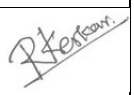
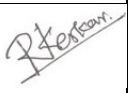
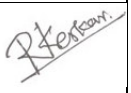
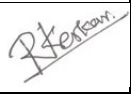
Finolex Academy of Management and Technology, Ratnagiri

Approved by AICTE, Recognized by DTE and Affiliated to The University of Mumbai

Department of MCA

Course Code: **MCALE331**	Course Name: **Blockchain Lab**	
Class: **SYMCA**	Semester: **III CBCGS (2 Year MCA)**	Academic Year: **2021-22**
Roll No:22	Name of Student: Shrinath Sunil Padave	

Index

MCALE331 – Block chain Lab				
SEM –III CBCGS			**SECOND HALF 2021**	
Expt. No.	**List of experiments**	**Date of Execution**	**Date of Assessment**	**Sign**
1	Cryptography	21/09/2021	05/10/2021	
2	Cryptocurrency	05/10/2021	02/11/2021	
3	Solidity Programming	02/11/2021	30/11/2021	
4	Ethereum	30/11/2021	13/12/2021	
5	Case Study: Use cases based on Hyper Ledger	13/12/2021	14/12/2021	

	HOPE Foundation's Finolex Academy Of Management And Technology, Ratnagiri Department of MCA		
Course Code: **MCALE331**	Course Name: **Blockchain Lab**		
Class: **SYMCA**	Semester: **III CBCGS (2 Year MCA)**		Academic Year: **2021-22**
Roll No: **22**	Name of Student: **Shrinath Sunil Padave**		
Assignment/Experiment No: **01**		Quiz Score:	6
Name of Assignment/Experiment: **Cryptography**			
Lab Objective applicable: **LOB1**. Impart a thorough understanding of cryptographic algorithm and hash functions.			
Lab Outcome applicable: **LO1**. Implement encryption algorithms and hash functions.			

Details of Quiz:

Q. No.	Question	Correct Answer	Given Answer
1	Cryptographic hash function takes an arbitrary block of data and returns _____.	d) fixed size bit string	d) fixed size bit string
2	Caesar Cipher is an example of _____.	c) Mono-alphabetic Cipher	c) Mono-alphabetic Cipher
3	RSA algorithm is best example of_____.	a) asymmetric key cryptography	a) asymmetric key cryptography
4	What is the number of round computation steps in the SHA-256 algorithm?	d) 64	d) 64
5	Merkle Tree is also known as _____.	a) Hash Tree	a) Hash Tree
6	Which of the following uses Merkle Trees structure?	c) Both A and B	c) Both A and B

Assignment/Experiment Evaluation:

Sr. No.	**Parameters**		**Marks Obtained**	**Out of**
1	Technical Understanding (Assessment based on Q and A through online quiz)		6	6
2	Neatness/Presentation		2	2
3	Punctuality		2	2
Date of Execution (DOE)	21/09/2021	Total Marks Obtained	10	10
Date of Assessment (DOA)	05/10/2021	Signature of Teacher		

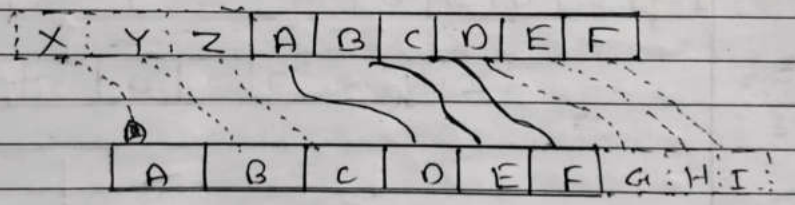
Practical No. 01

PAGE NO.:
DATE: / /

Cryptography

1. Symmetic Encryption using caesar cipher:

- The algorithem of caesar cipher having following features:
  - caesar cipher is simple and easy method of encryption technique.
  - It is simple type of substitution cipher.
  - Each letter of fixed plain text replaced by a letter with some fixed number of position down with alfabet.
- The following diagram depicts of working of caesar cipher algorithem implementation



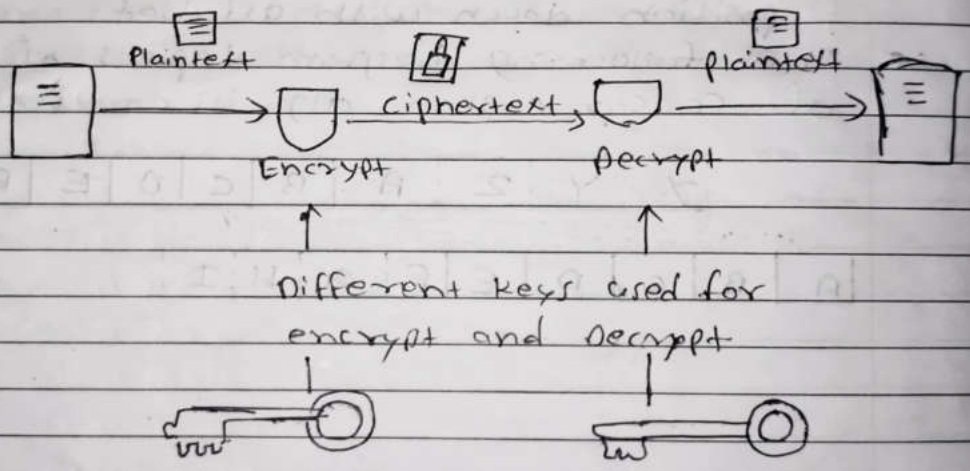
2. Asymmetic key cryptography using RSA:
Public key cryptography:

- Unlike Symmetic key cryptography, we do not find historical use of public key cryptography. It is relatively new concept.
- Symmetic key cryptography is suitable for government, military and big financial corporations were involved in classification communication.
- with spread of more unsecure computer

PAGE NO.:
DATE : / /

networks in last few decades a genuine need was felt to use cryptography at larger scale. The Symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to public key cryptosystems.

- The process of encryption and decryption is depicted in the following illustration:



RSA Cryptosystems:

- This cryptosystem is one the initial system. It remains most employed cryptosystem even today. The system was invented by three scholars Ron invest, Adi Shamir and len Adleman and hence, it is termed AS RSA cryptosystems.

PAGE NO.:
DATE: / /

- we will se two systems of cryptosystem firstly generation of key pairs and secondly encryption decryption algorithems.

Generation of RSA key pairs:

- Each person or a party who desires to participate in communication using encrypti-oin using encryption needs to generate a key pairs, namely public key and private key. The process followed in the generation of keys is described below:-
- find derived number (e):
  - Number e must greater than 1 and less than (p-1)(q-1).
- The e and (p-1)(q-1) must be coprime

Form the public key:

- The pair numbers (n, e) form the RSA public key and is made public.
- Interestingly, though n is part of the public key, difficulty in factorizing a large number prime, number ensures that attacker cannot find finite time the two primes (p & q) used to obtain n.
- Generate the private key:
  - private key d is calculated from p, q and e, for given n and e, there is unique number d.
  - number d is the inverse of e modulo (p-1)(q-1).

