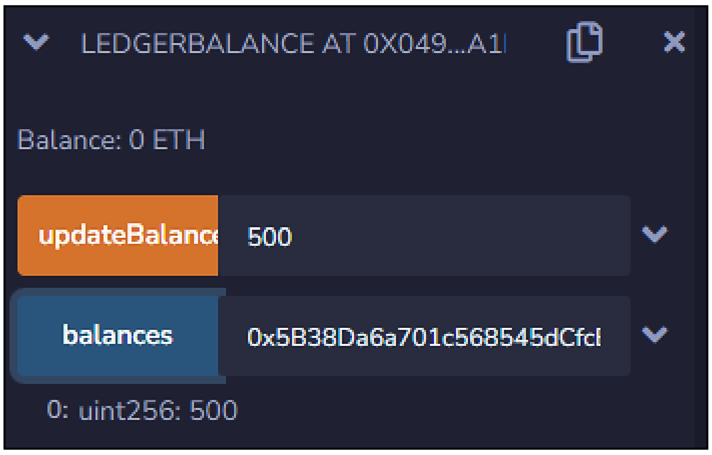
9.Mapping

Code:

```
pragma solidity ^0.5.0;
contract LedgerBalance {
mapping(address => uint256) public balances;
function updateBalance(uint256 newBalance) public {
```

```
balances[msg.sender] = newBalance;
}
}
contract Updater {
function updateBalance() public returns (uint256) {
LedgerBalance ledgerBalance = new LedgerBalance();
return ledgerBalance.balances(address(this));
}
}
```

Output:



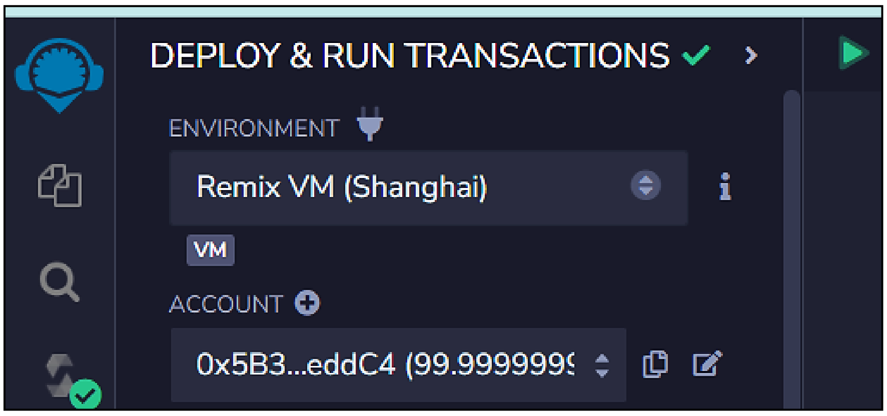
LEDGERBALANCE AT 0X049...A1

Balance: 0 ETH

updateBalanc 500

balances 0x5B38Da6a701c568545dCfcl

0: uint256: 500



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Remix VM (Shanghai)

VM

ACCOUNT

0x5B3...eddC4 (99.999999

(Use account info in balance function above as a address)

10.Conversions

Code:

(Implicit.sol)

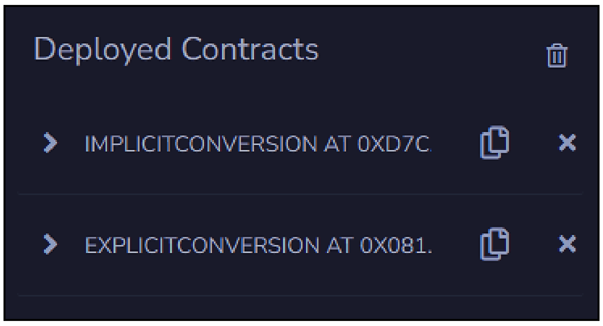
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
```

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

(explicit.sol)

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

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

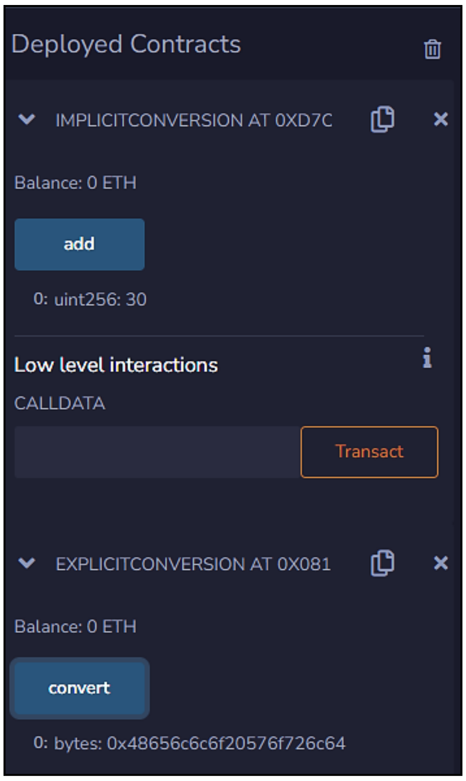
Output:



Deployed Contracts

IMPLICITCONVERSION AT 0XD7C.

EXPLICITCONVERSION AT 0X081.


Deployed Contracts

IMPLICITCONVERSION AT 0XD7C

Balance: 0 ETH

0: uint256: 30

Low level interactions

CALLDATA

EXPLICITCONVERSION AT 0X081

Balance: 0 ETH

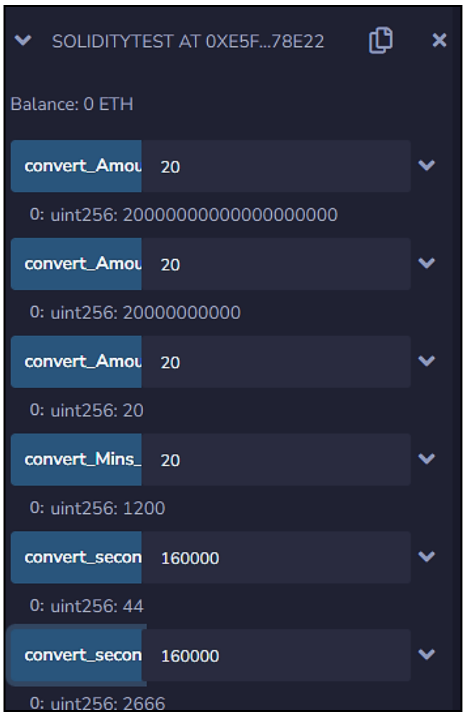
0: bytes: 0x48656c6c6f20576f726c64

11.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:



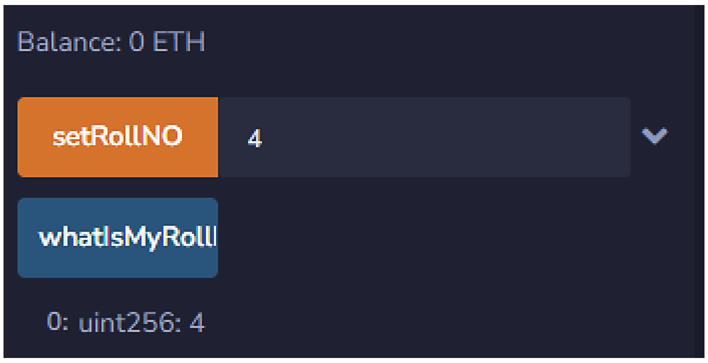
```
SOLIDITYTEST AT 0XE5F...78E22
Balance: 0 ETH
convert_Amou   20
0: uint256: 20000000000000000000
convert_Amou   20
0: uint256: 20000000000
convert_Amou   20
0: uint256: 20
convert_Mins_   20
0: uint256: 1200
convert_secon   160000
0: uint256: 44
convert_secon   160000
0: uint256: 2666
```

12.Special Variable

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:



```
0: uint256: 4
```

Practical - 4

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

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

Theory:

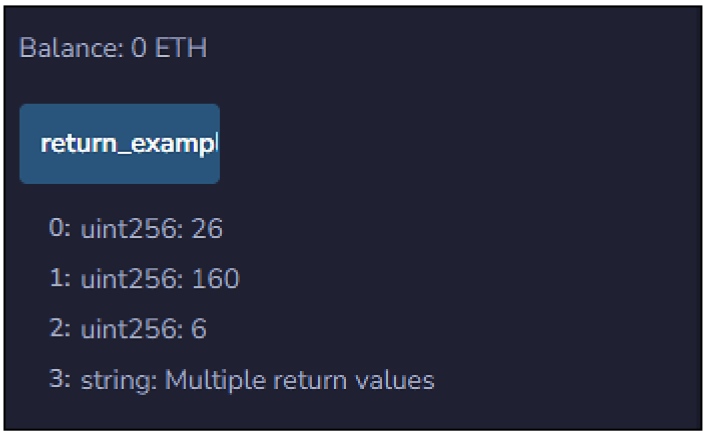
- **Functions:** Blocks of code that perform specific tasks and can be called and executed.
- **Function Modifiers:** Keywords that allow you to add additional behavior or restrictions to functions.
- **View Functions:** Functions that do not modify the state of the contract and only read data.
- **Pure Functions:** Functions that do not read or modify the contract's state and do not access external data.
- **Fallback Function:** A special function executed when a contract receives a transaction without any specific function call.
- **Function Overloading:** Defining multiple functions with the same name but different parameters.
- **Mathematical Functions:** Functions that perform mathematical operations, such as addition, subtraction, multiplication, etc.
- **Cryptographic Functions:** Functions that provide cryptographic functionalities, such as hash functions, digital signatures, etc.

1.Functions

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;
contract Test {
function return_example()
public
pure
returns (
uint256,
uint256,
uint256,
string memory
)
{
uint256 num1 = 10;
uint256 num2 = 16;
uint256 sum = num1 + num2;
uint256 prod = num1 * num2;
uint256 diff = num2 - num1;
string memory message = "Multiple return values";
return (sum, prod, diff, message);
}
}
```

Output:



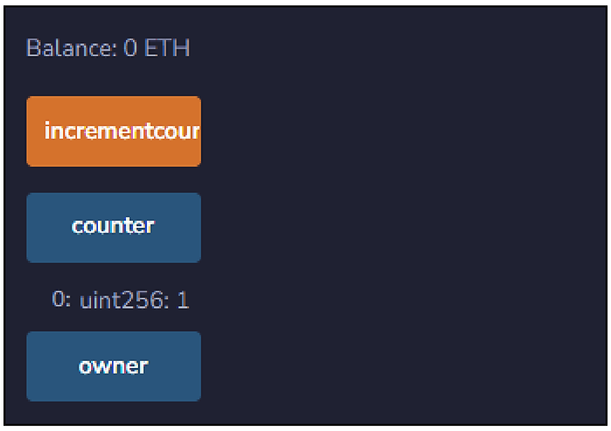
```
Balance: 0 ETH

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

2. 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:



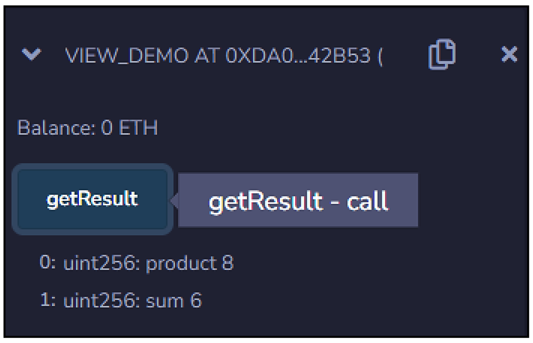
```
Balance: 0 ETH
0: uint256: 1
```

3.View function

Code:

```
pragma solidity ^0.5.0;
contract view_demo {
uint256 num1 = 2;
uint256 num2 = 4;
function getResult() public view returns (uint256 product, uint256 sum) {
product = num1 * num2;
sum = num1 + num2;
}
}
```

Output:



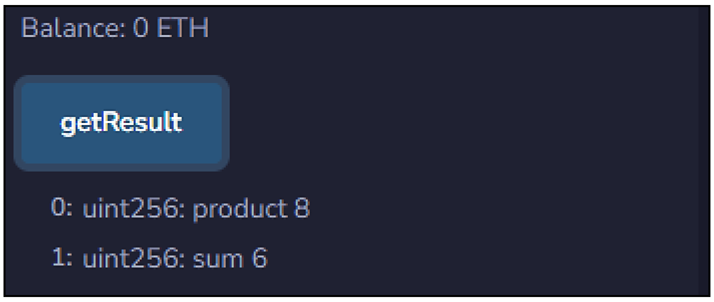
```
Balance: 0 ETH
0: uint256: product 8
1: uint256: sum 6
```

4.Pure function

Code:

```
pragma solidity ^0.5.0;
contract pure_demo {
function getResult() public pure returns (uint256 product, uint256 sum) {
uint256 num1 = 2;
uint256 num2 = 4;
product = num1 * num2;
sum = num1 + num2;
}
}
```

Output:



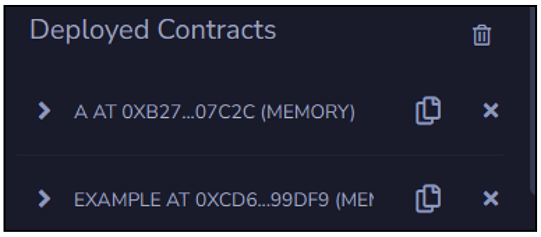
```
Balance: 0 ETH
getResult
0: uint256: product 8
1: uint256: sum 6
```

5.Fallback Fucntion

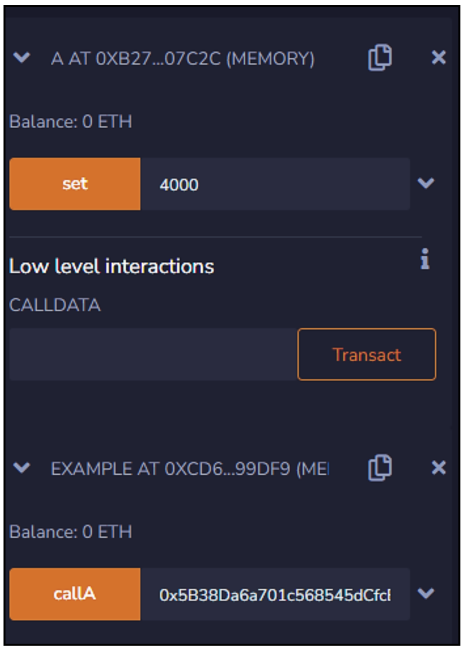
Code:

```
// 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:



```
Deployed Contracts
> A AT 0XB27...07C2C (MEMORY)
> EXAMPLE AT 0XCD6...99DF9 (MEI
```

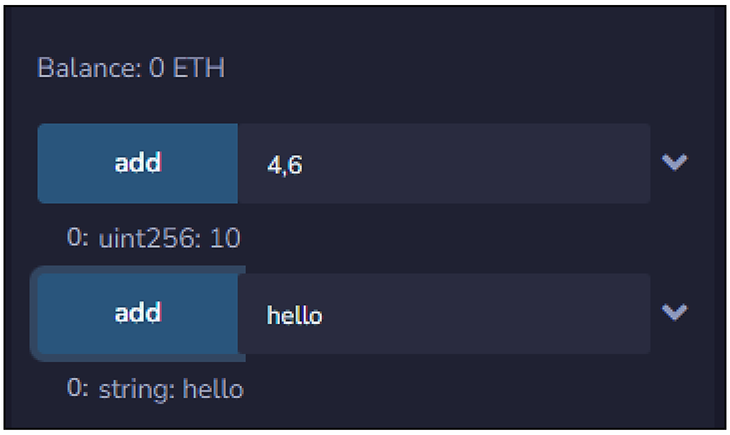

A AT 0XB27...07C2C (MEMORY)
Balance: 0 ETH
set
4000
Low level interactions
CALLDATA
Transact
EXAMPLE AT 0XCD6...99DF9 (ME
Balance: 0 ETH
callA
0x5B38Da6a701c568545dCfcl

6. 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:

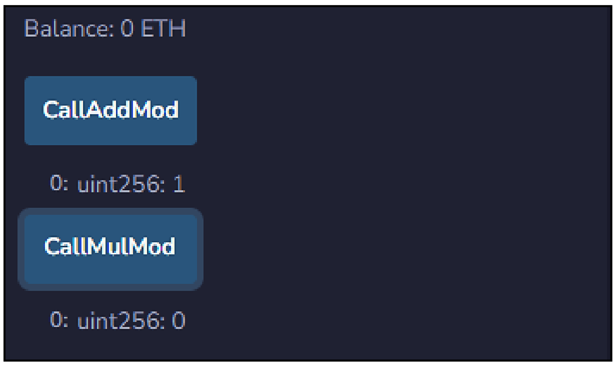

```
Balance: 0 ETH
add    4,6
0: uint256: 10
add    hello
0: string: hello
```

7.Mathematical Function

Code:

```
pragma solidity ^0.5.0;
contract Test{
function CallAddMod() public pure returns(uint){
return addmod(7,3,3);
}
function CallMulMod() public pure returns(uint){
return mulmod(7,3,3);
}
}
```

Output:

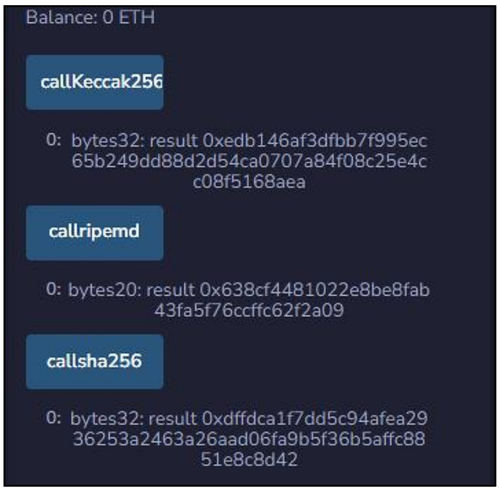

```
Balance: 0 ETH
CallAddMod
0: uint256: 1
CallMulMod
0: uint256: 0
```

8. Cryptographic function

Code:

```
pragma solidity ^0.5.0;
contract Test{
function callKeccak256() public pure returns(bytes32 result){
return keccak256("BLOCKCHAIN");
}
function callsha256() public pure returns(bytes32 result){
return sha256("BLOCKCHAIN");
}
function callripemd() public pure returns (bytes20 result){
return ripemd160("BLOCKCHAIN");
}
}
```

Output:



```
Balance: 0 ETH

callKeccak256

0: bytes32: result 0xedb146af3dfbb7f995ec
   65b249dd88d2d54ca0707a84f08c25e4c
   c08f5168aea

callripemd

0: bytes20: result 0x638cf4481022e8be8fab
   43fa5f76ccffc62f2a09

callsha256

0: bytes32: result 0xdffdca1f7dd5c94afea29
   36253a2463a26aad06fa9b5f36b5affc88
   51e8c8d42
```

Practical - 5

Aim: Implement and demonstrate the use of the following in Solidity.

Withdrawal Pattern, Restricted Access.

Theory:

Withdrawal Pattern:

A design pattern used in smart contracts to handle the withdrawal of funds from a contract, often involving the use of separate "withdraw" functions to facilitate secure and controlled fund transfers.

The withdrawal pattern in the context of blockchain refers to a specific behavior or strategy followed by participants in a decentralized application (DApp) or smart contract to withdraw their funds. It involves the process of retrieving or transferring digital assets or tokens from a specific account or contract address. The withdrawal pattern can vary depending on the rules and conditions set within the blockchain protocol or smart contract.

Restricted Access:

A mechanism implemented in smart contracts to restrict access to certain functions or data based on predefined conditions or permission levels, ensuring that only authorized parties can interact with specific contract functionalities or information.

Restricted access in blockchain refers to the implementation of limitations or restrictions on certain actions or functionalities within a blockchain network. These restrictions can be applied to specific user roles, accounts, or smart contracts to control access to certain features or data. Restricted access mechanisms can include permissions, access control lists (ACLs), or smart contract conditions that enforce specific rules on who can interact with certain blockchain assets or perform particular operations.

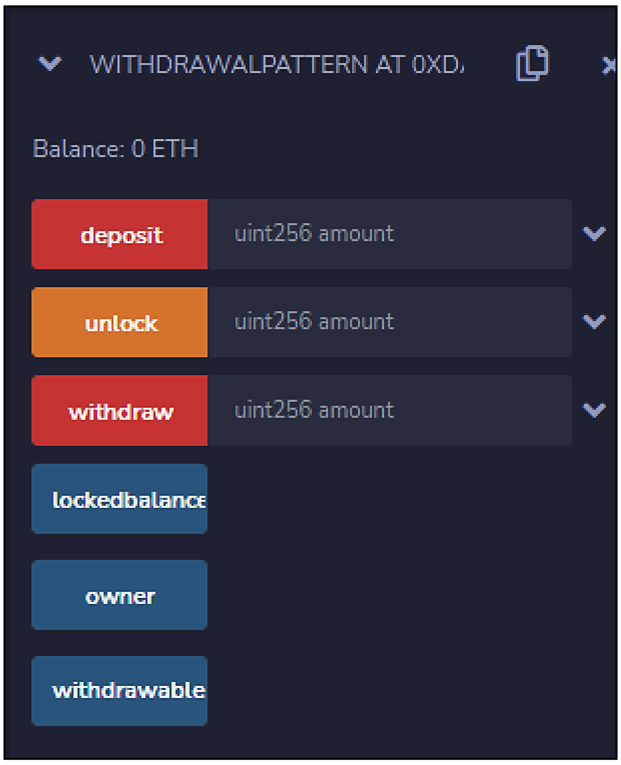
This helps enhance security, privacy, and governance within the blockchain network by ensuring that only authorized entities have access to sensitive functions or information.

