

**University of Mumbai**  
**PRACTICAL JOURNAL – PAPER I**



**PSIT4P1**  
**Blockchain**

**SUBMITTED BY**

**NEGI VIVEKSINGH DHIRAJ SINGH**

**SEAT NO: 40379**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR QUALIFYING M.Sc. (I.T.) PART-II (SEMESTER – IV)  
EXAMINATION  
2023-2024**

**DEPARTMENT OF INFORMATION  
TECHNOLOGY 3<sup>RD</sup> FLOOR, DR. SHANKAR DAYAL  
SHARMA BHAVAN, VIDYANAGRI, SANTACRUZ (E),  
MUMBAI – 400098.**

# University of Mumbai



## Department of Information Technology

### Certificate

This is to certify that Mr. VIVEKSINGH NEGI, Seat No. 40379 studying in **Master of Science in Information Technology Part II Semester IV** has satisfactorily completed the Practical of **PSIT4P1 Blockchain** as prescribed by the University of Mumbai, during the academic year **2023-24**.

---

Signature

Subject-In-Charge

---

Signature

Head of the Department

---

Signature

External Examiner

College Seal: \_\_\_\_\_

Date: \_\_\_\_\_

## INDEX

Practical No.	Name of Practical	Page No.	Date	Signature
1	Write the following programs for Blockchain in Python: A] A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and testing it. B] A transaction class to send and receive money and test it. C] Create multiple transactions and display them. D] Create a blockchain and a genesis block and execute it. E] Create a mining function and test it. F] Add blocks to the miner and dump the blockchain.	1	17/04/24	
2	Install and configure Go Ethereum and the Mist browser. Develop and test a sample application.	13	07/05/24	
3	Implement and demonstrate the use of the following in Solidity: A] Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. B] Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.	17	17/05/24	
4	Implement and demonstrate the use of the following in Solidity: A] Withdrawal Pattern, Restricted Access. B] Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces. C] Libraries, Assembly, Events, Error handling.	37	24/05/24	
5	Write a program to demonstrate the mining of Ether.	50	04/06/24	
6	Create your own blockchain and demonstrate its use.	53	24/06/24	

**PRACTICAL NO: 1**

**AIM: Write the following programs for Blockchain in Python:**

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

**CODE:**

```
import binascii

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

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

        self._private_key = RSA.generate(1024, random)

        self._public_key = self._private_key.publickey()

        self._signer = PKCS1_v1_5.new(self._private_key)

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

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

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

**OUTPUT:**

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

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

**CODE:**

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

class Client:
    def __init__(self):
        random = Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

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

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

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

```
{
    "sender": identity,
    "recipient": self.recipient,
    "value": self.value,
    "time": self.time,
}

)

def sign_transaction(self):
    private_key = self.sender._private_key
    signer = PKCS1_v1_5.new(private_key)
    h = SHA.new(str(self.to_dict()).encode("utf8"))
    return binascii.hexlify(signer.sign(h)).decode("ascii")
```

UDIT = Client()

UGC = Client()

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

print("\nTransaction Recipient:\n", t.recipient) # print("\nTransaction Sender:\n", t.sender)

print("\nTransaction Value:\n", t.value)

signature = t.sign\_transaction()

print("\nSignature:\n", signature)

### OUTPUT:

#### Transaction Recipient:

```
30819f300d06092a864886f70d010101050003818d0030818902818100be0978593e24a777060c9b
95c383a53f3e3213905fc67538a37f15e9c2c4bd6e0eb667999074079928c3e3d7f0636f0d7e9f8
cc0bd7a38da76342c612c103fe43a603e97c602d1f2a0dbbe794ab983aa1d1062d70485c0e0c734
3c265e90ae4094c6ede9037b0d38af530b27914df1a08c339ec4bc76de28b2dbe5cd6a0e3ebf020
3010001
```

#### Signature:

```
a3d6c2635b55cd891bc43dc63b137bfb10eaa68dd7ce276f8e098367cc83195f5a508fa2e4e2c74a
341e143d38432b36e4dc240f5f4b17d07d5be3ebf67a59f42583c215e9d78059091580629eee052
985b2bf355ab58eae226b41dbfaa9690bfce61745392e9f172173ce5a2f12040b753610468ca4a3
fccb1437968a11fc00
```

[C] Create multiple transactions and display them.

**CODE:**

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

class Client:
    def __init__(self):
        random = Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

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

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

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

```
return collections.OrderedDict(  
    {  
        "sender": identity,  
        "recipient": self.recipient,  
        "value": self.value,  
        "time": self.time,  
    }  
)
```

```
def sign_transaction(self):  
    private_key = self.sender._private_key  
    signer = PKCS1_v1_5.new(private_key)  
    h = SHA.new(str(self.to_dict()).encode("utf8"))  
    return binascii.hexlify(signer.sign(h)).decode("ascii")
```

```
def display_transaction(transaction):  
    # for transaction in transactions:  
    dict = transaction.to_dict()  
    print("sender: " + dict['sender'])  
    print('-----')  
    print("recipient: " + dict['recipient'])  
    print('-----')  
    print("value: " + str(dict['value']))  
    print('-----')  
    print("time: " + str(dict['time']))  
    print('-----')
```

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



```

t1 = Transaction(UDIT, UGC.identity, 15.0)

t1.sign_transaction()

transactions = [t1]

t2 = Transaction(UDIT, AICTE.identity, 6.0)

t2.sign_transaction()

transactions.append(t2)

t3 = Transaction(UGC, MU.identity, 2.0)

t3.sign_transaction()

transactions.append(t3)

t4 = Transaction(AICTE, UGC.identity, 4.0)

t4.sign_transaction()

transactions.append(t4)

for transaction in transactions:

    Transaction.display_transaction(transaction)

    print("      ")

```

**OUTPUT:**

```

sender:
30819f300d06092a864886f70d010101050003818d00308189028181009c383a3fc7
248a85bb871be70bdad857db28da890de9de76f3df85950fa0452ab1615bf6b7967f
28c642e433a04335befdb4232c0735d271a0949a28e51e9154f327c7edd6be6e5588
73c12afb6d66a48c6fc726515789d1109438f75446829b3832be14e894a40f27e1331
cc48e536a30cc47a1039b4d87364a3ac2c3f7553110203010001
-----
recipient:
30819f300d06092a864886f70d010101050003818d0030818902818100a597e8496e
f09e903eac0b9f97d8bfde76565f58599206bbd47fb020b0d0daef0568449ef7cb44
68abe58a30e7877276404a5264840f4e0ddda570cbefec408c459d58429573427694
382caa972f3878f7ae04ff27bbea057bf9360dd65e193eaf8301c5168840e6a55812
867b746de1da1695c5130d49cbee519c6cb23698710203010001
-----
value: 15.0
-----
time: 2023-06-25 18:38:53.536480
-----

```

```

sender:
30819f300d06092a864886f70d010101050003818d00308189028181009c383a3fc7
248a85bb871be70bdad857db28da890de9de76f3df85950fa0452ab1615bf6b7967f
28c642e433a04335befdb4232c0735d271a0949a28e51e9154f327c7edd6be6e5588
73c12afb6d66a48c6fc726515789d1109438f75446829b3832be14e894a40f27e1331
cc48e536a30cc47a1039b4d87364a3ac2c3f7553110203010001
-----
recipient:
30819f300d06092a864886f70d010101050003818d0030818902818100cfc0b7e441
5f62ea994a001e1d30f8d23d89ec17062f162c5f2ac56c4c5883ace946a59854562d
116efa15269b50a0d180e8b46db923ec942c347826037007c3b638bd666a4949e773
bc98ba3ff5e0f4fa06e97f0d31986ee2090e02a4ba9dd60951d48e8253763a08730b
0c4e4f3aae04ac568b9c9708406a5c9564d03ace250203010001
-----
value: 6.0
-----
time: 2023-06-25 18:38:53.538479
-----

```

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

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

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

**CODE:**

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

class Client:
    def __init__(self):
        random = Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

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

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

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