PAGE NO.:
DATE: / /

$ed = 1 \mod (p-1)(q-1)$

3 Hash functions: (SHA 256)

- SHA stands for secure hash algorithems. These are set of cryptographic hash functions. This functions can be used for various applications like password etc.
- This hashlib modules of python is used to implement common interface to different secure hash and messages digest algorithems. This hash algorithem included in this module are.
  - SHA 1: 160 bit hash.
  - SHA 224: 32 bit hash.
  - SHA 256: 32 bit hash
  - SHA 384: 32 bit hash.
  - SHA 512: 64 bit hash.
  - BLAKE algorithem.

4. Merkle Tree:

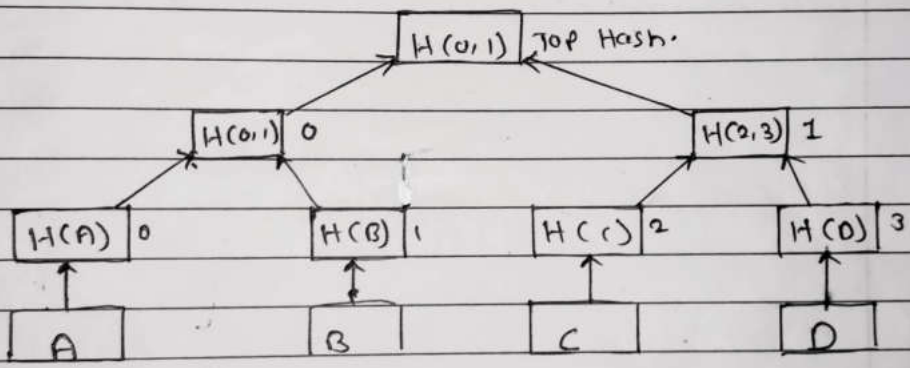
- merkle tree also known as hash tree, is a data structure used for data verification and synchronization.
- It is tree like data structure that store hash of each leaf node is hash of its child node. All the leaf nodes are the same depth and as far left as possible.

PAGE NO.:
DATE: / /

- It maintain data integrity and use hash functions for it.

Hash functions:

- Hash functions maps an input to fixed size output and output is called hash value. The output is unique for every unique input. This output is like fingerprint of data.



Applications:

- merkle trees are useful in distributed systems, where same data should be exist in multiple places.
- merkle trees can be used to check inconsistencies.
- It is used in bitcoin blockchain.

Practical No. 01

Write a program to implement symmetric Encryption using Caesar Cipher algorithm.

Code:

```
def encrypt(text,s):
        result = ""
        #transverse the plaintext
        for i in range(len(text)):
                char = text[i]
                # encrypt uppercase characters in plaintext
                if(char==""):
                        result +="";
                else:
                        if(char.isupper()):
                        #chr() function the characters that represents the specified
unicode#ord() function the number representing the unicode code of a specified characters
                                result += chr((ord(char) + 5 - 65) % 26 + 65)
                        else:
                                result += chr((ord(char) + 5 - 97) % 26 + 97)
                return result


text="CEASER CIPHER DEMO"
s=4
print("Plain Text:"+text)
print("shift pattern:"+str(s))
print("Cipher:"+encrypt(text,s))
```

Output

```
D:\Blockchain\python>python caeser.py
Plain Text:CEASER CIPHER DEMO
shift pattern:4
Cipher:H
```

Write a program to implement asymmetric Encryption using RSA algorithm.

Code:

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from binascii import hexlify #The message to be encrypted
message = b"Public and Private keys encryption"
private_key = RSA.generate(1024)
public_key = private_key.publickey()
print(type(private_key), type(public_key))
private_pem = private_key.export_key().decode()
public_pem = public_key.export_key().decode()
print(type(private_pem),type(public_pem))
with open('private_pem.pem','w') as pr:
    pr.write(private_pem)
with open('public_pem.pem','w') as pu:
    pu.write(private_pem)
pr_key = RSA.import_key(open('private_pem.pem','r').read())
pu_key = RSA.import_key(open('private_pem.pem','r').read())
print(type(pr_key),type(pu_key))
cipher = PKCS1_OAEP.new(key=pu_key)
cipher_text = cipher.encrypt(message)
print(cipher_text)
decrypt = PKCS1_OAEP.new(key=pr_key)
decrypted_message = decrypt.decrypt(cipher_text)
print(decrypted_message)
```

Output:

```
D:\Blockchain\python>pip install pycryptodome
Collecting pycryptodome
  Downloading pycryptodome-3.12.0-cp35-abi3-win_amd64.whl (1.8 MB)
     |████████████████████████████████| 1.8 MB 652 kB/s
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.12.0
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\Lenovo\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
```

```
D:\Blockchain\python>python rsa.py
<class 'Crypto.PublicKey.RSA.RsaKey'> <class 'Crypto.PublicKey.RSA.RsaKey'>
<class 'str'> <class 'str'>
<class 'Crypto.PublicKey.RSA.RsaKey'> <class 'Crypto.PublicKey.RSA.RsaKey'>
b'0\xc8\x00\x14\x97Oj\x00=\x01\xa8\x8f|\xc2\xa7\x07\xc9\xa8\x98u\x13\xe4\xd4\x15~\xdfW/\x7fw\x13=\x1aPNW\rM\x87g\x0f\x96
\xe6\x19B\xb5\xeb\x1cA\x98\xc4*\xf5\xecE`>7\xf6\x83\xe0\xf0\x13)K\x94\xaf"\xdc;V\x93_I,\xe5\xce\xadU\x0b\xf9O\xff\x82\xd
e\x9c\x0f\xd7\xc3;\xba\x8d\x92\xacy\xf4\x94e\xdb\xa0\x138\xc9\xc8\x8d\x86\xb1d\xe01\xfa\xff\n\xf0\xa0\xc62\x1f_\x1e;\x84
L\xf2\xda\xbfg\x8c'
b'Public and Private keys encryption'
```

Write a program to hash the message using SHA-256 Hash Function and print Hash Value, hexadecimal equivalent value, digest size and block size.

Code:

```
import hashlib

string="hello how are you?"

encoded=string.encode()

result = hashlib.sha256(encoded)

print("string : ", end = "")

print(string)

print("Hash Value : ",end ="")

print(result)

print("Hexadecimal equivalent : ", result.hexdigest())

print("Digest size : ",end="")

print(result.digest_size)

print("Block Size : ", end ="")

print(result.block_size)
```

Output:

```
D:\Blockchain\python>python sha.py
string : ishan
Hash Value : <sha256 _hashlib.HASH object @ 0x000002202DAF8A10>
Hexadecimal equivalent :  7290e492345306186996df9f3385a3c986b6e3914a6d2563dfc69ff060a77098
Digest size : 32
Block Size : 64
```

Write a program to create Merkle Tree and print root hash, depth of tree, levels of tree, tree nodes.

Code:

```
const { MerkleTree } = require('../OUTLINE')
const SHA256 = require('crypto-js/sha256')

const leaves = ['a', 'b', 'c'].map(x => SHA256(x))
const tree = new MerkleTree(leaves, SHA256)
const root = tree.getRoot().toString('hex')
const leaf = SHA256('a')
const proof = tree.getProof(leaf)
console.log(tree.verify(proof, leaf, root)) // true


const badLeaves = ['a', 'x', 'c'].map(x => SHA256(x))
const badTree = new MerkleTree(badLeaves, SHA256)
const badLeaf = SHA256('x')
const badProof = tree.getProof(badLeaf)
console.log(tree.verify(badProof, leaf, root))
console.log(tree);
```

Output:

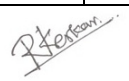
```
PS D:\shrinath\webpract\script> node test.js
true
false
MerkleTree {
  duplicateOdd: false,
  hashLeaves: false,
  isBitcoinTree: false,
  leaves: [
    <Buffer ca 97 81 12 ca 1b bd ca fa c2 31 b3 9a 23 dc 4d a7 86 ef f8 14 7c 4e 72 b9 80 77 85 af ee 48 bb>,
    <Buffer 3e 23 e8 16 00 39 59 4a 33 89 4f 65 64 e1 b1 34 8b bd 7a 00 88 d4 2c 4a cb 73 ee ae d5 9c 00 9d>,
    <Buffer 2e 7d 2c 03 a9 50 7a e2 65 ec f5 b5 35 68 85 a5 33 93 a2 02 9d 24 13 94 99 72 65 a1 a2 5a ef c6>
  ],
  layers: [
    [
      <Buffer ca 97 81 12 ca 1b bd ca fa c2 31 b3 9a 23 dc 4d a7 86 ef f8 14 7c 4e 72 b9 80 77 85 af ee 48 bb>,
      <Buffer 3e 23 e8 16 00 39 59 4a 33 89 4f 65 64 e1 b1 34 8b bd 7a 00 88 d4 2c 4a cb 73 ee ae d5 9c 00 9d>,
      <Buffer 2e 7d 2c 03 a9 50 7a e2 65 ec f5 b5 35 68 85 a5 33 93 a2 02 9d 24 13 94 99 72 65 a1 a2 5a ef c6>
    ],
    [
      <Buffer e5 a0 1f ee 14 e0 ed 5c 48 71 4f 22 18 0f 25 ad 83 65 b5 3f 97 79 f7 9d c4 a3 d7 e9 39 63 f9 4a>,
      <Buffer 2e 7d 2c 03 a9 50 7a e2 65 ec f5 b5 35 68 85 a5 33 93 a2 02 9d 24 13 94 99 72 65 a1 a2 5a ef c6>
    ],
    [
      <Buffer 70 75 15 2d 03 a5 cd 92 10 48 87 b4 76 86 27 78 ec 0c 87 be 5c 2f a1 c0 a9 0f 87 c4 9f ad 6e ff>
    ]
  ],
  sortLeaves: false,
  sortPairs: false,
  sort: false,
  fillDefaultHash: null,
  hashFn: [Function (anonymous)]
}
PS D:\shrinath\webpract\script>
```

	HOPE Foundation's Finolex Academy Of Management And Technology, Ratnagiri Department of MCA		
Course Code: **MCALE331**	Course Name: **Blockchain Lab**		
Class: **SYMCA**	Semester: **III CBCGS (2 Year MCA)**		Academic Year: **2021-22**
Roll No: **22**	Name of Student: **Shrinath Sunil Padave**		
Assignment/Experiment No: **02**		Quiz Score:	6
Name of Assignment/Experiment: **Cryptocurrency**			
Lab Objective applicable: **LOB2**. Understand the concepts of Bitcoin and Smart Contract.			
Lab Outcome applicable: **LO2**. Construct a bitcoin blocks and validating.			

Details of Quiz:

Q. No.	Question	Correct Answer	Given Answer
1	Bitcoins are created as a reward for a process known as _____.	C. mining	C. mining
2	Transactions are verified by network nodes through cryptography and recorded in a public distributed ledger called a _____.	A. blockchain	A. blockchain
3	To claim the reward, a special transaction called a_____.	D. coinbase	D. coinbase
4	Which of the following statement is true about bitcoin?	D. All of the above	D. All of the above
5	The size of block in bitcoin is limited to _____.	B. 1MB	B. 1MB
6	Bitcoin have a central authority.	B. FALSE	B. FALSE

Assignment/Experiment Evaluation:

Sr. No.	**Parameters**		**Marks Obtained**	**Out of**
1	Technical Understanding (Assessment based on Q and A through online quiz)		6	6
2	Neatness/Presentation		2	2
3	Punctuality		2	2
Date of Execution (DOE)	05/10/2021	Total Marks Obtained	10	10
Date of Assessment (DOA)	02/11/2021	Signature of Teacher		

PAGE NO.:
DATE: / /

Practical No. 02

Cryptocurrency

- **Bitcoin:**

Bitcoin is a decentralized digital currency, without single bank or single administrator, that can be sent from one user to another via peer-to-peer bitcoin network, without need any intermediatory. Transactions are rectified by network nodes through cryptography and recorded in public distributed ledger, called blockchain. It was invented in 2008 by anonymous developer satoshi nakamoto.

- **Block:**

A Block in blockchain can be thought as page in ledger, whic contain pages of transactions, the all new transactions are recorded in current block. The transaction that are recorded in the block are immutable.

- **Blockchain:**

Blockchain is a system that recording some information in the manner that it difficult to change, the Blockchain is "distributed immutable ledger, essentially the Block-

chain digital ledger of transactions that can't be duplicated, and distributed across the entire network computer system on the network. Blockchain solve the problem of double spending by using transaction based ledger and timestamp.

- Immutable ledger
  The word immutable means "cannot be changed" and ledger is fancy term for record, thusth immutable ledger in the record that cannot be changed. In blockchain each block store the hash of previous block, so tempring a block is not possible, If someone changed the content of block the all further blocks become invalid, because the previous hash of next block will be changed. In blockchain every miner has a copy of that blockchain, before validating any change, the 51% network must be ready for that change.

- Public and private Blockchain:
  public Blockchain:
  It is also called permissionless blockchain anyone can join these network, anyone can mine the block, A public blockchain has substantial amount of

PAGE NO.:
DATE: / /

computational power which is neccessary to maintain a distributed power which is neccessary. TO maintain state the in public blockchain, public blockchain use algorithems like Pow and Pos. The famous bitcoin use the consensus machanism proof-of-work, where miner need to solve complex hash puzzle.

private Blockchain:
private blockchain develope for utilizing blockchain technology for corporate businesses. In private blockchain users need permission to join the blockchain. Authorize entities can know the transactions in the network. The permissioned blockchain use the consensus machanism like paxos, Raft, BFT, PBFT. The transactions are private in permission blockchain like public blockchain transactions are not openly available.

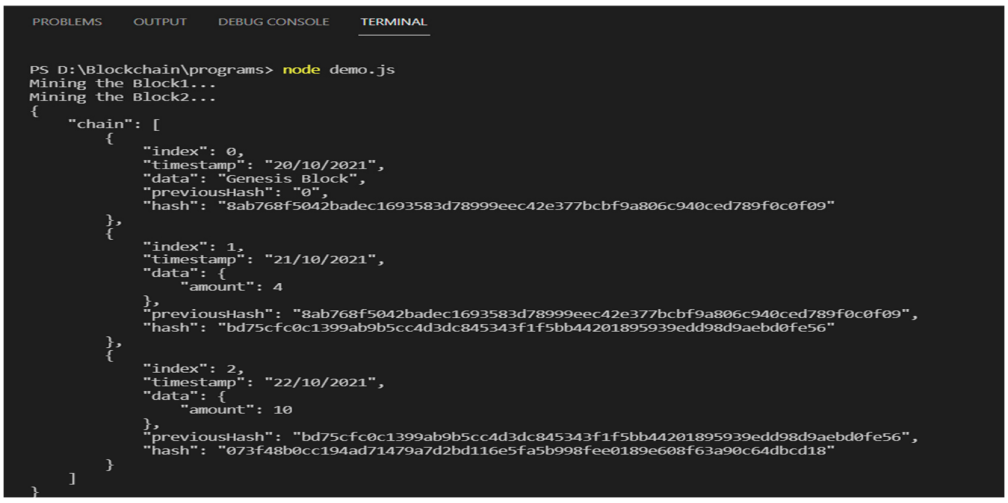
Practical 2

A) Write a program to create the chain with Genesis block and adding block into blockchain and validating the chain for any alteration. (Part-I)