1. **Withdrawal pattern**

Code:

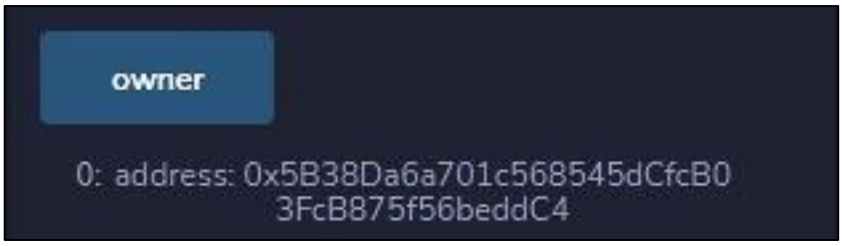
```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.18;

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:



Flow of execution

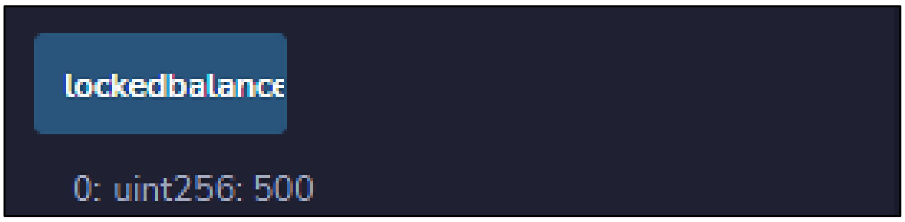
Step 1: Click on owner



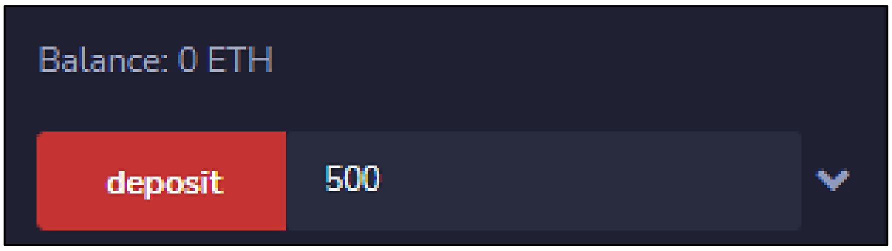
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

Step 2 : Enter an amount and click on deposit

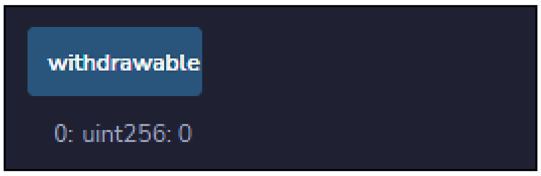
Step 3: Click on locked balance button to display the locked amount in the account



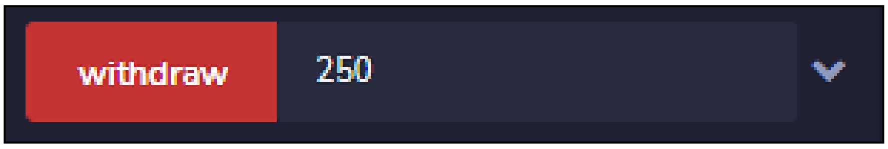
0: uint256: 500



Step 4: Click on withdrawable balance button

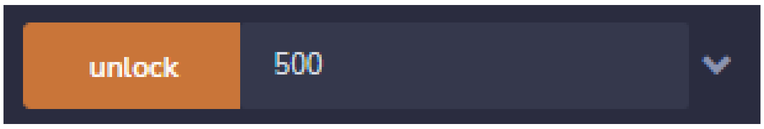


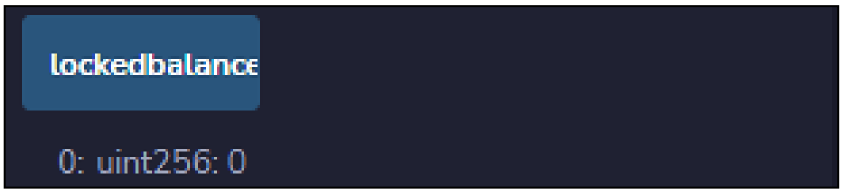
```
0: uint256: 0
```



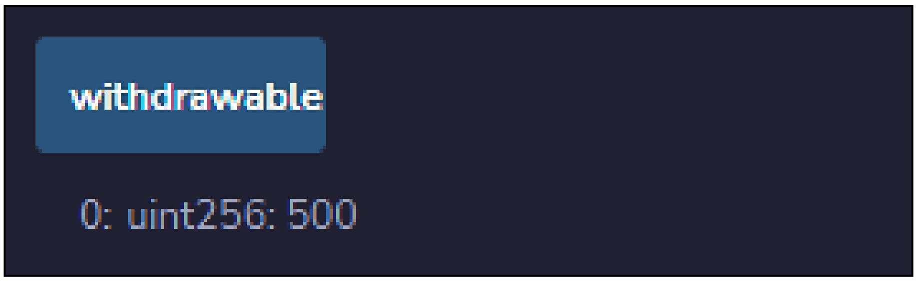
Step 5: Click on unlock button and enter any amount to transfer amount to withdrawable balance. Check locked balance and withdrawable balance.

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.

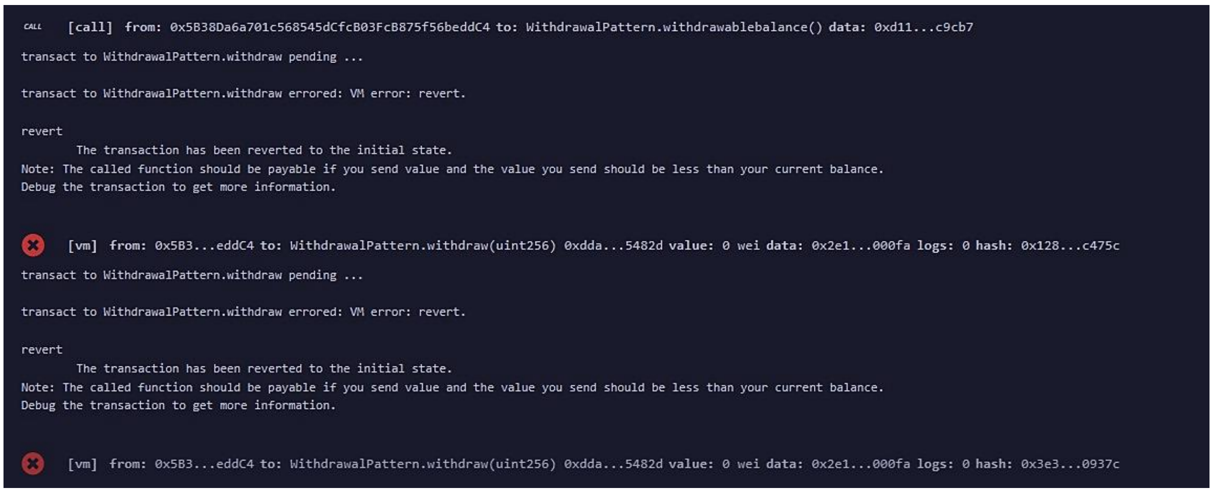




```
0: uint256: 0
```



```
0: uint256: 500
```



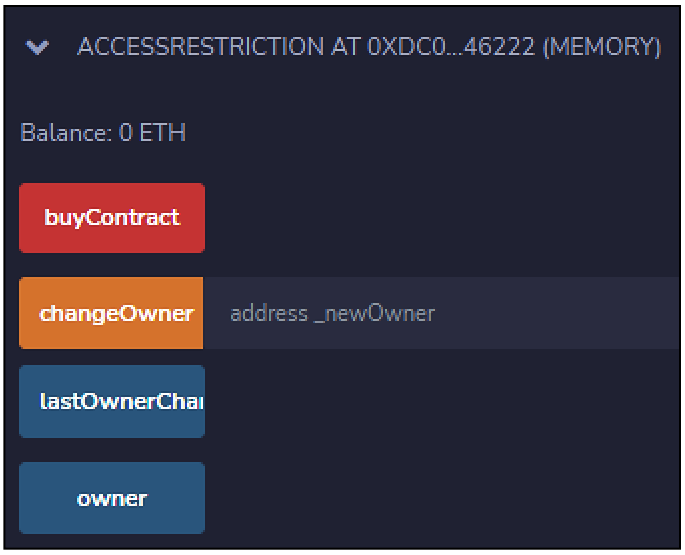
```
CALL   [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: WithdrawalPattern.withdrawablebalance() data: 0xd11...c9cb7
transact to WithdrawalPattern.withdraw pending ...

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

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


       [vm]  from: 0x5B3...eddC4 to: WithdrawalPattern.withdraw(uint256) 0xdda...5482d value: 0 wei data: 0x2e1...000fa logs: 0 hash: 0x128...c475c
transact to WithdrawalPattern.withdraw pending ...

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

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


       [vm]  from: 0x5B3...eddC4 to: WithdrawalPattern.withdraw(uint256) 0xdda...5482d value: 0 wei data: 0x2e1...000fa logs: 0 hash: 0x3e3...0937c
```

2.Restrict action

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;
}
}
```

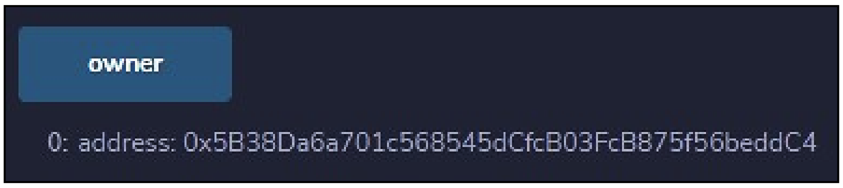
Output:



Balance: 0 ETH

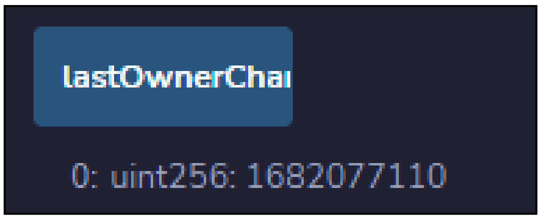
Flow of execution

Step 1-> Click on owner to create an owner object



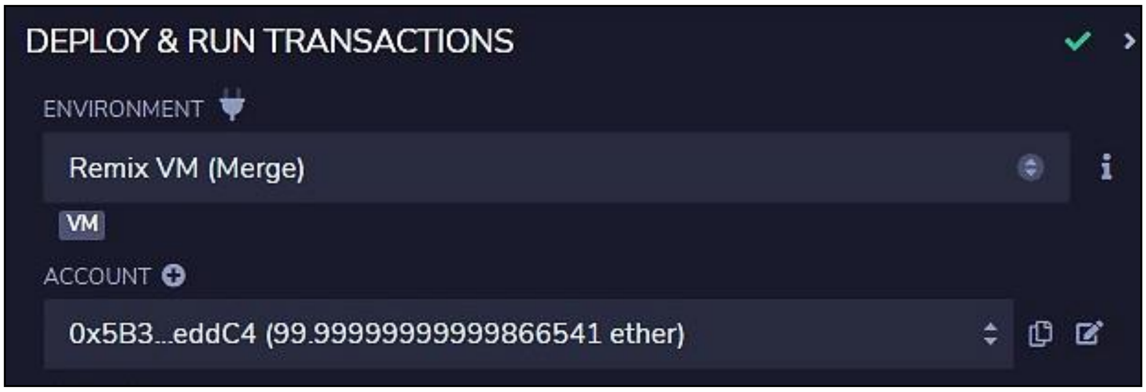
```
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
```

Step 2-> Click on lastOwnerChange button



```
0: uint256: 1682077110
```

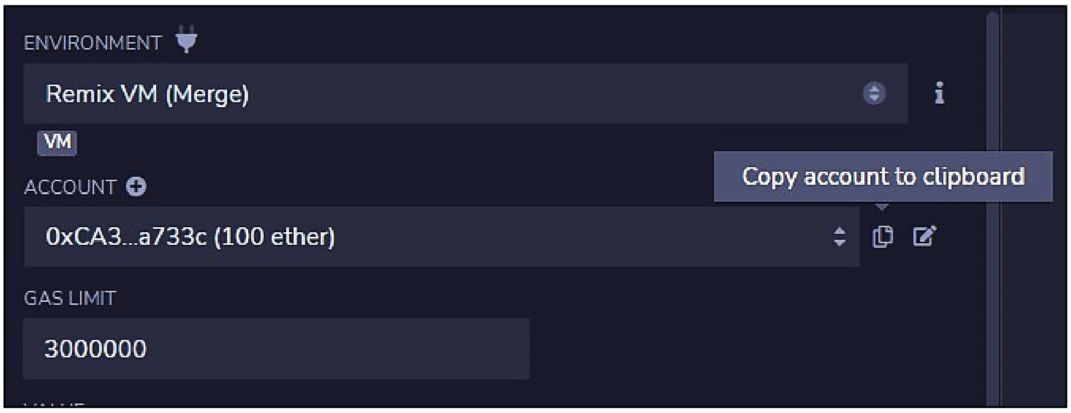
Step 3-> Change the address of the account from Account dropdown in Deploy tab of Remix IDE.