```
return collections.OrderedDict(  
    {  
        "sender": identity,  
        "recipient": self.recipient,  
        "value": self.value,  
        "time": self.time,  
    }  
)
```

```
def sign_transaction(self):  
    private_key = self.sender._private_key  
    signer = PKCS1_v1_5.new(private_key)  
    h = SHA.new(str(self.to_dict()).encode("utf8"))  
    return binascii.hexlify(signer.sign(h)).decode("ascii")
```

```
def display_transaction(transaction):  
    # for transaction in transactions:  
    dict = transaction.to_dict()  
    print("sender: " + dict['sender'])  
    print('-----')  
    print("recipient: " + dict['recipient'])  
    print('-----')  
    print("value: " + str(dict['value']))  
    print('-----')  
    print("time: " + str(dict['time']))  
    print('-----')
```

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

```
self.Nonce = ""  
self.client = client
```

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

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

**OUTPUT:**

```
Number of blocks in the chain: 1  
block # 0  
sender: Genesis  
-----  
recipient:  
30819f300d06092a864886f70d0101050003818d0030818902818100eca6bb5563  
9066584a301764d24f95860798f3cc060ee433ea8aa72458506e8c279812b5415f5d  
a25d966065b3eadd12dfa6bd819411253bf9883a11947b7a093a52c47afe7fff165a  
454859d297d02a9d0baadff621af0cee fb302183762305aa53c66c6681940f8fb9c3  
05dfdd132cfb49e6d24ee01578043f591e6b28b7910203010001  
-----  
value: 500.0  
-----  
time: 2023-06-25 18:44:33.262135  
-----
```

[E] Create a mining function and test it.

**CODE:**

```
import hashlib

def sha256(message):
    return hashlib.sha256(message.encode("ascii")).hexdigest()

def mine(message, difficulty=1):
    assert difficulty >= 1
    prefix = "1" * difficulty
    for i in range(1000):
        digest = sha256(str(hash(message)) + str(i))
        if digest.startswith(prefix):
            print(f"After {str(i)} iterations found nonce: {digest}")
            return digest

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

**OUTPUT:**

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

[F] Add blocks to the miner and dump the blockchain.

**CODE:**

```
import datetime

import hashlib

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()

blockchain = [Block("First block", "0")]

blockchain.append(Block("Second block", blockchain[0].hash))

blockchain.append(Block("Third block", blockchain[1].hash))

# Dumping the blockchain

for block in blockchain:

    print(

        f"Timestamp: {block.timestamp}\nData: {block.data}\nPrevious Hash: {block.previous_hash}\nHash: {block.hash}\n"

    )
```

**OUTPUT:**

```
Timestamp: 2023-06-25 13:18:33.950453+00:00
Data: First block
Previous Hash: 0
Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9

Timestamp: 2023-06-25 13:18:33.950453+00:00
Data: Second block
Previous Hash:
876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9
Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd

Timestamp: 2023-06-25 13:18:33.950453+00:00
Data: Third block
Previous Hash:
8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd
Hash: 06e369fbbf5362a8115a5c6f3e2d3ec7292cc4272052dcc3280898e3206208d
```

**PRACTICAL NO: 2**

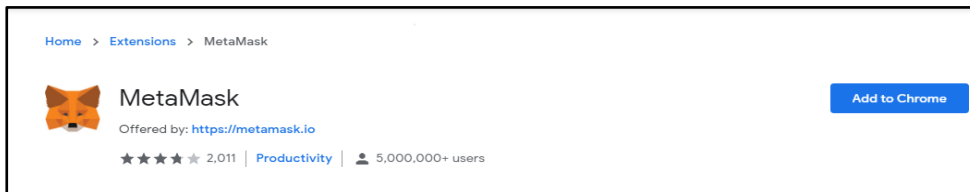
**AIM:** Install and configure Go Ethereum and the Mist browser. Develop and test a sample application.

**Step 1:** Go to [Chrome Web Store Extensions Section](#).

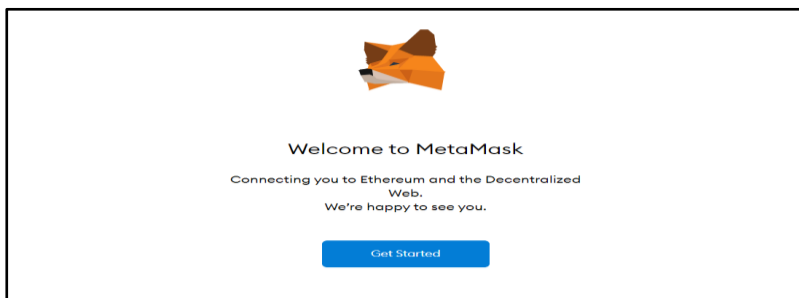
**Step 2:** Search MetaMask.

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

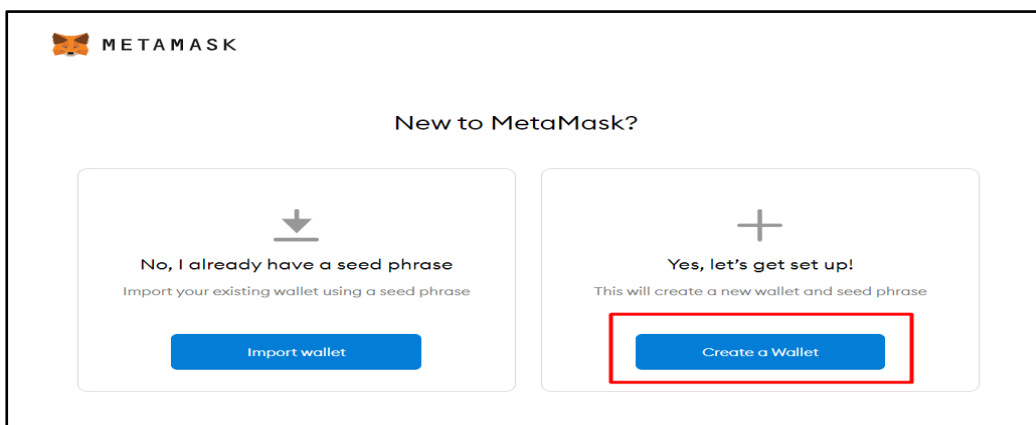
**Step 4:** Click the Add to Chrome button.



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





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



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



 METAMASK  


### Help Us Improve MetaMask

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


MetaMask will..

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click & pageview events

- ✗ **Never** collect keys, addresses, transactions, balances, hashes, or any personal information
- ✗ **Never** collect your full IP address
- ✗ **Never** sell data for profit. Ever!

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

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

 METAMASK  
< Back  

## Create Password

New password (min 8 chars)


Confirm password

☐ I have read and agree to the [Terms of Use](#)

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


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

 METAMASK  

## Secret Backup Phrase

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

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

 [CLICK HERE TO REVEAL SECRET WORDS](#)

**Tips:**

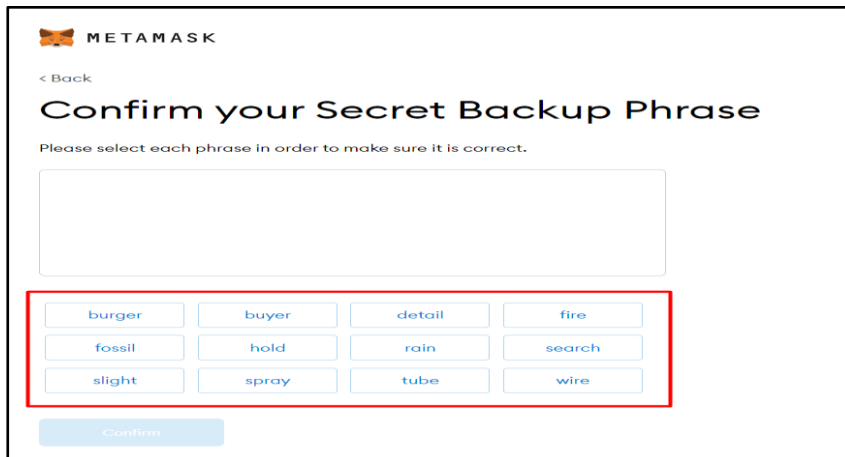
Store this phrase in a password manager like 1Password.