```
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, "20/10/2021","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);
}
}
 let MCoin = new Blockchain();
console.log('Mining the Block1...')
MCoin.addBlock(new Block(1,"21/10/2021", { amount: 4}));
console.log('Mining the Block2...')
MCoin.addBlock(new Block(2,"22/10/2021", { amount: 10}));
console.log(JSON.stringify(MCoin,null,4));
```

Output:



B) Write a program to implementing proof of work for blockchain.

Blockchain.js

```
const SHA256=require('crypto-js/sha256');
class Transaction
{
   constructor(fromAddress,toAddress,amount)
   {
      this.fromAddress=fromAddress;
      this.toAddress=toAddress;
      this.amount=amount;
   }
}
class Block
{
   constructor(timestamp,transactions,previousHash ='')
   {
   this.timestamp = timestamp;
   this.transactions = transactions;
   this.previousHash = previousHash;
   this.hash = this.calculateHash();
   this.nonce=0;
   }
calculateHash()

{
   return SHA256(this.previousHash + this.timestamp + JSON.stringify(this.transactions)
+this.nonce).toString();
}
mineBlock(difficuity)
{
   while(this.hash.substring(0,difficuity) !== Array(difficuity + 1).join("0"))
```

```
    {
    this.nonce++;
    this.hash = this.calculateHash();

    }
      console.log("Block mined: " + this.hash);
}
}
class Blockchain
{
    constructor()
    {
    this.chain =[this.createGenesisBlock()];
    this.difficuity =2;
    this.pendingTransactions=[];
    this.miningReward=100;
    }

createGenesisBlock()
{
    return new Block("20/10/2021","Genesis Block","0");

}
getLatestBlock()
{
    return this.chain[this.chain.length - 1];

}
minePendingTransactions(miningRewardAddress)
{
```

```
    let block = new Block(Date.now(), this.pendingTransactions);
    block.mineBlock(this.difficuity);
    console.log('Block Successfully mined');
    this.chain.push(block);
    this.pendingTransactions=[new Transaction(null,
miningRewardAddress,this.miningReward)];
}

createTransaction(transaction)
{
    this.pendingTransactions.push(transaction);
}
getBalanceOfAddress(address)
{
    let balance = 0;
    for(const block of this.chain)
    {
        for(const trans of block.transactions)
        {
            if(trans.fromAddress == address)
            {
                balance -=trans.amount;
            }
            if(trans.toAddress == address)
            {
                balance += trans.amount;
            }
        }
    }
    return balance;
}
```

```
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 SCoin = new Blockchain();
 SCoin.createTransaction(new Transaction('address1','address2',100));
 SCoin.createTransaction(new Transaction('address2','address1',50));
console.log('\n Strating the miner');
SCoin.minePendingTransactions('tata-address');
console.log('\n Balance of tata-address is' , SCoin.getBalanceOfAddress('tata-address'));
console.log('\n Strating the miner');
SCoin.minePendingTransactions('tata-address');
console.log('\n Balance of tata-address is' , SCoin.getBalanceOfAddress('address2'));
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS D:\Blockchain\programs\2> node blockchain.js
Strating the miner
Block mined: 001c42a448282c61aed21197808b8237b316475e95b21c07c6d645a21681aaac
Block Successfully mined
Balance of tata-address is 0
Strating the miner
Block mined: 00725da47b4c0e21d7163ecc34a2845dd8a55b115513cab93e9197fac9d462ad
Block Successfully mined
Balance of tata-address is 50
PS D:\Blockchain\programs\2>

C) Write a program to add multiple transactions into block and give reward to miner for successful

mining of block in blockchain.

```
const SHA256=require('crypto-js/sha256');
class Transaction
{
   constructor(fromAddress,toAddress,amount)
   {
      this.fromAddress=fromAddress;
      this.toAddress=toAddress;
      this.amount=amount;
   }
}
class Block
{
   constructor(timestamp,transactions,previousHash ='')
   {
   this.timestamp = timestamp;
   this.transactions = transactions;
   this.previousHash = previousHash;
   this.hash = this.calculateHash();
   this.nonce=0;
   }

calculateHash()

{
   return SHA256(this.previousHash + this.timestamp + JSON.stringify(this.transactions)
+this.nonce).toString();
}
mineBlock(difficuity)
{
   while(this.hash.substring(0,difficuity) !== Array(difficuity + 1).join("0"))
```

```
    {
    this.nonce++;
    this.hash = this.calculateHash();

    }
      console.log("Block mined: " + this.hash);
}
}
class Blockchain
{
    constructor()
    {
    this.chain =[this.createGenesisBlock()];
    this.difficuity =2;
    this.pendingTransactions=[];
    this.miningReward=100;
    }

createGenesisBlock()
{
    return new Block("20/10/2021","Genesis Block","0");

}
getLatestBlock()
{
    return this.chain[this.chain.length - 1];

}
minePendingTransactions(miningRewardAddress)
{
    let block = new Block(Date.now(), this.pendingTransactions);
    block.mineBlock(this.difficuity);
```

```
    console.log('Block Successfully mined');
    this.chain.push(block);
    this.pendingTransactions=[new Transaction(null, miningRewardAddress,this.miningReward)];
}

createTransaction(transaction)
{
    this.pendingTransactions.push(transaction);
}
getBalanceOfAddress(address)
{
    let balance = 0;
    for(const block of this.chain)
    {
        for(const trans of block.transactions)
        {
            if(trans.fromAddress == address)
            {
                balance -=trans.amount;
            }
            if(trans.toAddress == address)
            {
                balance += trans.amount;
            }



        }
    }
    return balance;
}

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 MCoin = new Blockchain();

 MCoin.createTransaction(new Transaction('address1','address2',100));
 MCoin.createTransaction(new Transaction('address2','address1',50));

console.log('\n Strating the miner');
MCoin.minePendingTransactions('tata-address');
console.log('\n Balance of tata-address is' , MCoin.getBalanceOfAddress('tata-address'));

console.log('\n Strating the miner');
MCoin.minePendingTransactions('tara-address');
console.log('\n Balance of tata-address is' , MCoin.getBalanceOfAddress('tata-address'));
```