ENVIRONMENT: Remix VM (Merge)

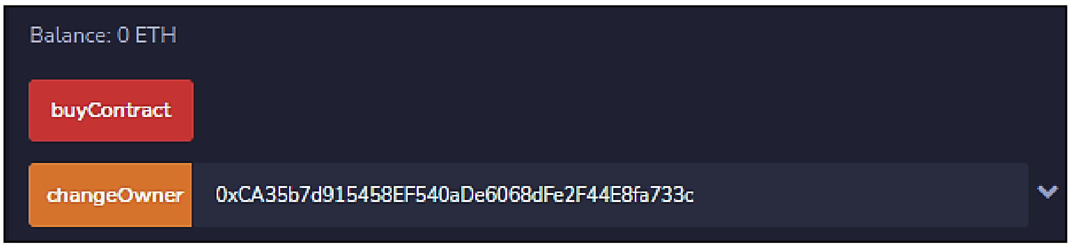
ACCOUNT: 0x5B3...eddC4 (99.99999999999866541 ether)

Step 4-> Copy the address



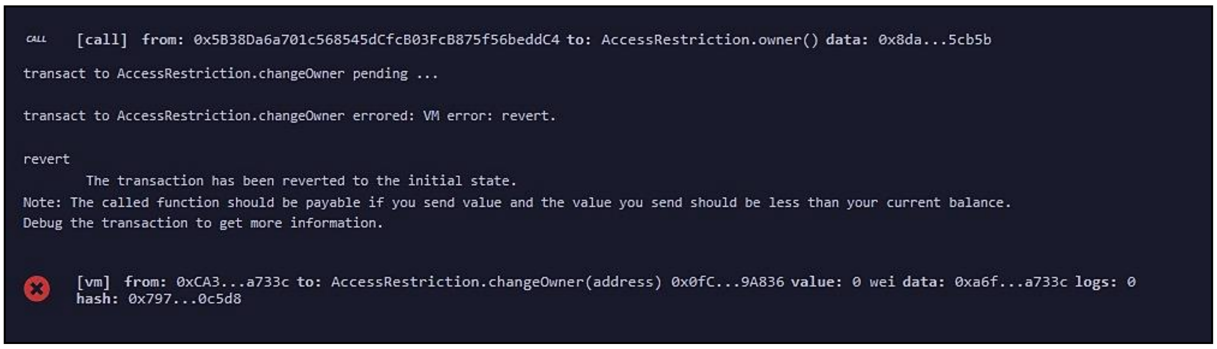
```
ENVIRONMENT
Remix VM (Merge)
VM
ACCOUNT
0xCA3...a733c (100 ether)
GAS LIMIT
3000000
```

Step 5-> Paste the address in changeOwner input and click on changeOwner.



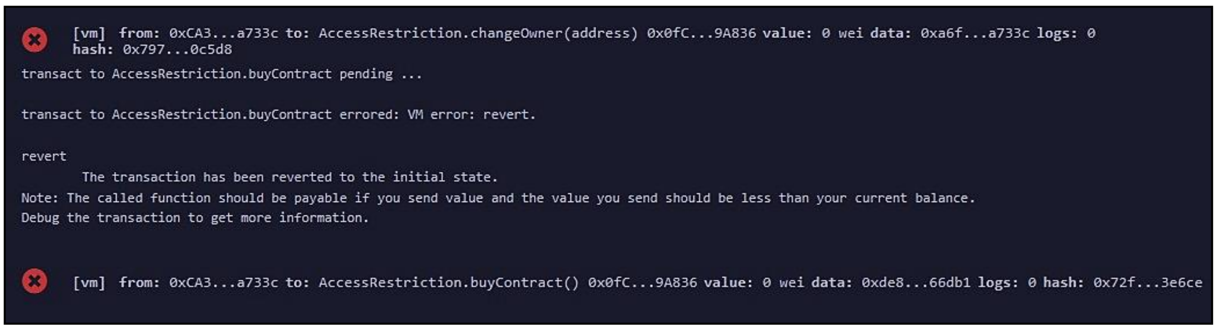
```
Balance: 0 ETH
changeOwner  0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c
```

Step 6-> You should get an error as following



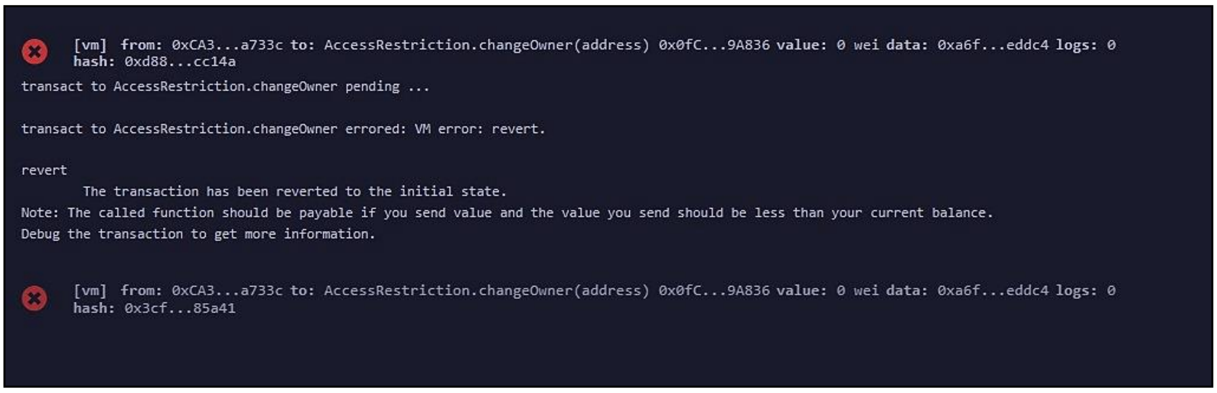
```
CALL   [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: AccessRestriction.owner() data: 0x8da...5cb5b
transact to AccessRestriction.changeOwner pending ...

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

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

       [vm] from: 0xCA3...a733c to: AccessRestriction.changeOwner(address) 0x0fC...9A836 value: 0 wei data: 0xa6f...a733c logs: 0
       hash: 0x797...0c5d8
```

Step 7-> If you click on buycontract it should give an error as follows



```
       [vm] from: 0xCA3...a733c to: AccessRestriction.changeOwner(address) 0x0fC...9A836 value: 0 wei data: 0xa6f...a733c logs: 0
       hash: 0x797...0c5d8
transact to AccessRestriction.buyContract pending ...

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

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

       [vm] from: 0xCA3...a733c to: AccessRestriction.buyContract() 0x0fC...9A836 value: 0 wei data: 0xde8...66db1 logs: 0 hash: 0x72f...3e6ce
```

Step 8-> Now, paste the actual address of the account in the changeowner input and click on changeowner



```
[vm] from: 0xCA3...a733c to: AccessRestriction.changeOwner(address) 0x0fC...9A836 value: 0 wei data: 0xa6f...eddc4 logs: 0
hash: 0xd88...cc14a
transact to AccessRestriction.changeOwner pending ...

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

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


[vm] from: 0xCA3...a733c to: AccessRestriction.changeOwner(address) 0x0fC...9A836 value: 0 wei data: 0xa6f...eddc4 logs: 0
hash: 0x3cf...85a41
```

Practical - 6

Aim: Implement and demonstrate the use of the following in Solidity.
Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.

Theory:

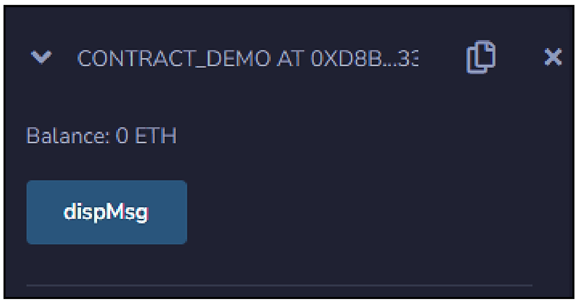
- **Contracts:** Building blocks of smart contracts that contain variables, functions, and data structures.
- **Inheritance:** Mechanism allowing contracts to inherit properties and functionalities from other contracts, promoting code reuse and modularity.
- **Constructors:** Special functions used to initialize contract instances when they are created.
- **Abstract Contracts:** Contracts that cannot be instantiated directly and may contain unimplemented functions, serving as blueprints for derived contracts to implement.
- **Interfaces:** Abstract contracts containing function declarations without implementations, specifying a set of functions that must be implemented by contracts adhering to the interface.

1. Contracts

Code:

```
pragma solidity ^0.5.0;

contract Contract_demo {
string message = "Hello";

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

Output:

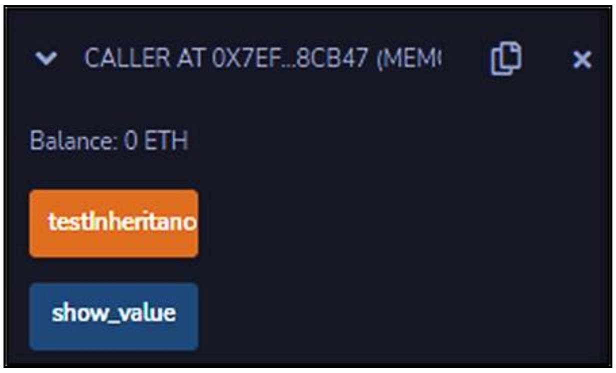


2.Inheritence

Code:

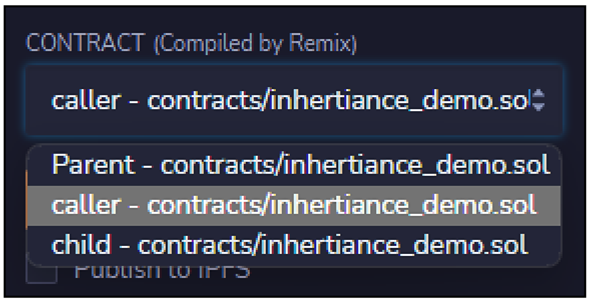
```
pragma solidity >=0.4.22 <0.6.0;