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

Memorize this phrase.

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

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



METAMASK

< Back

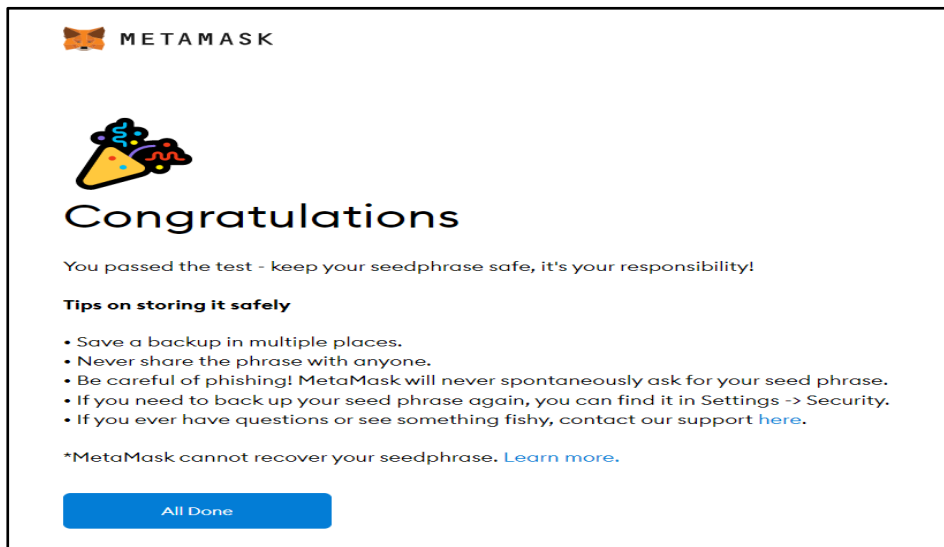
### Confirm your Secret Backup Phrase

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


burger	buyer	detail	fire
fossil	hold	rain	search
slight	spray	tube	wire

Confirm

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



METAMASK



## Congratulations

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

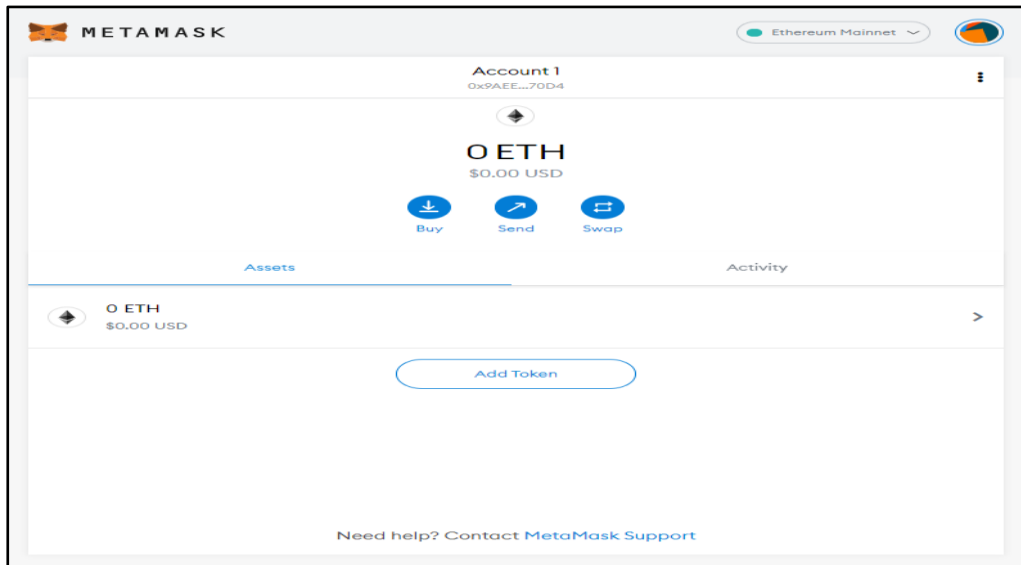
**Tips on storing it safely**

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

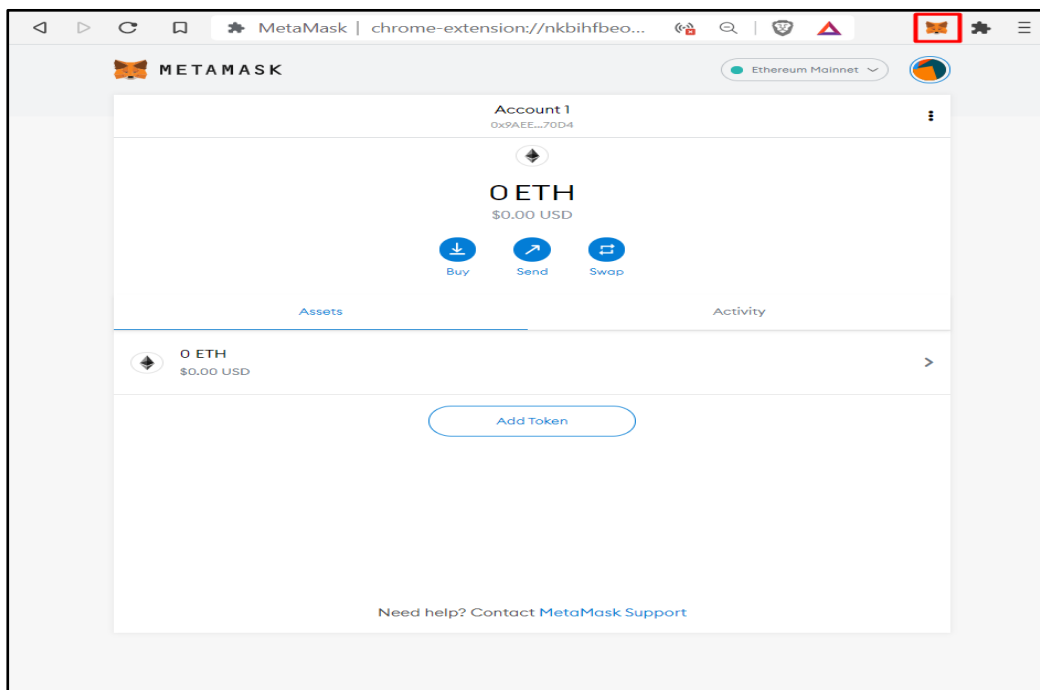
\*MetaMask cannot recover your seedphrase. [Learn more](#).