D) Write a program to sign the transaction with private key and verify the signed transactions for blockchain.

```
const EC  = require('elliptic').ec;
const ec  = new EC('secp256k1');


 const key = ec.genKeyPair();
 const publicKey = key.getPublic('hex');
 const privateKey =  key.getPrivate('hex');


 console.log("private key : ",privateKey);
 console.log('Public key : ', publicKey);
```

Output:

```
PS D:\Blockchain\Pract1> node keygenerator.js
private key :  44dfc9e0bc751358753422a2ce32ce26e57975d156bfa664bd328352814a85ab
Public key :  047c96ed89dbe0ff40f3dd408b85ff89b874a9fba459c20a8c4f48e67b1e571723306205531aa83065cd1da0622b2f97cb2e752835fcf46c5cd634412868c9fab7
PS D:\Blockchain\Pract1>
```

Blockchain.js

```
const SHA256 = require('crypto-js/sha256');
const EC  = require('elliptic').ec;
const ec  = new EC('secp256k1');

class Transaction
{
   constructor(fromAddress,toAddress,amount)
   {
      this.fromAddress = fromAddress;
      this.toAddress = toAddress;
      this.amount = amount;
   }

   calculateHash()
   {
      return SHA256(this.fromAddress + this.toAddress + this.amount).toString();
   }

   signTransaction(signingKey)
   {
      if(signingKey.getPublic('hex') !== this.fromAddress)
      {
        throw new Error("You cannot sign transaction");
      }
      const hashTx = this.calculateHash();
      const sig =signingKey.sign(hashTx, 'base64');
      this.signature = sig.toDER('hex');
   }

   isValid()
   {
      if(this.fromAddress ===  null) return true;

      if(!this.signature || this.signature.length === 0)
      {
         throw new Error('No signature is this transction....!');
      }

      const publicKey = ec.keyFromPublic(this.fromAddress,'hex');
      return publicKey.verify(this.calculateHash(),this.signature);
   }
}

class Block
{
   constructor(timestamp, transactions, previousHash = '')
   {
      this.timestamp = timestamp;
```

```
        this.transactions = transactions;
        this.previousHash = previousHash;
        this.hash = this.calculateHash();
        this.nonce = 0;
    }
    calculateHash()
    {
        return SHA256(this.index + this.previousHash + this.timestamp + this.nonce +
JSON.stringify(this.data)).toString();
    }

    mineBlock(diffculty)
    {
        while(this.hash.substring(0,diffculty) !== Array(diffculty+1).join("0"))
        {
            this.hash = this.calculateHash();
            this.nonce++;
        }
        console.log("Block mined " + this.hash);
    }

    hasValidTransactions()
    {
        for(const tx of this.transactions)
        {
            if(!tx.isValid())
            {
                return false;
            }
        }
        return true;
    }
}
 class Blockchain
 {
     constructor()
     {
         this.chain = [this.createGenesisBlock()];
         this.diffculty = 4;
         this.pendingTransactions = [];
         this.miningReward = 100;
     }

     createGenesisBlock()
     {
         return new Block("10/ 08/ 2021","GenesisBlaock","0");
     }

     getLatestBlock()
     {
         return this.chain[this.chain.length -1];
```

```
 }

 minePendingTransactions(miningRewardAddress)
 {
  const rewardTx  = new Transaction(null,miningRewardAddress,this.miningReward);
  this.pendingTransactions.push(rewardTx);

  let block  = new Block(Date.now(),this.pendingTransactions,this.getLatestBlock.hash);
  block.mineBlock(this.diffculty);

  console.log("Block successfully mined....");
  this.chain.push(block);

  this.pendingTransactions = [];

}

addTransaction(transaction)
{
   if (!transaction.fromAddress || !transaction.toAddress)
   {
      throw new Error('Cannot add new transactions .....');
   }

   if (!transaction.isValid())
   {
      throw new Error('Caonnot add invalid transaction...!');
   }

   this.pendingTransactions.push(transaction);
}

getBalanaceAddress(address)
{
   let balance = 0;
   for (const block of this.chain)
   {
      for(const trans of block.transactions)
      {
         if(trans.fromAddress === address)
         {
            balance -= trans.amount;
         }
         if(trans.toAddress === address)
         {
            balance  += trans.amount;
         }
      }
   }
   return balance;
}
```

```
    isChainVlaid()
    {
        for(let i = 1 ; i < this.chain.length; i++)
        {
            const currentBlock = this.chain[i];
            const previousBlock = this.chain[i-1];

            if(!currentBlock.hasValidTransactions())
            {
                return false;
            }

            if(currentBlock.hash !== currentBlock.calculateHash())
            {
                return false;
            }
            if(currentBlock.previousHash !== previousBlock.hash)
            {
                return false;
            }
            return true;
        }
    }
}

module.exports.Blockchain = Blockchain;
module.exports.Transaction = Transaction;
```

mainDemo.js

```
const {Blockchain,Transaction} = require('./blockchain');

const EC  = require('elliptic').ec;

const ec  = new EC('secp256k1');

const myKey =
ec.keyFromPrivate('ee2d1d82aad3cbd7ecfd5fae8b5f463d63c97c7cebf7c10918c41a31bbf82cb7');

const myWalleteAddress = myKey.getPublic('hex');

let SPcoin = new Blockchain();

const Tx1  = new Transaction(myWalleteAddress, 'public key goes here', 10);

Tx1.signTransaction(myKey);

SPcoin.addTransaction(Tx1);

console.log("\n Starting the miner.....");

SPcoin.minePendingTransactions(myWalleteAddress);

console.log("\n Balance of sp is ",SPcoin.getBalanaceAddress(myWalleteAddress));
```

Output:

```
PS D:\Blockchain\Pract1> node mainDemo.js

 Starting the miner......
Block mined 00006b62033ef7667736bb984076e0b5d9aaf5fd05f0b2f0173eeac04a314af4
Block successfully mined....

 Balance of sp is  90
PS D:\Blockchain\Pract1>
```

	HOPE Foundation's Finolex Academy Of Management And Technology, Ratnagiri Department of MCA		
Course Code: **MCALE331**	Course Name: **Blockchain Lab**		
Class: **SYMCA**	Semester: **III CBCGS (2 Year MCA)**		Academic Year: **2021-22**
Roll No: **22**	Name of Student: **Shrinath Sunil Padave**		
Assignment/Experiment No: **03**		Quiz Score:	6
Name of Assignment/Experiment: **Solidity Programming**			
Lab Objective applicable: **LOB3**. Understand the concepts of Solidity language.			
Lab Outcome applicable: **LO3**. Construct a smart contract in Ethereum.			