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

Output:



```
CALLER AT 0X7EF...8CB47 (MEM
Balance: 0 ETH
```

Flow of execution
Step1: Select caller contract to deploy in Contract and deploy



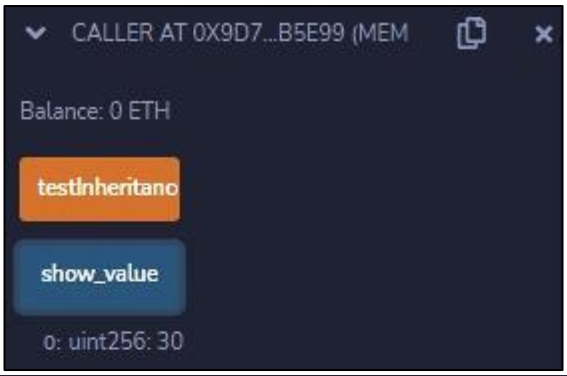
```
CONTRACT (Compiled by Remix)
caller - contracts/inhertiance_demo.sol
Parent - contracts/inhertiance_demo.sol
caller - contracts/inhertiance_demo.sol
child - contracts/inhertiance_demo.sol
Publish to IPFS
```

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

3.Constructor

Code:

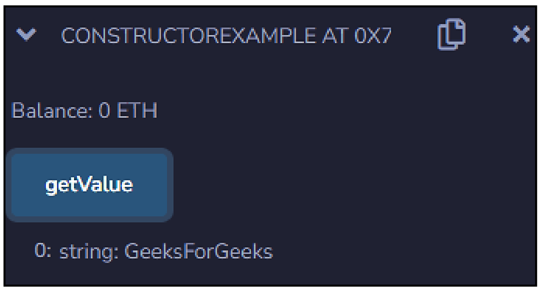
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.0;

// Creating a contract
contract constructorExample { string str;
```



```
0: uint256: 30
```

```

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

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

Output:



```
0: string: GeeksForGeeks
```

4. Abstract Constructor

Code:

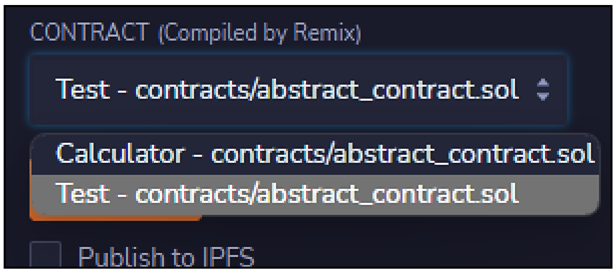
```
// SPDX-License-Identifier: MIT

 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:

Flow of execution

Step1: Select Test contract and deploy



CONTRACT (Compiled by Remix)

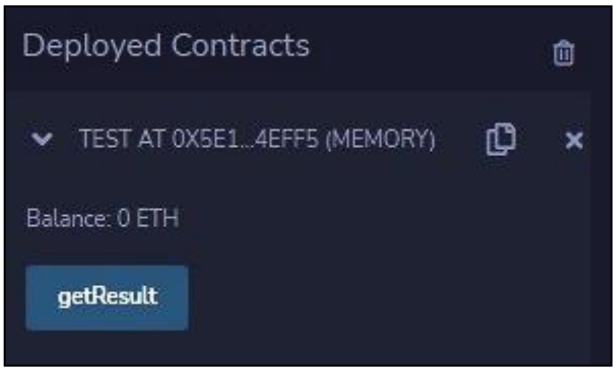
Test - contracts/abstract_contract.sol

Calculator - contracts/abstract_contract.sol

Test - contracts/abstract_contract.sol

Publish to IPFS

Step2: The contact will deploy as below



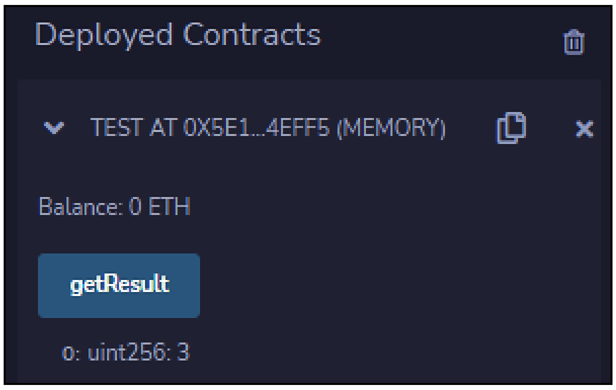
Deployed Contracts

TEST AT 0X5E1...4EFF5 (MEMORY)

Balance: 0 ETH

getResult

Step 3: Click on getResult to get sum of a+b



Deployed Contracts

TEST AT 0X5E1...4EFF5 (MEMORY)

Balance: 0 ETH

getResult

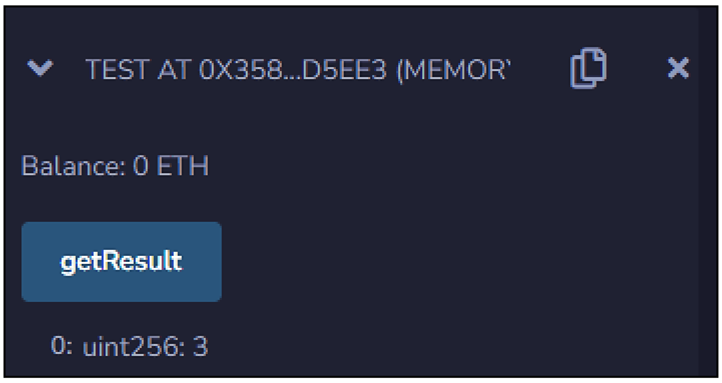
0: uint256: 3

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:



```
TEST AT 0X358...D5EE3 (MEMOR'
Balance: 0 ETH
getResult
0: uint256: 3
```

Practical - 7

Aim: Implement and demonstrate the use of the following in Solidity.
Libraries, Assembly, Events, Error handling.

Theory:

- **Libraries:** Reusable code containers that can be deployed independently and used by multiple contracts to provide shared functionalities.
- **Assembly:** Low-level programming language for writing code directly in machine language instructions, used in Solidity to optimize gas usage or perform advanced operations.
- **Events:** Mechanism for emitting and logging events within a contract, allowing external entities to listen and react to specific occurrences on the blockchain.
- **Error Handling:** Techniques used to handle and manage errors or exceptional conditions that may occur during contract execution, such as throwing and catching exceptions, reverting transactions, or returning error codes.

1.Libraries.

Code:

File : mylib.sol

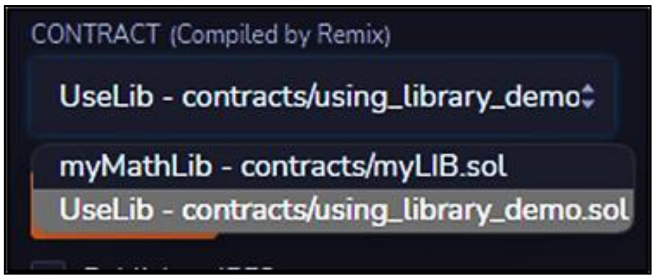
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

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;
}
}
```

File : use_mylib.sol

```
// SPDX-License-Identifier: MIT
 pragma solidity >=0.7.0 <0.9.0;
import "4c/4c_1_libraries_mylib.sol";
 contract UseLib {
function getsum(uint256 x, uint256 y) public pure returns (uint256) {
return myMathLib.sum(x, y);
}

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

Output:

Step 1 : Change contract to UseLib and deploy.



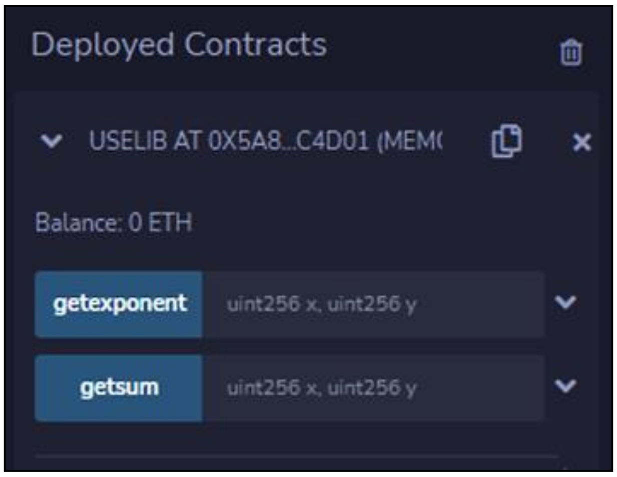
CONTRACT (Compiled by Remix)

UseLib - contracts/using_library_demo

myMathLib - contracts/myLIB.sol

UseLib - contracts/using_library_demo.sol

Step 2: The deployed contract should be same as below


Deployed Contracts

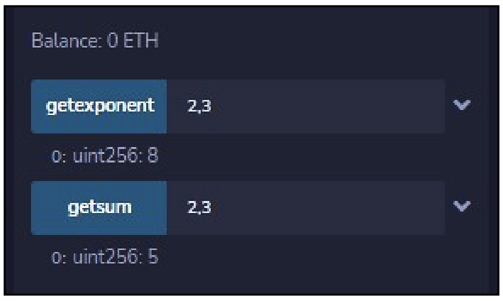
USELIB AT 0X5A8...C4D01 (MEM(

Balance: 0 ETH

Step 3: Input values to both getexponent and getsum functions as below

Step 4: Execute both functions. You will get below

Output :


Balance: 0 ETH

```
0: uint256: 8
```

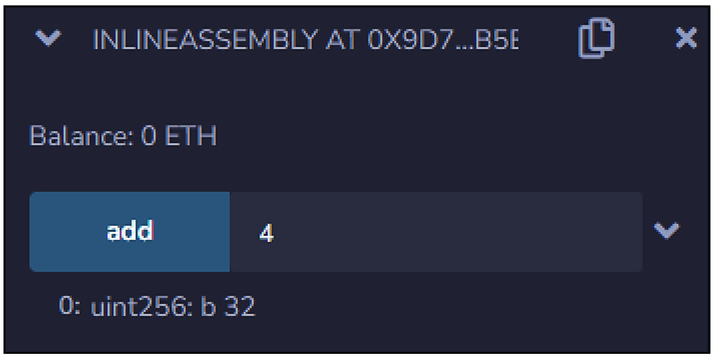
```
0: uint256: 5
```

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:

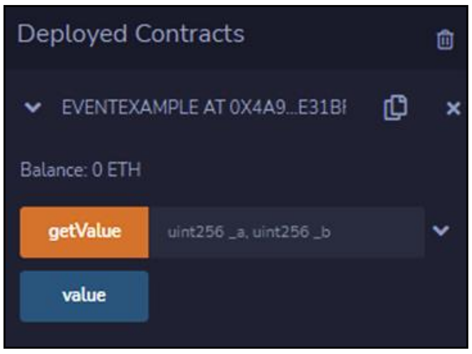