All Done

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



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



**PRACTICAL NO: 3**

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

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

**[I] Variable**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract variable\_demo {

uint256 sum = 4;

uint256 x;

address a;

string s = "welcome";

function add(uint256) public {

uint256 y = 2;

sum = sum + x + y;

}

function display() public view returns (uint256) {

return sum;

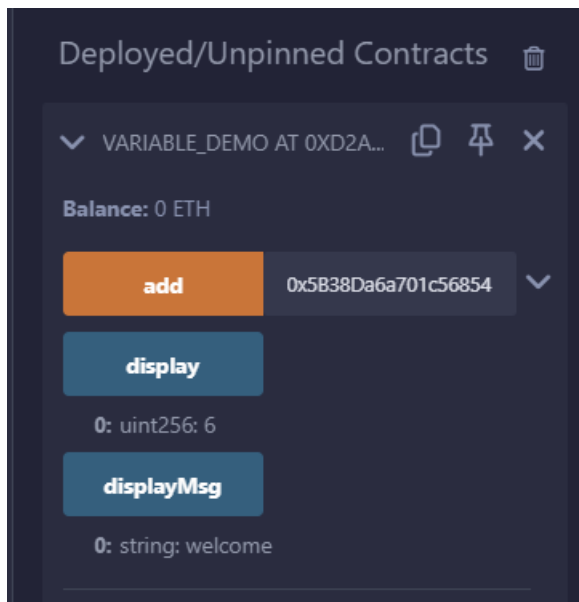
}

function displayMsg() public view returns (string memory){

return s;

}

}

**OUTPUT:-****[II] Operators****CODE:**

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

```
pragma solidity ^0.8.0;
```

```
contract Operators {
```

```
    uint256 result = 0;
```

```
    function addition(uint256 a, uint256 b) public pure returns (uint256) {
```

```
        return a + b;
```

```
    }
```

```
    function subtraction(uint256 a, uint256 b) public pure returns (uint256) {
```

```
        return a - b;
```

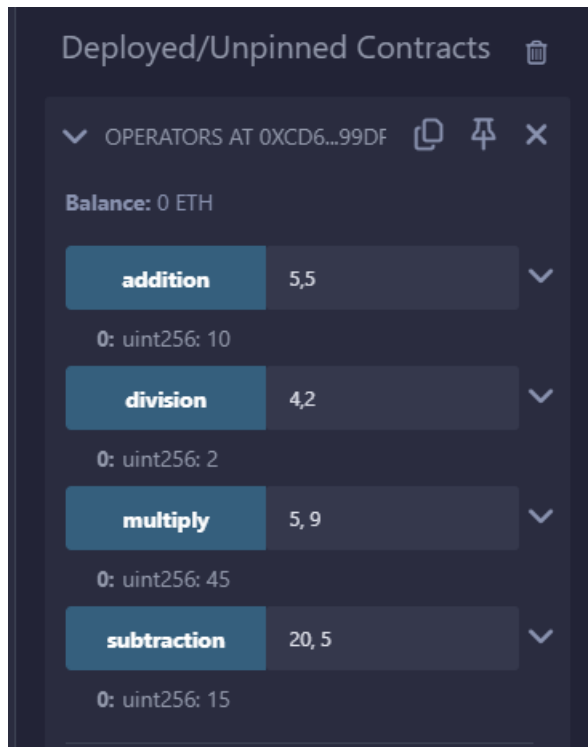
```
    }
```

```
    function division(uint256 a, uint256 b) public pure returns (uint256) {
```

```
        return a / b;
```

```
    }
```

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

**OUTPUT:****[III] Loops****CODE:**

```
pragma solidity ^0.5.0;  
contract LoopingTest {  
    uint256 storedData;  
    constructor() public {  
        storedData = 10;  
    }  
    function getResult() public pure returns (string memory) {  
        uint256 a = 10;  
        uint256 b = 2;  
        uint256 result = a + b;  
    }  
}
```

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

**OUTPUT:**

**[IV] 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:****[V] Strings****CODE:-**

```
pragma solidity ^0.5.0;

contract LearningStrings {
    string text;

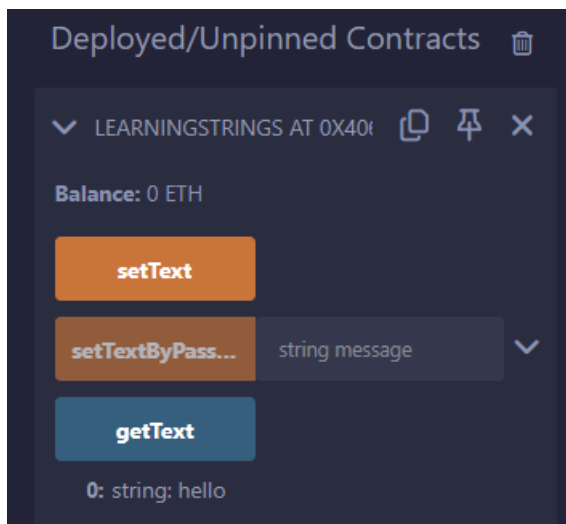
    function getText() public view returns (string memory) {
```



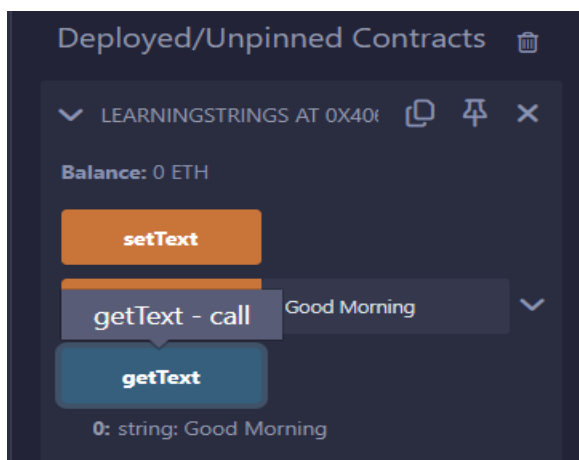
```
return text;
}
function setText() public {
text = "hello";
}
function setTextByPassing(string memory message) public {
text = message;
}
}
```

**OUTPUT :-**

Before setting new string value



After setting string value



**[VI] Arrays****CODE:**

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

```
pragma solidity 0.7.0;
```

```
contract Arrays {
```

```
    function initArray() public pure returns (uint256) {
```

```
        uint128[3] memory array = [1, 2, uint128(3)];
```

```
        return array[0];
```

```
    }
```

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

```
        uint128[] memory array = new uint128[](a);
```

```
        uint128 val = 5;
```

```
        for (uint128 j = 0; j < a; j++) {
```

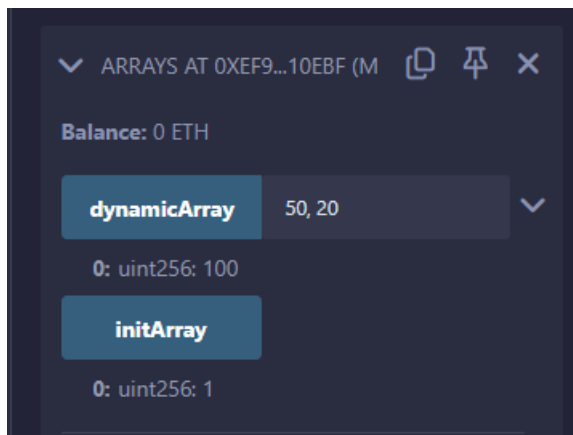
```
            array[j] = j * val;
```

```
        }
```

```
        return array[b];
```

```
    }
```

```
}
```

**OUTPUT:****[VII] Enums****CODE:**

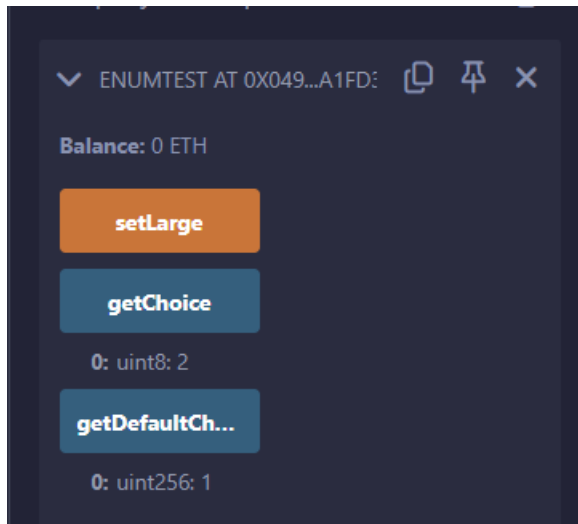
```
pragma solidity ^0.5.0;
```

```
contract enumTest {  
    enum FreshJuiceSize {  
        SMALL,  
        MEDIUM,  
        LARGE  
    }  
    FreshJuiceSize choice;  
    FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;  
  
    function setLarge() public {  
        choice = FreshJuiceSize.LARGE;  
    }  
  
    function getChoice() public view returns (FreshJuiceSize) {  
        return choice;  
    }  
  
    function getDefaultChoice() public pure returns (uint256) {
```

```

        return uint256(defaultChoice);
    }
}