Details of Quiz:

Q. No.	Question	Correct Answer	Given Answer
1	In which programming language is Smart Contracts written?	B. solidity	B. solidity
2	Which among the following statements are/is true? S1: Smart contract can interact with other smart contracts. S2: Smart contract can call an API on the web.	A. S1 only	A. S1 only
3	Which among the following statements are/is true with respect to Solidity? S1: Solidity is compiled to bytecode. S2: A single Solidity file have several smart contracts.	C. Both S1 and S2	C. Both S1 and S2
4	What are state-machines in EVM?	D. All of the above	D. All of the above
5	In Solidity, if you divide integer -5 by integer 2, the result is _____.	A. -2, because the decimal is truncated (rounded to zero)	A. -2, because the decimal is truncated (rounded to zero)
6	Once an Ethereum smart contract is deployed to the blockchain, it cannot:	D. All of the above	D. All of the above

Assignment/Experiment Evaluation:

Sr. No.	**Parameters**	**Marks Obtained**	**Out of**
1	Technical Understanding (Assessment based on Q and A through online quiz)	6	6
2	Neatness/Presentation	2	2
3	Punctuality	2	2

Date of Execution (DOE)	02/11/2021	Total Marks Obtained	10	10
Date of Assessment (DOA)	30/11/2021	Signature of Teacher		

Practical No. 03

Solidity

- Ethereum:
Ethereum is decentralized blockchain network that run smart contracts i.e. applications that run without any posibility of downtime censorship, fraud or thirtparty interference.

- Solidity:
- Solidity is high level object oriented programming language for writtening smart contracts for Ethereum blockchain.
- Smart contracts are the program which govern the behaviour of accounts within the Ethereum state.
- solidity was influenced by c++, python and javascript and it designed to target the Ethereum virtual machine.
- solidity is statically typed language it supports inheritence, libraries and complex user defined types among other features.
- with solidity you can create contracts for uses such as voting crowdfunding, blind auctions and multi-signature wallets.
- when deploying contracts, you should use latest version of

PAGE NO.:
DATE: / /

solidity. Because new version has some new features as well as some bugs are fixed. we are currently using 0.x. solidity p version for first pace.

- Remix:
  Remix is online Integrated development Environment for writing smart contracts in solidity, where we can write the smart contracts, deploy the contracts on test network and test the contracts.

- structure of contracts:

```
pragma solidity ^0.5.0;

contract SimpleStorage {

    uint storeData // state variable.
        public.
    function functionname(uint) public
            view returns(uint){
        return storeData;
    //...
    }
}
```

* If we make state variable public we get automatically getter function from solidity.

- functions:
  - functions are executable unit of code within the contract.

e.g.

```
function bid() payable {
    //.....
}.
```

- function modifiers:
  - function modifiers can be used to amend the semantics of function in declarative way.
  - overloading that is same function name with different parameters but having same modifier name with different parameter not possible.

```
modifier OSeller() {
    require(
        msg.sender == seller,
        "only seller can call this".
    );
    _;
}
function abort() public view
        oseller {
}
```

PAGE NO.:
DATE: / /

- Variables:
  # State variables: whose values are permantly store in contract storage.

  # local variables: values are present till executing that function.

  # global variables: The variables that use to access values of global namespace, use to get information about Blockchain

- View functions:
  View function ensures that they will not modify the state, e.g.
  - modifying state variable.
  - Emmiting events
  - creating other contracts.
  - sending ethers.

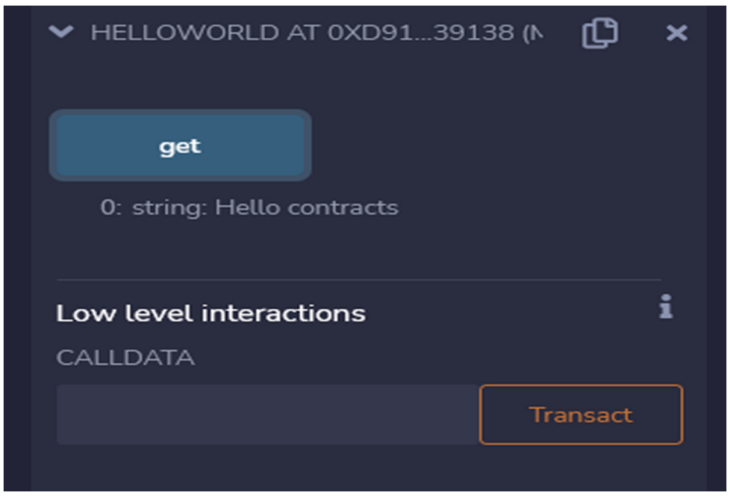
- pure functions:
  pure functions ensures that it not read or modify the state variable during execution.

1) **Write a solidity smart contract to display hello world message.**

Code:

```
pragma solidity >= 0.4.16 < 0.7.0;
contract HelloWorld{
    function get() public pure returns(string memory){
        return "Hello contracts";
    }
}
```

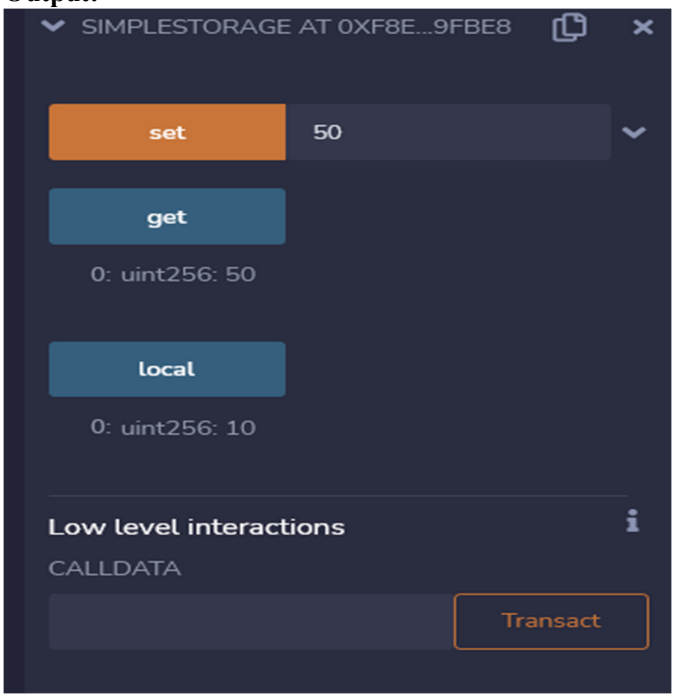
Output:



2) **Write a solidity smart contract to demonstrate state variable, local variable and global variable.**

```
pragma solidity 0.7.0;
contract SimpleStorage {
uint storedData; //state variable

function set(uint x) public {
storedData = x;
}

function get() public view returns (uint) {
return storedData;
}
function local() public pure returns(uint){
   uint a=10;
   return a;
}
}
```

Output:

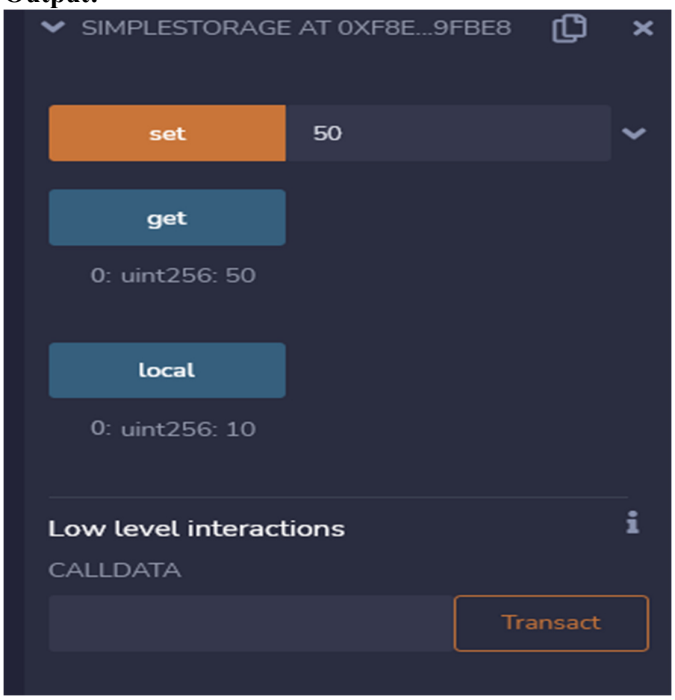


3) Write a solidity smart contract to demonstrate getter and setter methods.