```
INLINEASSEMBLY AT 0X9D7...B5E

Balance: 0 ETH

add   4

0: uint256: b 32
```

3.Events.

Code:

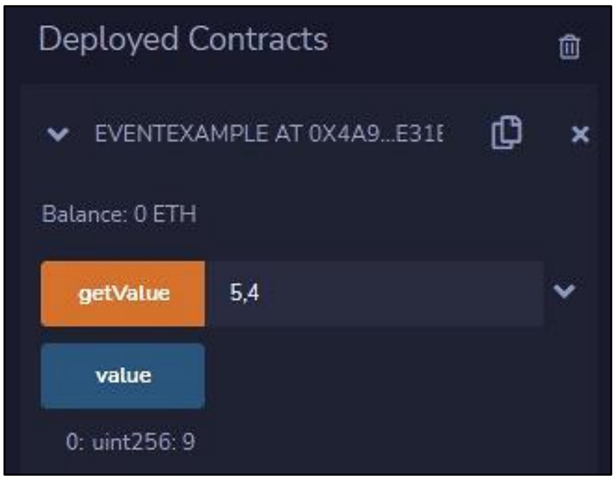
```
// SPDX-License-Identifier: MIT
 pragma solidity ^0.5.0;

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

Output:

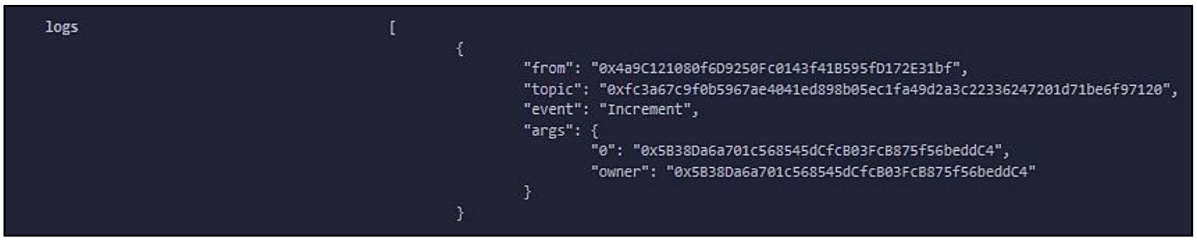

```
Deployed Contracts

EVENTEXAMPLE AT 0X4A9...E31BE

Balance: 0 ETH

getValue   uint256 _a, uint256 _b
value
```

Flow of execution

Step 1: Provide values to getValue function and click on it.



Sep 2: In the terminal check for logs



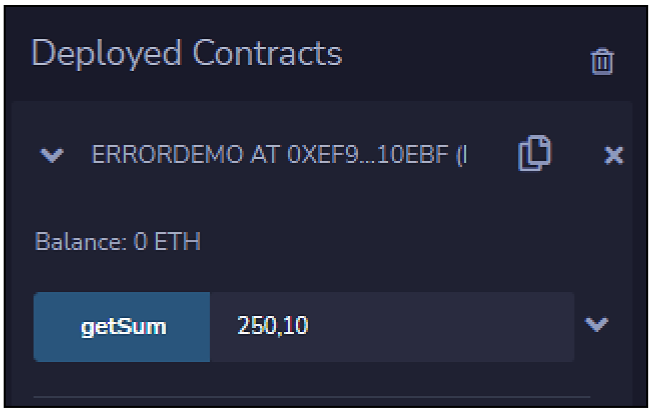
```
logs                                    [
                                                {
                                                        "from": "0x4a9C121080f6D9250Fc0143f41B595fD172E31bf",
                                                        "topic": "0xfc3a67c9f0b5967ae4041ed898b05ec1fa49d2a3c22336247201d71be6f97120",
                                                        "event": "Increment",
                                                        "args": {
                                                                "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                                "owner": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                                        }
                                                }
```

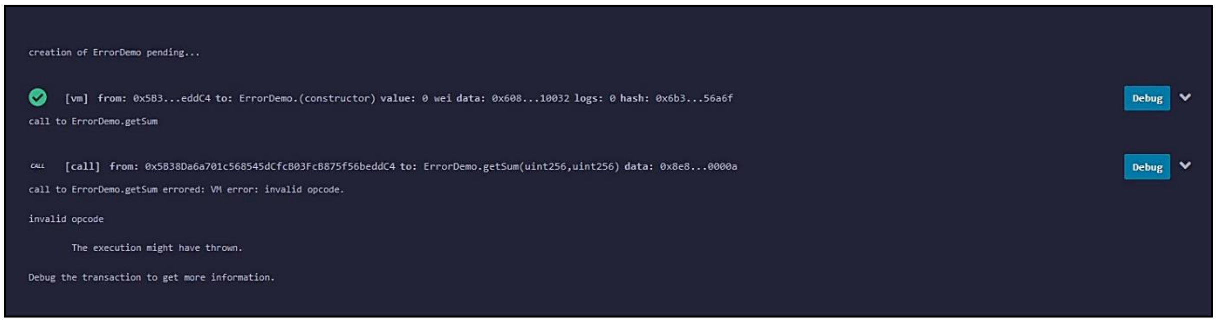
4.Error handling

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:





```
creation of ErrorDemo pending...

[vm] from: 0x5B3...eddC4 to: ErrorDemo.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x6b3...56a6f
call to ErrorDemo.getSum

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

invalid opcode

        The execution might have thrown.

Debug the transaction to get more information.
```

Practical - 8

Aim: Write a program to demonstrate mining of Ether.

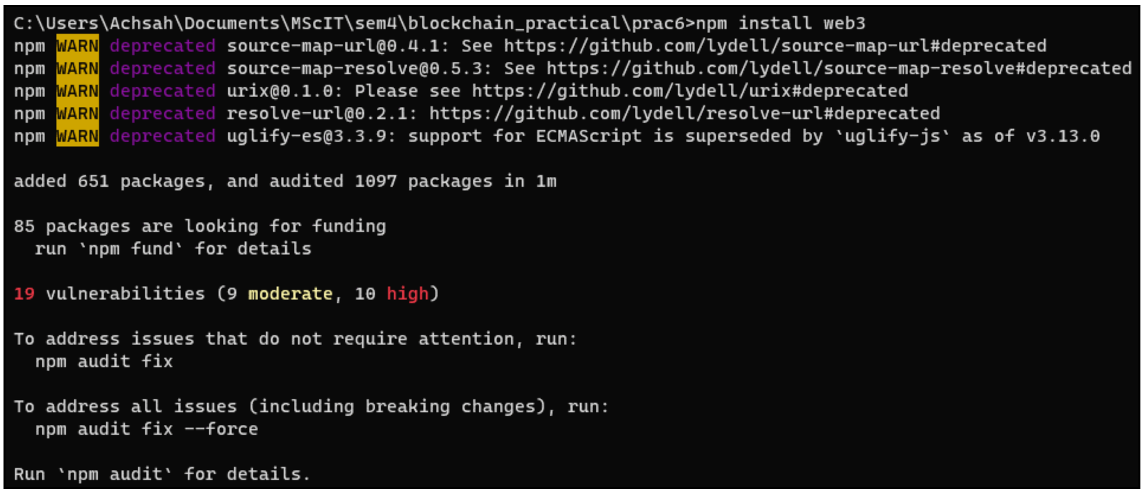
Theory:

Ether mining involves validating and adding new blocks to the Ethereum blockchain using specialized hardware. Miners solve complex mathematical puzzles to find a nonce that satisfies the PoW algorithm. Validated blocks include transactions and are rewarded with newly created Ether. The process requires significant computational power and energy consumption. Ethereum is transitioning to a PoS consensus algorithm, Ethereum 2.0, which will replace mining with staking.

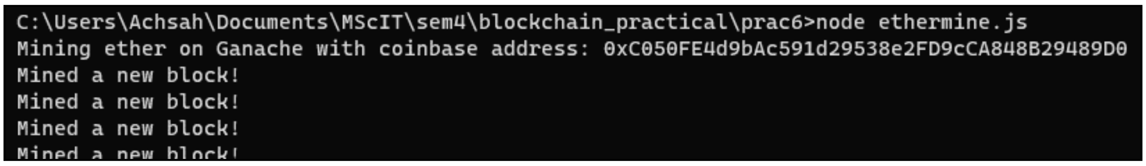
Code:

```
const Web3 = require('web3');
const web3 = new Web3(new
Web3.providers.HttpProvider('http: 127.0.0.1:7545'));
//Replace with your Ganache HTTP provider

async function mine() {
        const accounts = await web3.eth.getAccounts();
        const coinbaseacc1 = accounts[0];
        const coinbaseacc2 = accounts[1];
        console.log(`Mining ether on Ganache with coinbase address:
${coinbaseacc1}`);

while (true) {
         try {
                await web3.eth.sendTransaction({ f
                        rom: coinbaseacc1,
                        to: coinbaseacc2,
                        value: 50,
                });
                console.log(`Mined a new block!`);
        }
catch (err) {
                console.error(err);
        }
        }
}
mine();
```

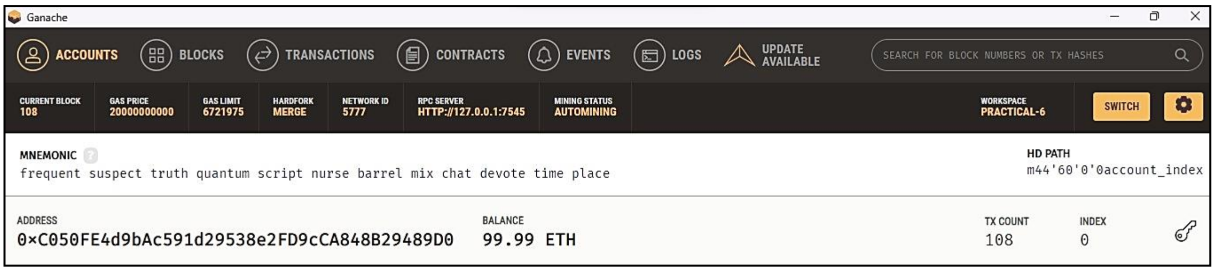
Output:



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac6>npm install web3
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated uglify-es@3.3.9: support for ECMAScript is superseded by `uglify-js` as of v3.13.0