```

**OUTPUT:****[VIII] 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);

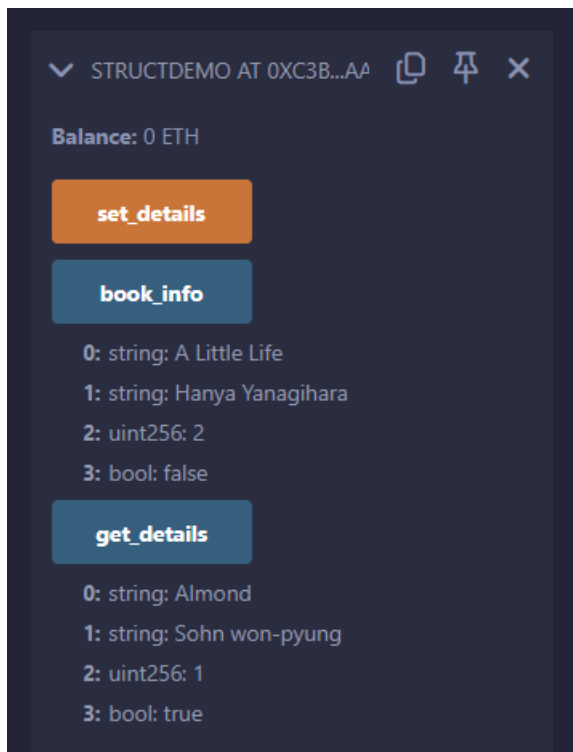
    function set_details() public {
        book2 = Book("Almond", "Sohn won-pyung", 1, true);
    }

    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:****[IX] Mappings****CODE:**

```
pragma solidity ^0.5.0;

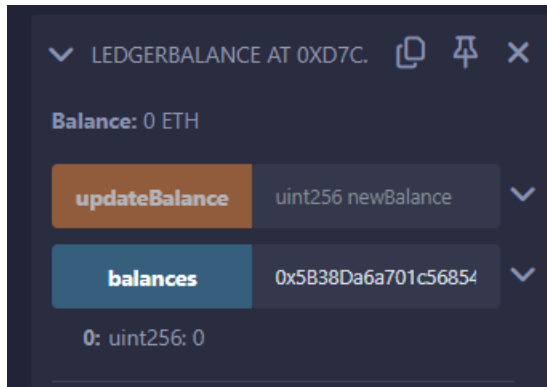
contract LedgerBalance {
    mapping(address => uint256) public balances;

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

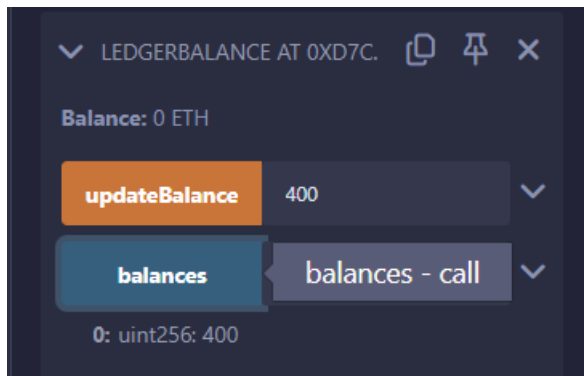
contract Updater {
    function updateBalance() public returns (uint256) {
        LedgerBalance ledgerBalance = new LedgerBalance();
        return ledgerBalance.balances(address(this));
    }
}
```

**OUTPUT:**

Before Updating Balance



After Updating Balance

**[X] Conversions****CODE:**

```
pragma solidity ^0.4.0;
```

```
contract Conversions {
```

```
    function intToUint(int8 a) public constant returns (uint256) {
```

```
        uint256 b = uint256(a);
```

```
        return b;
```

```
    }
```

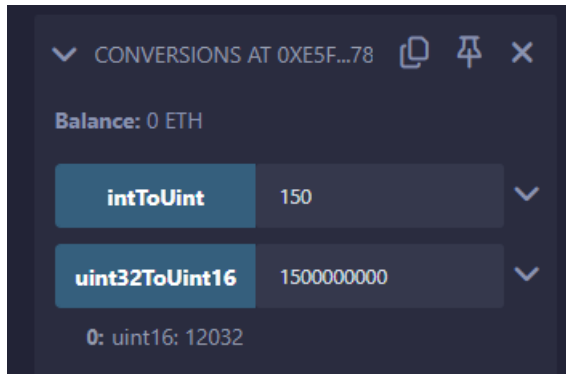
```
    function uint32ToUint16(uint32 a) public constant returns (uint16) {
```

```
        uint16 b = uint16(a);
```

```
        return b;
```

```
    }
```

}

**OUTPUT:****[XI] 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:****[XII] Special Variables****CODE:**

```

// SPDX-License-Identifier: MIT

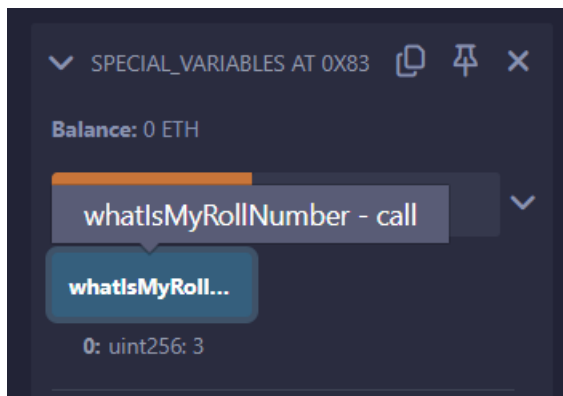
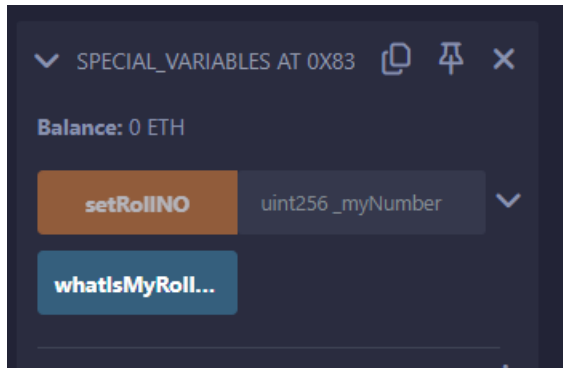
pragma solidity ^0.8.0;

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:**

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

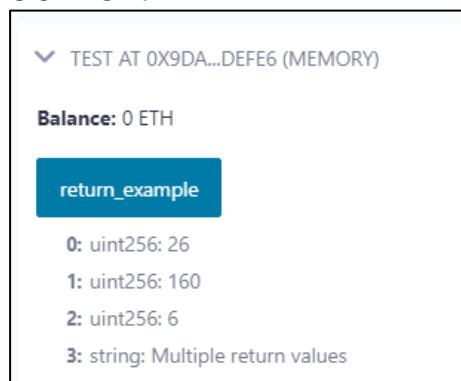
**[I] Functions**

**CODE:**

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



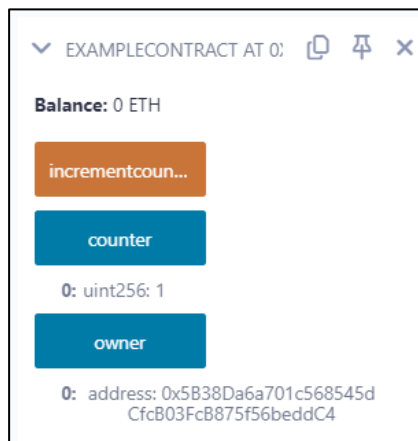
**[II] Function Modifiers****CODE:**

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

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

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

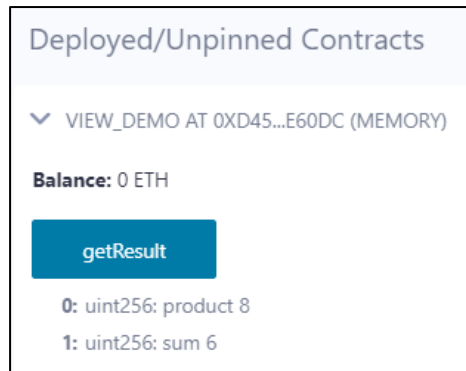
    function incrementcounter() public onlyowner {
        counter++;
    }
}
```