added 651 packages, and audited 1097 packages in 1m

85 packages are looking for funding
  run `npm fund` for details

19 vulnerabilities (9 moderate, 10 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac6>node ethermine.js
Mining ether on Ganache with coinbase address: 0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
```



CURRENT BLOCK 108 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE PRACTICAL-6

MNEMONIC
frequent suspect truth quantum script nurse barrel mix chat devote time place

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×C050FE4d9bAc591d29538e2FD9cCA848B29489D0	99.99 ETH	108	0

Practical - 9

Aim: Demonstrate the running of the blockchain node.

Theory:

A blockchain node is a participant in the network that maintains a copy of the blockchain ledger and participates in the consensus process. It stores and updates the ledger, validates transactions, and communicates with other nodes to maintain synchronization.

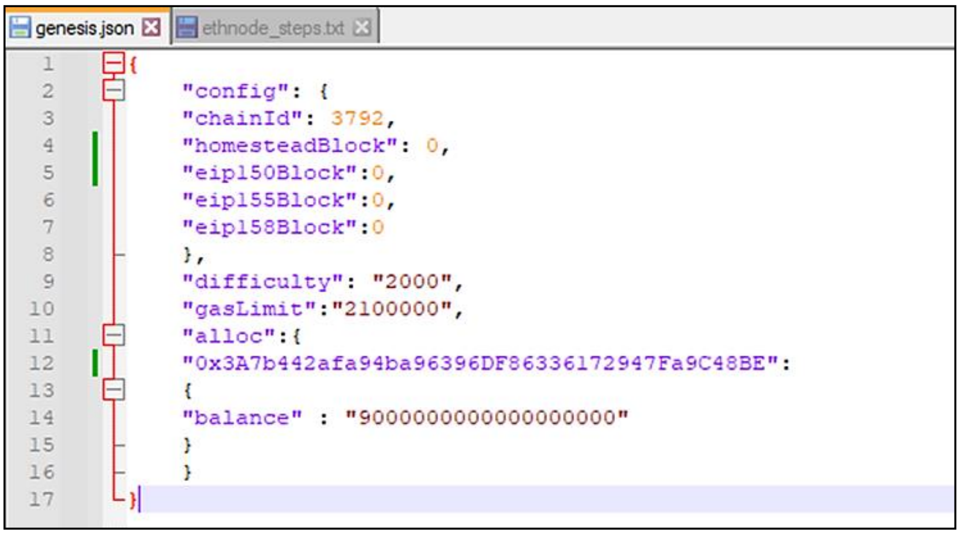
Nodes contribute to the security and integrity of the blockchain network by participating in consensus algorithms. They play a crucial role in processing transactions and ensuring the network's availability and redundancy.

A blockchain node refers to a participant or entity in a blockchain network that maintains a copy of the entire blockchain ledger and actively participates in the consensus mechanism of the network. Each node stores a complete record of all transactions that have occurred on the blockchain and has the ability to validate and verify new transactions.

Code:

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

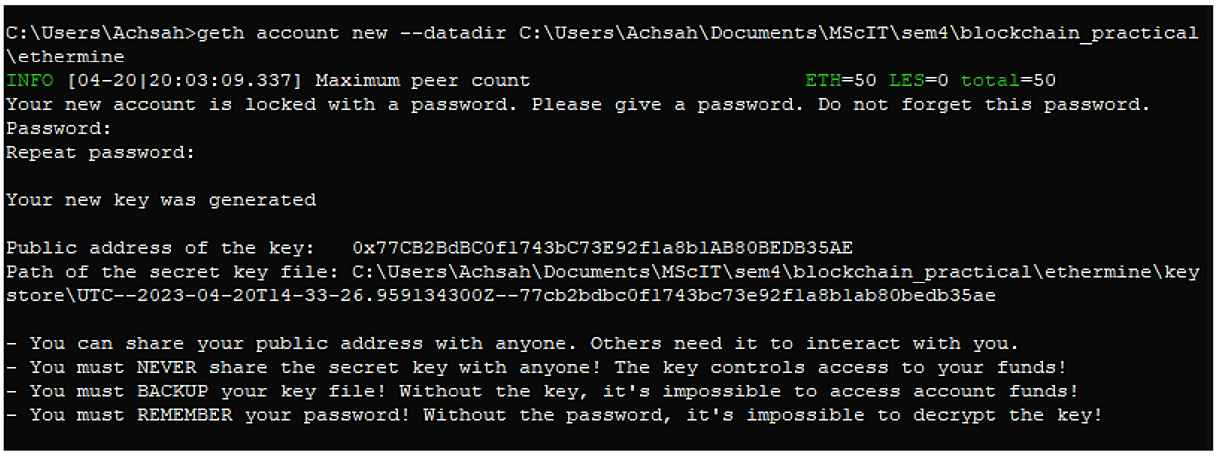
```
{
"config": {
"chainId": 3792,
"homesteadBlock": 0,
"eip150Block": 0,
"eip155Block": 0,
"eip158Block": 0
},
"difficulty": "2000",
"gasLimit": "2100000", "alloc": {
"0×0b6C4c81f58B8d692A7B46AD1e16a1147c25299F": { "balance":
"9000000000000000000"
}
}
}
```



```
{
    "config": {
    "chainId": 3792,
    "homesteadBlock": 0,
    "eip150Block":0,
    "eip155Block":0,
    "eip158Block":0
    },
    "difficulty": "2000",
    "gasLimit":"2100000",
    "alloc":{
    "0x3A7b442afa94ba96396DF86336172947Fa9C48BE":
    {
    "balance" : "9000000000000000000"
    }
    }
}
```

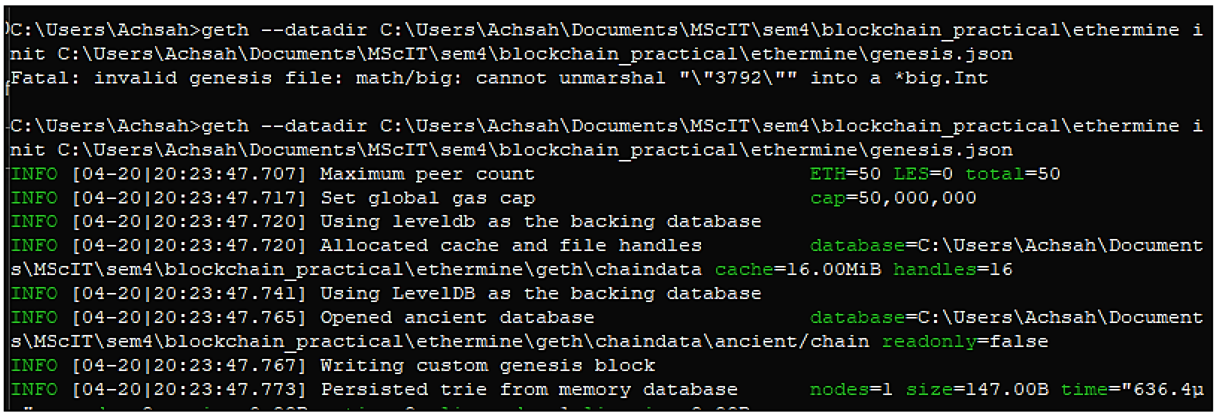
Step 2: Run command geth account new –datadir

C:\Users \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

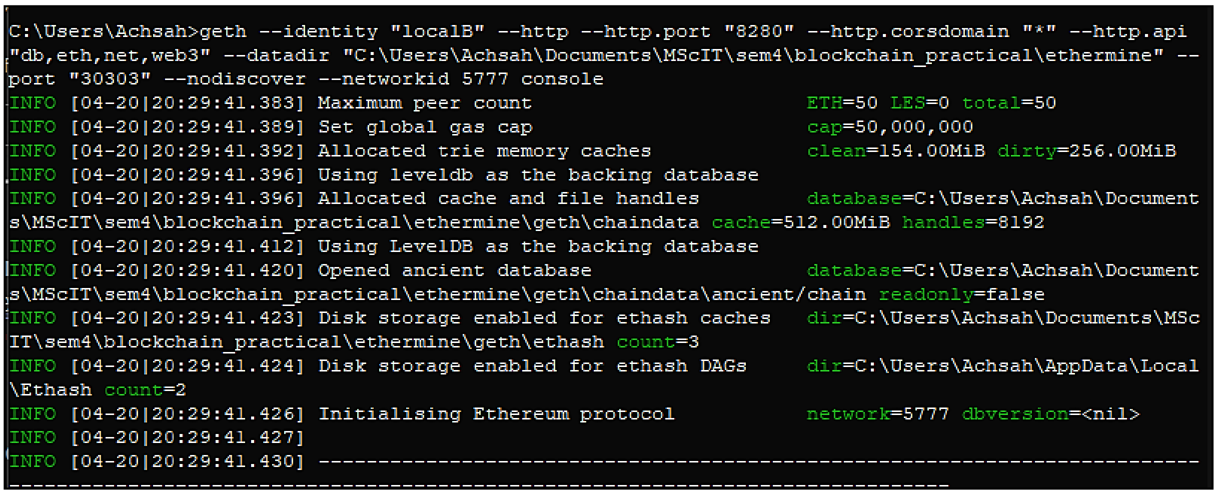


```
C:\Users\Achsah>geth --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine i
nit C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\genesis.json
Fatal: invalid genesis file: math/big: cannot unmarshal "\"3792\"" into a *big.Int

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"

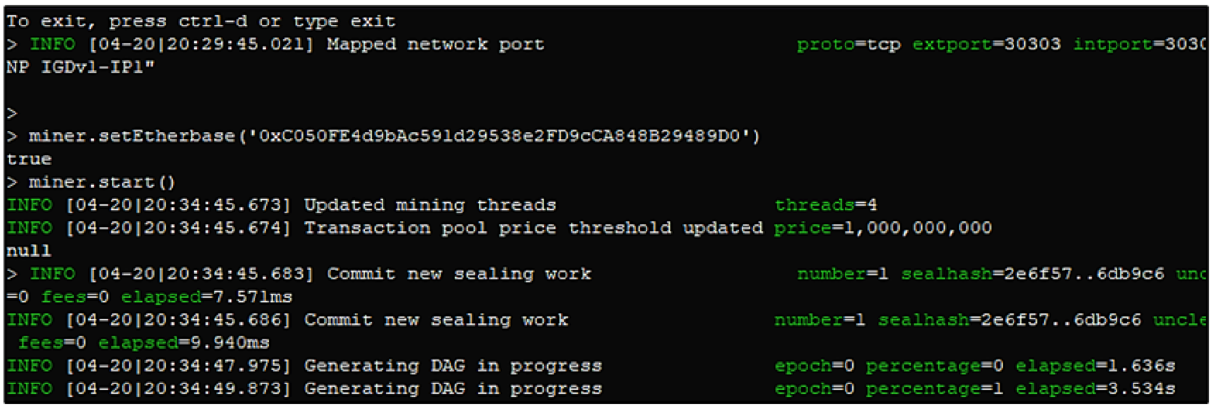
--port "30303" --nodiscover --networkid 5777 console. This command will enable geth console.



```
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] -----------------------------------------------------------------------------
------------------------------------------------------------------------------
```

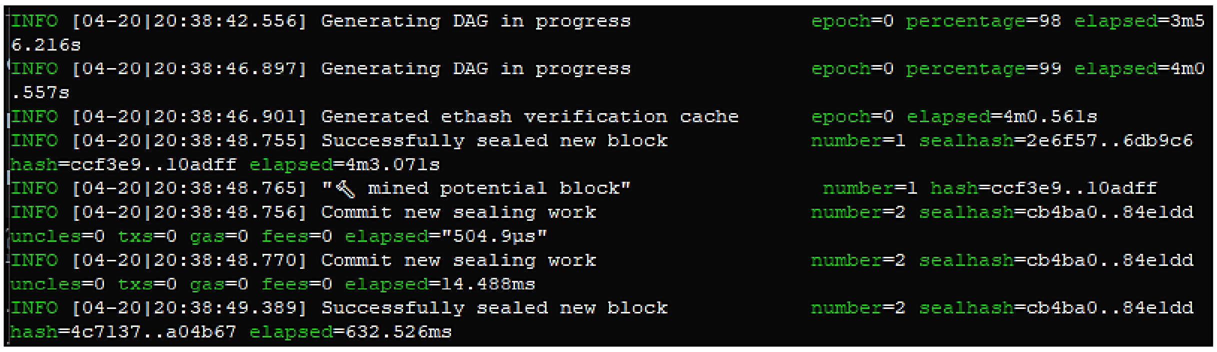
Step 5: Run the command miner.setEtherbase('0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0') in the geth console

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



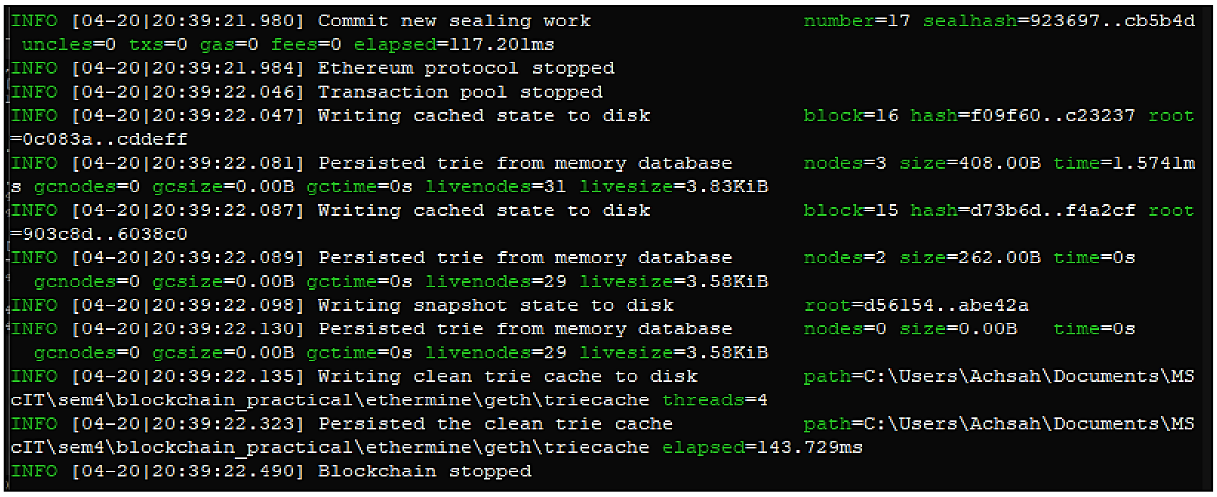
```
To exit, press ctrl-d or type exit
> INFO [04-20|20:29:45.021] Mapped network port                      proto=tcp extport=30303 intport=303(
NP IGDv1-IP1"

>
> 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..84e1dd
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..84e1dd
uncles=0 txs=0 gas=0 fees=0 elapsed=14.488ms
INFO [04-20|20:38:49.389] Successfully sealed new block           number=2 sealhash=cb4ba0..84e1dd
hash=4c7137..a04b67 elapsed=632.526ms
```

Step 8: To stop the mining press Ctrl+D



```
INFO [04-20|20:39:21.980] Commit new sealing work                 number=17 sealhash=923697..cb5b4d
 uncles=0 txs=0 gas=0 fees=0 elapsed=117.201ms
INFO [04-20|20:39:21.984] Ethereum protocol stopped
INFO [04-20|20:39:22.046] Transaction pool stopped
INFO [04-20|20:39:22.047] Writing cached state to disk            block=16 hash=f09f60..c23237 root
=0c083a..cddeff
INFO [04-20|20:39:22.081] Persisted trie from memory database     nodes=3 size=408.00B time=1.5741m
s gcnodes=0 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 - 10

Aim: Create your own blockchain and demonstrate its use.

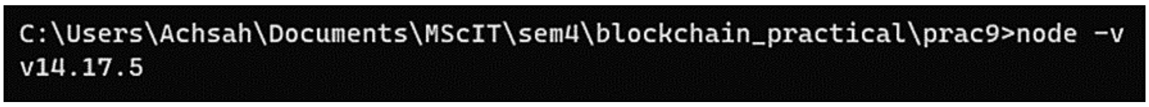
Code:

```
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, "21/04/2023", "Genesis Block", "0");

}

getLatestBlock() {

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

}

addBlock(newBlock) {

newBlock.previousHash = this.getLatestBlock().hash;
newBlock.hash = newBlock.calculateHash(); this.chain.push(newBlock);
}
isChainValid() {
for (let i = 1; i < this.chain.length; i +) { const currentBlock = this.chain[i];
const previousBlock = this.chain[i - 1];
if (currentBlock.hash  currentBlock.calculateHash()) { return false;
}
if (currentBlock.previousHash           previousBlock.hash) { return false;
}
}
return true;
}
}
 Blockchain Implementation
let myCoin = new Blockchain();
myCoin.addBlock(new Block(1, "22/04/2023", { amount: 4 }));
 myCoin.addBlock(new Block(2, "22/04/2023", { amount: 8 }));
 console.log('Is blockchain valid? ' + myCoin.isChainValid());
console.log(JSON.stringify(myCoin, null, 4));
```

Output:

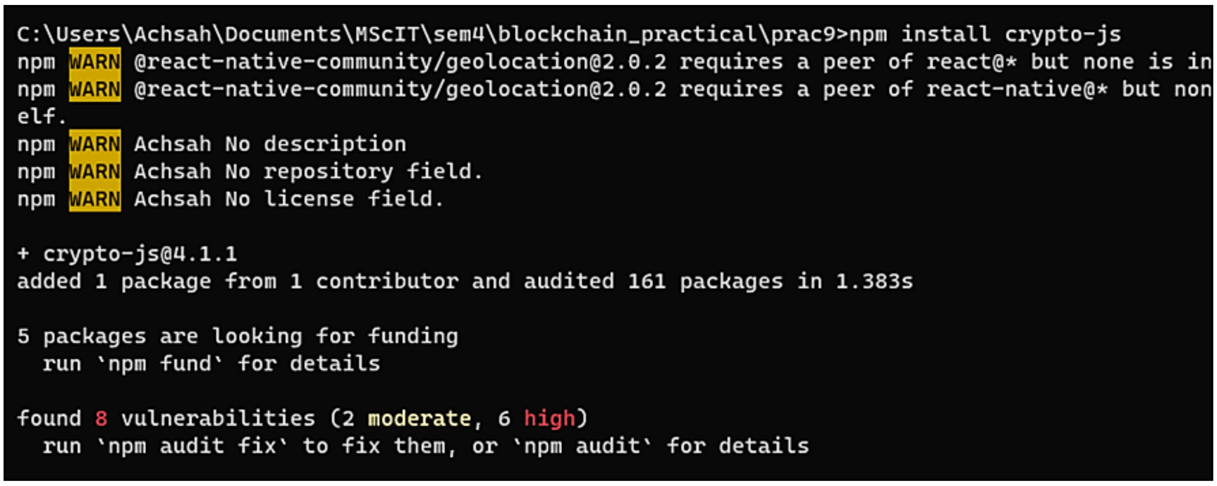
Flow of execution

Step 1: Make sure you have installed nodejs in your system



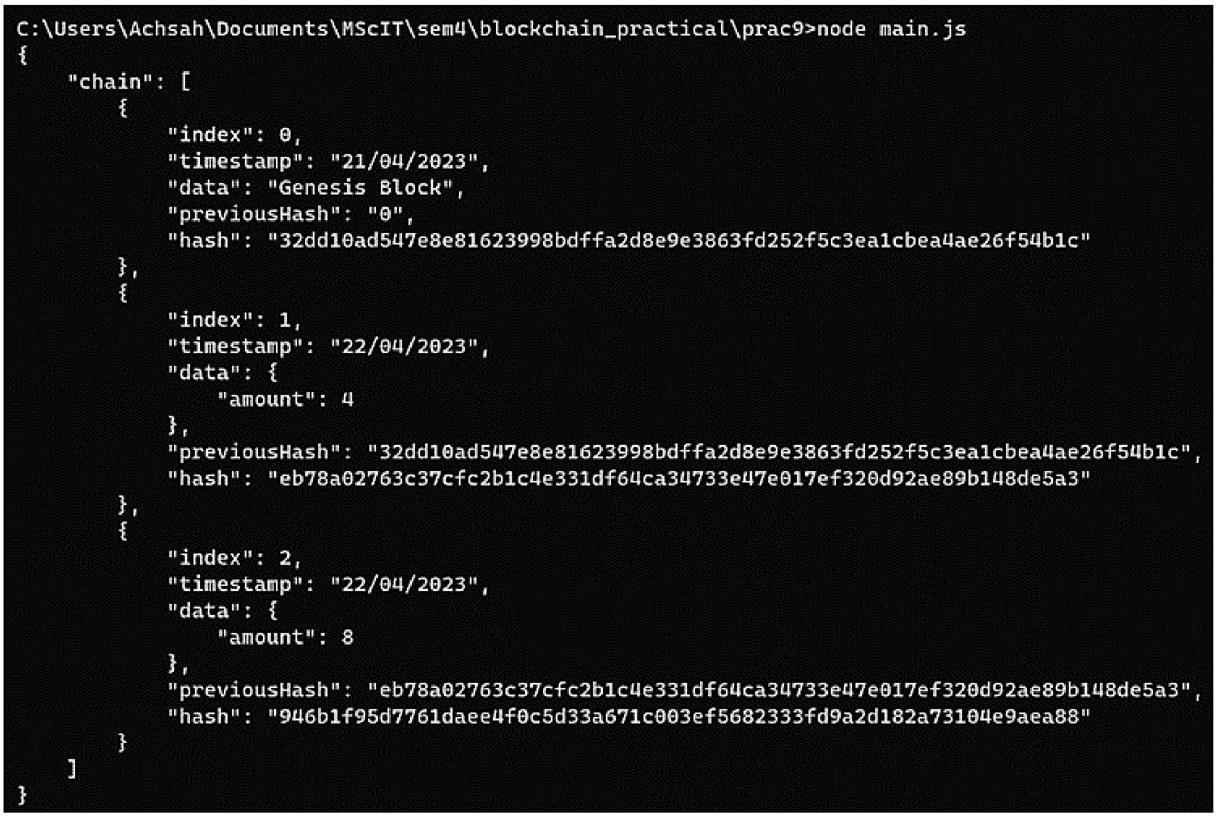
```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac9>node -v
v14.17.5
```

Step 2: We need crypto –js node module to make our own blockchain. So install it as following



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac9>npm install crypto-js
npm WARN @react-native-community/geolocation@2.0.2 requires a peer of react@* but none is in
npm WARN @react-native-community/geolocation@2.0.2 requires a peer of react-native@* but non
elf.
npm WARN Achsah No description
npm WARN Achsah No repository field.
npm WARN Achsah No license field.

+ crypto-js@4.1.1
added 1 package from 1 contributor and audited 161 packages in 1.383s

5 packages are looking for funding
  run `npm fund` for details

found 8 vulnerabilities (2 moderate, 6 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

Step 3: Run the above code in command line using command: node main.js



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac9>node main.js
{
    "chain": [
        {
            "index": 0,
            "timestamp": "21/04/2023",
            "data": "Genesis Block",
            "previousHash": "0",
            "hash": "32dd10ad547e8e81623998bdffa2d8e9e3863fd252f5c3ea1cbea4ae26f54b1c"
        },
        {
            "index": 1,
            "timestamp": "22/04/2023",
            "data": {
                "amount": 4
            },
            "previousHash": "32dd10ad547e8e81623998bdffa2d8e9e3863fd252f5c3ea1cbea4ae26f54b1c",
            "hash": "eb78a02763c37cfc2b1c4e331df64ca34733e47e017ef320d92ae89b148de5a3"
        },
        {
            "index": 2,
            "timestamp": "22/04/2023",
            "data": {
                "amount": 8
            },
            "previousHash": "eb78a02763c37cfc2b1c4e331df64ca34733e47e017ef320d92ae89b148de5a3",
            "hash": "946b1f95d7761daee4f0c5d33a671c003ef5682333fd9a2d182a73104e9aea88"
        }
    ]
}
```