**[III] View Functions****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:**

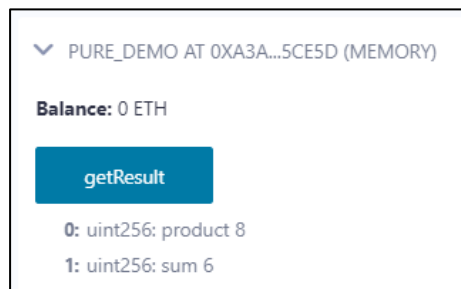


#### IV] Pure Function

pragma solidity >0.5.0;

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

#### OUTPUT:



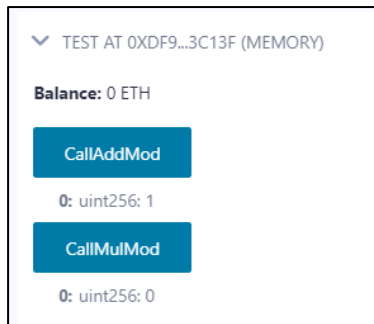
#### V] Mathematical Functions

##### 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:



## VI] Cryptographic Functions

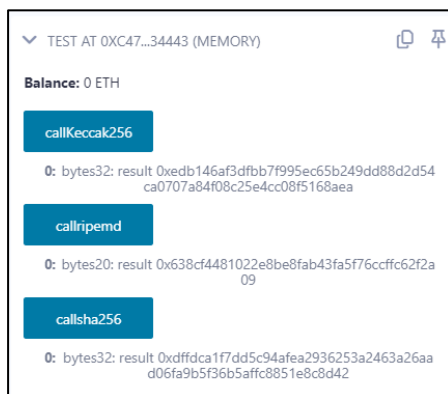
### CODE:

```
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");
    }
}
```

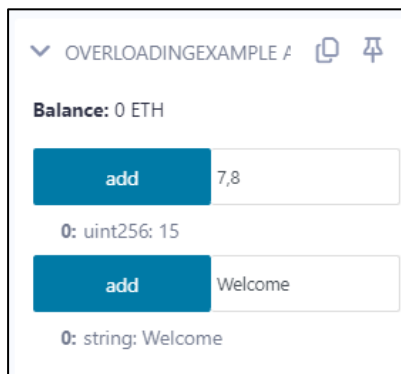
### OUTPUT:



**VII] Function Overloading**

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

```
contract OverloadingExample {  
    function add(uint256 a, uint256 b) public pure returns (uint256) {  
        return a + b;  
    }  
  
    function add(string memory a, string memory b)  
        public  
        pure  
        returns (string memory)  
    {  
        return string(abi.encodePacked(a, b));  
    }  
}
```

**OUTPUT:**

**PRACTICAL NO: 4**

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

**[A] Withdrawal Pattern, Restricted Access.**

**[I] Withdrawal Pattern**

**CODE:**

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

```
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:

▼ WITHDRAWALPATTERN AT

deposit

500

▼

unlock

500

▼

withdraw

200

▼

lockedbalance

0: uint256: 0

owner

0: address: 0x5B38Da6a701c56854d  
CfcB03FcB875f56beddC4

withdrawableb...

0: uint256: 500

**[II] 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!");
        _;
    }

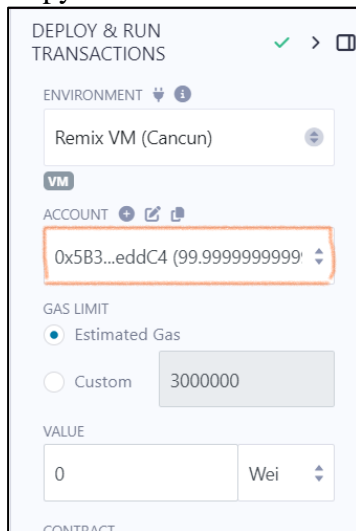
    function forceOwnerChange(address _newOwner)
        public
        payable
        costs(200 ether)
    {
        owner = _newOwner;
    }

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


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

**OUTPUT:**

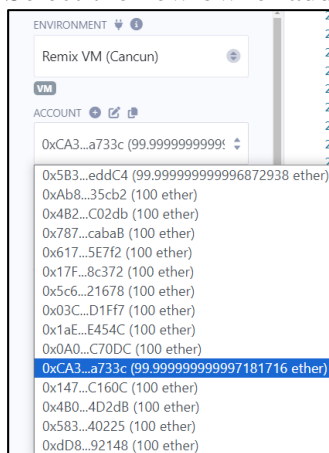
Copy the owner Address



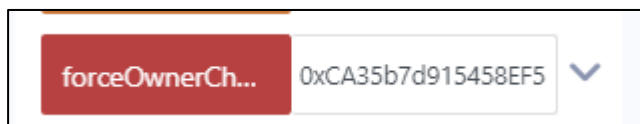
Paste it here



Select the new owner address



Paste the new owner address






Click on new time and owner

creationTime

0: uint256: 1721990676

owner

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

▼ RESTRICTEDACCESS AT 0X   

Balance: 0 ETH

changeOwner

0x5B38Da6a701c5685- ▼

disown

forceOwnerCh...

0xCA35b7d915458EF5 ▼

creationTime

0: uint256: 1721990676

owner

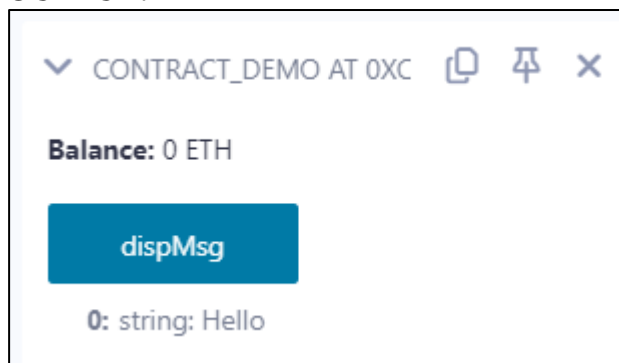
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

**[B] Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.****[I] Contracts****CODE:**

```
pragma solidity >0.5.0;

contract Contract_demo {
    string message = "Hello";

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

**OUTPUT:****[II] Inheritance****CODE:**

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

contract Parent {
    uint256 internal sum;

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

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

```

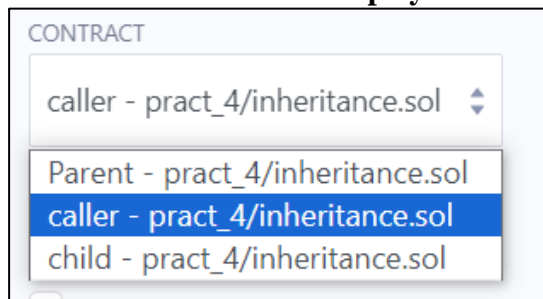
contract caller {
  child cc = new child();
  function testInheritance() public returns (uint256) {
    cc.setValue();
    return cc.getValue();
  }

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

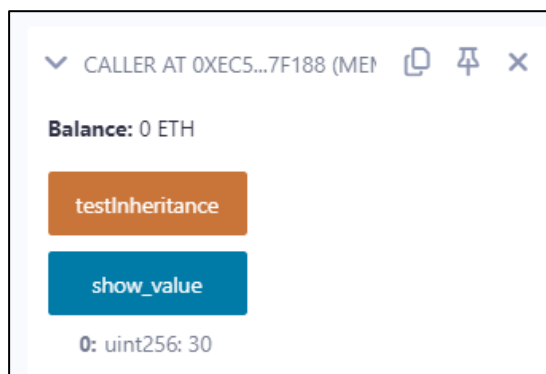
```

**OUTPUT:**

**Select caller contract to deploy in Contract and deploy**



**Click test Inheritance and then click on show\_value to view value**

**[III] Constructors****CODE:**

```

// SPDX-License-Identifier: MIT
pragma solidity >0.5.0;

// Creating a contract
contract constructorExample {
  string str;

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

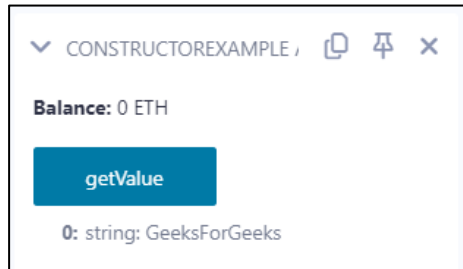
```

```

    }

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

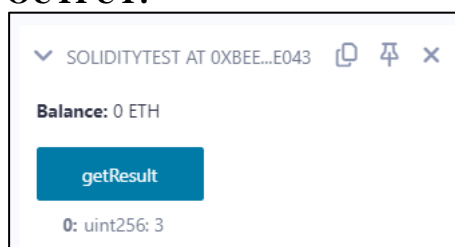
```

**OUTPUT:****[IV] Abstract Contracts****CODE:**

```

pragma solidity >0.5.0;
contract SolidityTest {
    constructor() public{
    }
    function getResult() public view returns(uint){
        uint a = 1;
        uint b = 2;
        uint result = a + b;
        return result;
    }
}

```

**OUTPUT:****[V] Interfaces****CODE:**

```

// SPDX-License-Identifier: MIT
pragma solidity >0.5.0;

interface Calculator {
    function getResult() external view returns(uint);
}

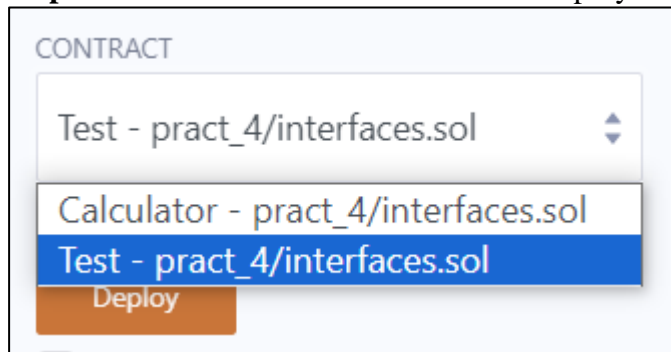
contract Test is Calculator {

```

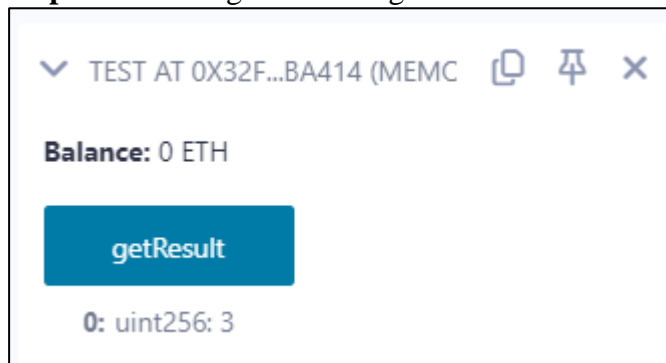
```
    constructor() public {}  
    function getResult() external view returns(uint){  
        uint a = 1;  
        uint b = 2;  
        uint result = a + b;  
        return result;  
    }  
}
```

**OUTPUT:**

**Step 1:** Select Test interface contract and deploy



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





**[C] Libraries, Assembly, Events, Error handling.****[I] Libraries****CODE:**

```
// Create library myLib.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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

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

// create library.sol
// SPDX-License-Identifier: MIT
pragma solidity >0.8.0;

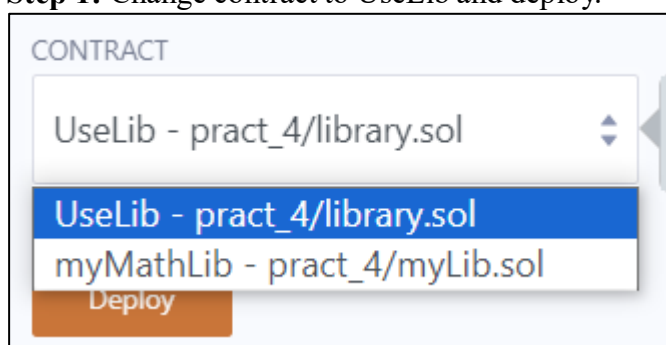
// put the myLib.sol path
import "pract_4/myLib.sol";

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

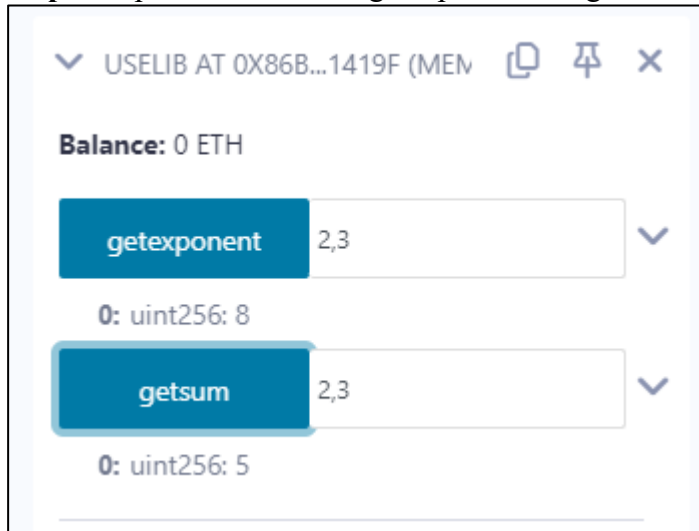
    function getexponent(uint256 x, uint256 y) public pure returns (uint256) {
        return myMathLib.exponent(x, y);
    }
}
```

**OUTPUT:**

**Step 1:** Change contract to UseLib and deploy.



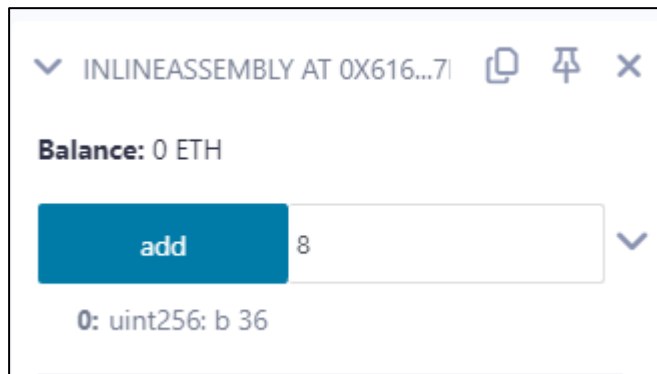
**Step 2:** Input values to both getexponent and getsum functions as below



### [II] 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:****[III] Events****CODE:**

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

// Creating a contract
contract eventExample {
    // Declaring state variables
    uint256 public value = 0;

    // Declaring an event
    event Increment(address owner);

    // Defining a function for logging event
    function getValue(uint256 _a, uint256 _b) public {
        emit Increment(msg.sender);
        value = _a + _b;
    }
}
```

**OUTPUT:**

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



**Step 2:** In the terminal, check for logs.

```
logs [
  {
    "from": "0xf02A102153D0f132032B7De5D19F43aA049052Dd",
    "topic": "0xfc3a67c9f0b5967ae4041ed898b05ec1fa49d2a3c22336247201d71be6f97120",
    "event": "Increment",
    "args": {
      "0": "0x5838Da6a701c568545dCfc803Fc8875f56beddC4",
      "owner": "0x5838Da6a701c568545dCfc803Fc8875f56beddC4"
    }
  }
]
```

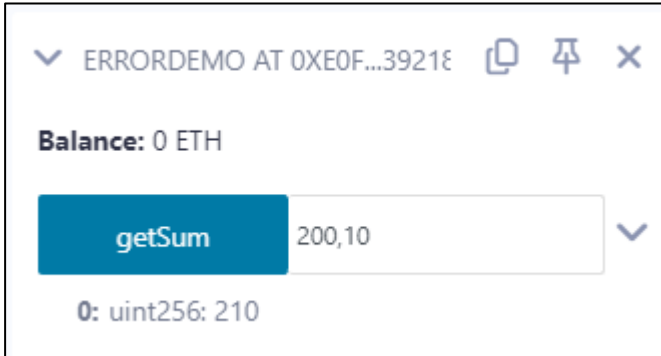
#### [IV] Error handling

##### CODE:

```
// SPDX-License-Identifier: MIT
pragma solidity >0.5.0;
contract ErrorDemo {
    function getSum(uint256 a, uint256 b) public pure returns (uint256) {
        uint256 sum = a + b;
        require(sum < 255, "Invalid");
        assert(sum < 255);
        return sum;
    }
}
```

##### OUTPUT:

Input values to the getSum function.



▼ ERRORDEMO AT 0xE0F...3921E

Balance: 0 ETH

getSum 200,10 ▼

0: uint256: 210

**PRACTICAL NO: 5**

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

**Step 1:** Download and install Node.js.

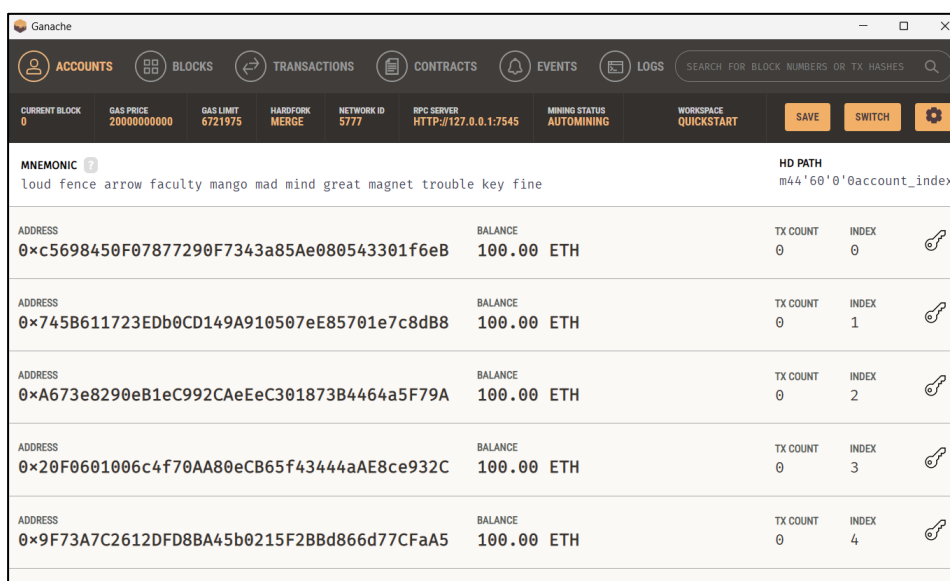
(<https://nodejs.org/en/download/prebuilt-installer>)

**Step 2:** Download and install Ganache (Ethereum Simulator).

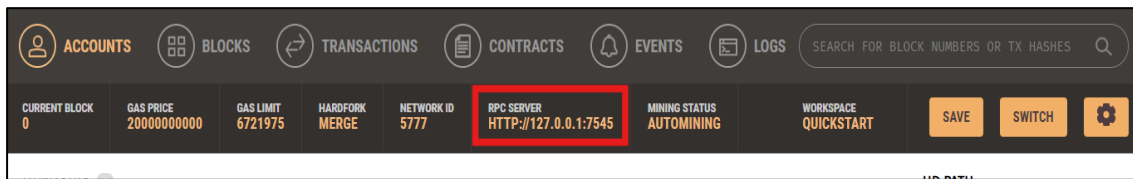
(<https://archive.trufflesuite.com/ganache/>)



**Step 3:** Click on 'QUICKSTART'



**Step 4:** Note down the RPC Server URL.



**Step 5:** Create a new file with the following code and save it as 'ethermine.js'.

Note: Replace the URL in Line 3 with your RPC server URL.

### Source code

```
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 ether on Ganache with coinbase address: ${coinbaseacc1}');

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

```
mine();
```

**Step 6:** Open Command Prompt. Install Web3 using 'npm install web3'

```
PS C:\Users\Admin\Desktop> npm install web3

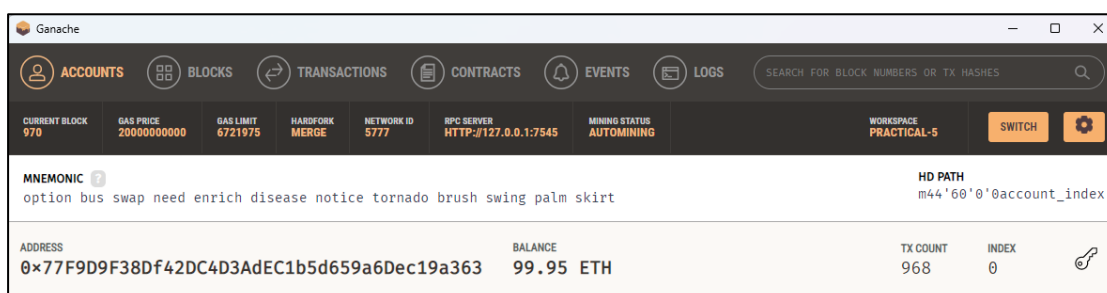
added 67 packages in 1m

21 packages are looking for funding
  run 'npm fund' for details
npm notice
npm notice New minor version of npm available! 10.7.0 -> 10.8.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.2
npm notice To update run: npm install -g npm@10.8.2
npm notice
```

**Step 7:** Run the JavaScript File using 'node ethermine.js'

```
PS C:\Users\Admin\Desktop> node ethermine.js
Mining ether on 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!
```

**Step 8:** Open Ganache and check the output.



**PRACTICAL NO: 6**

**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;  
}  
  
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(JSON.stringify(myCoin, null, 4));
```

## OUTPUT

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

```
E:\JS>node -v  
v20.15.1
```

Step 2: Step 2-> We need crypto-js node module to make our own blockchain. So install it using the following command

```
E:\JS>npm install crypto-js  
added 1 package in 38s
```

Step 3-> Run the above code in command prompt using command: node main.js

```
E:\JS>node main.js  
{  
  "chain": [  
    {  
      "index": 0,  
      "timestamp": "21/04/2023",  
      "data": "Genesis Block",  
      "previousHash": "0",  
      "hash": "32dd10ad547e8e81623998bdfafa2d8e9e3863fd252f5c3ea1cbea4ae26f54b1c"  
    },  
    {  
      "index": 1,  
      "timestamp": "22/04/2023",  
      "data": {  
        "amount": 4  
      },  
      "previousHash": "32dd10ad547e8e81623998bdfafa2d8e9e3863fd252f5c3ea1cbea4ae26f54b1c",  
      "hash": "eb78a02763c37cfc2b1c4e331df64ca34733e47e017ef320d92ae89b148de5a3"  
    },  
    {  
      "index": 2,  
      "timestamp": "22/04/2023",  
      "data": {  
        "amount": 8  
      },  
      "previousHash": "eb78a02763c37cfc2b1c4e331df64ca34733e47e017ef320d92ae89b148de5a3",  
      "hash": "946b1f95d7761daee4f0c5d33a671c003ef5682333fd9a2d182a73104e9aea88"  
    }  
  ]  
}
```