Code:

```
pragma solidity 0.7.0;
contract SimpleStorage {
uint storedData; //state variable

function set(uint x) public {
storedData = x;
}

function get() public view returns (uint) {
return storedData;
}
function local() public pure returns(uint){
   uint a=10;
   return a;
}
}
```

Output:

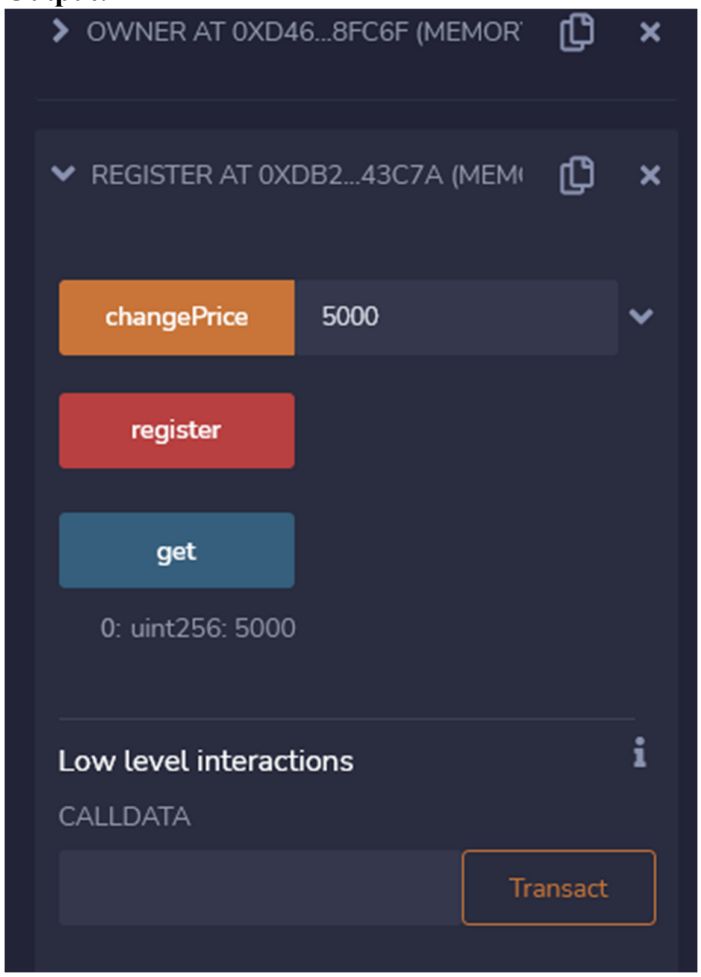


4) Write a solidity smart contract to demonstrate function modifier.

Code:

```
pragma solidity ^0.5.0;
contract Owner {
address owner;
   constructor() public
   {
      owner = msg.sender;
   }
   modifier onlyOwner
   {
      require(msg.sender == owner);
      _;
   }
   modifier costs(uint price)
   {
   if (msg.value >= price)
   {
      _;
   }
}
}
contract Register is Owner {
   mapping (address => bool) registeredAddresses;
   uint price;
   constructor(uint initialPrice) public
   {
       price = initialPrice;
   }
   function register() public payable costs(price)
   {
      registeredAddresses[msg.sender] = true;
   }
   function changePrice(uint _price) public onlyOwner
   {
      price = _price;
   }
function get() public view returns(uint)
   {
      return price;
   }
}
```

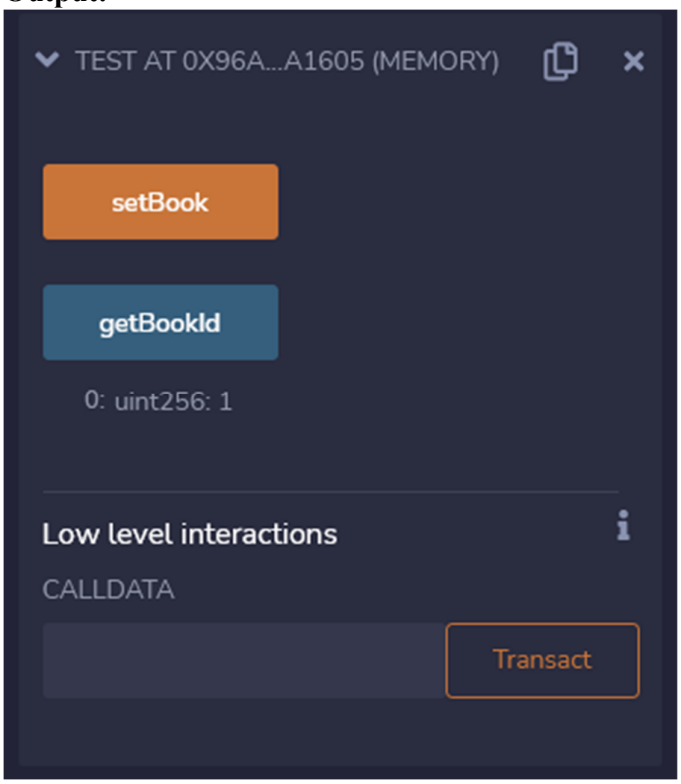
Output:


OWNER AT 0XD46...8FC6F (MEMOR
REGISTER AT 0XDB2...43C7A (MEM
changePrice
5000
register
get
0: uint256: 5000
Low level interactions
CALLDATA
Transact

5) Write a solidity smart contract to demonstrate use of structure.

```
pragma solidity ^0.5.0;

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

  function setBook() public {
    book = Book('Learn Java', 'TP', 1);
  }
  function getBookId() public view returns (uint) {
    return book.book_id;
  }
}
```

Output:

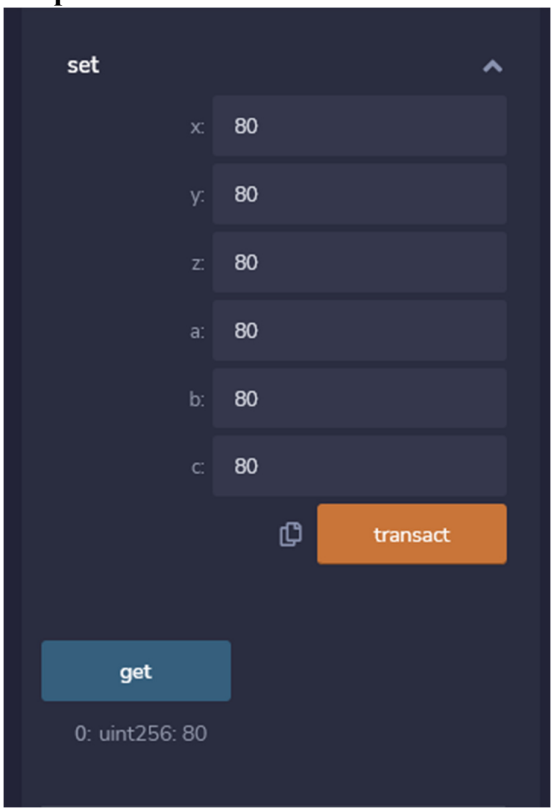


6) **Write a solidity smart contract to calculate percentage of marks obtained by students for six subject in final examination.**

Code:

```
pragma solidity 0.7.0;
contract SimpleStorage {
uint result; //state variable

function set(uint x,uint y,uint z,uint a,uint b,uint c) public {
 uint temp=(x+y+z+a+b+c);
 result=temp/6;
}

function get() public view returns (uint) {
return result;
}
}
```

Output:



7) **Write a solidity smart contract to find the factorial of entered number.**

Code:

```
pragma solidity 0.7.0;
contract Factorial {
  function fact(uint x) public view returns (uint y) {
    if (x == 0) {
      return 1;
    }
    else {
      return x*fact(x-1);
    }
  }
}
```

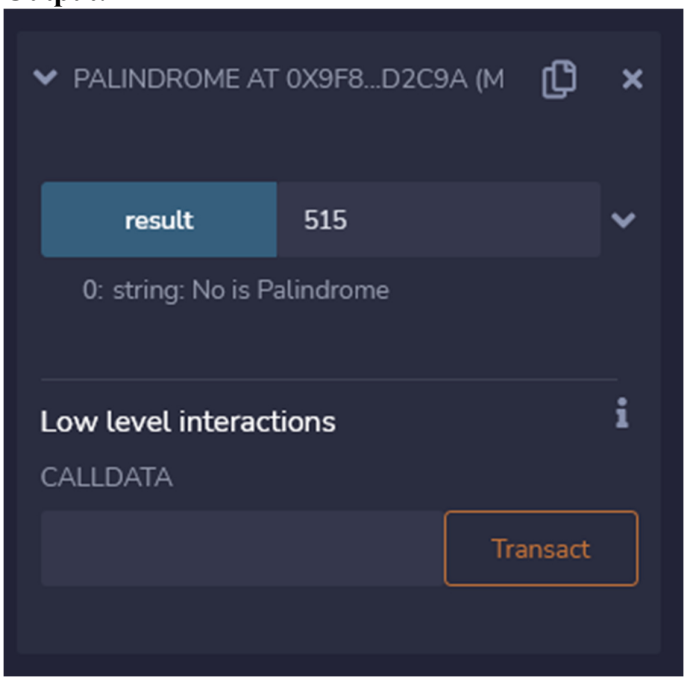
Output:

8) **Write a solidity smart contract to check whether entered number is palindrome or not.**

Code:

```
pragma solidity 0.7.0;
contract Palindrome{
function result(uint x) public view returns(string memory){
uint temp=x;
uint rev;
uint rem;
while(x>0){
rem=x%10;
rev=rev*10+rem;
x=x/10;
}
   if(temp==rev){
      return "No is Palindrome";
   }
   else{
      return "No is not Palindrome";
   }
}
}
```

Output:

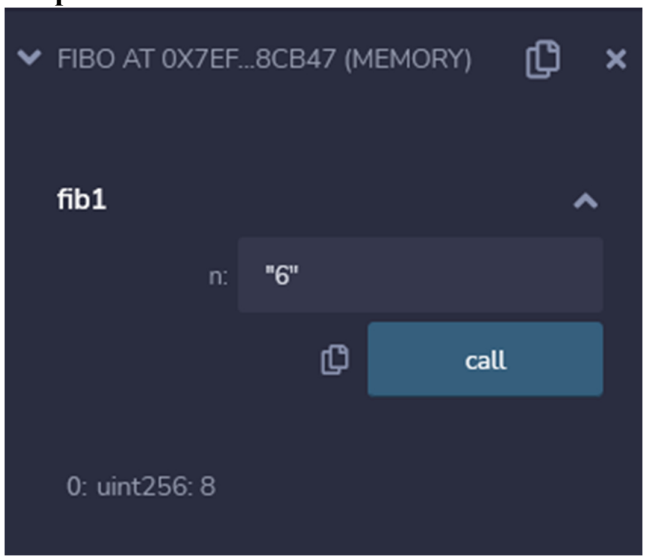


9) **Write a solidity smart contract to generate Fibonacci Series upto given number.**

Code:

```
pragma solidity 0.5.0;
contract Fibo{
   function fib1(uint n) external pure returns(uint) {
   uint[] memory sequence = new uint[](n+1);
   for (uint i = 0; i <= n; i++) {
      if (i <= 1) {
         sequence[i] = i;
      } else {
         sequence[i] = sequence[i -1] + sequence[i -2];
      }
   }
   return sequence[n];
}
```

Output:

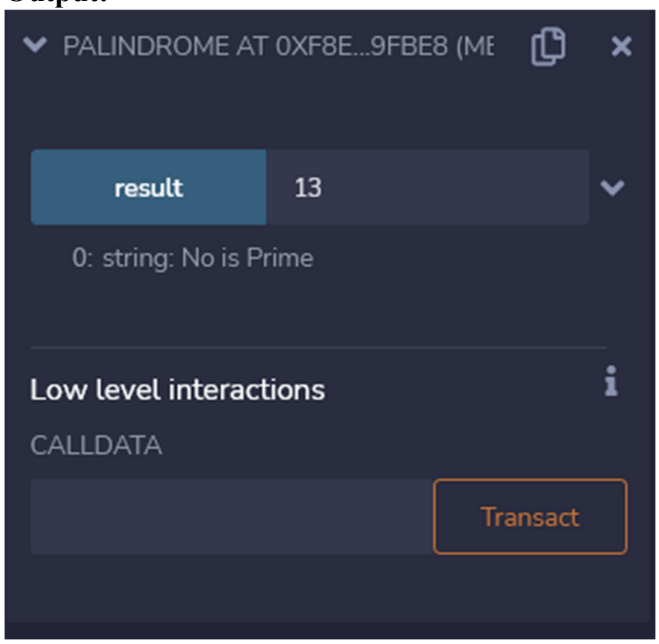


10) Write a solidity smart contract to check whether entered number is prime number or not.

Code:

```
pragma solidity 0.7.0;
contract Palindrome{
function result(uint x) public view returns(string memory){
uint temp=1;

for(uint i=2;i<=(x/2);i++)
{
   if(x%i==0){
      temp=0;
      break;
   }
}
   if(temp==1){
      return "No is Prime";
   }
   else{
      return "No is not prime";
   }
}
}
```

Output:

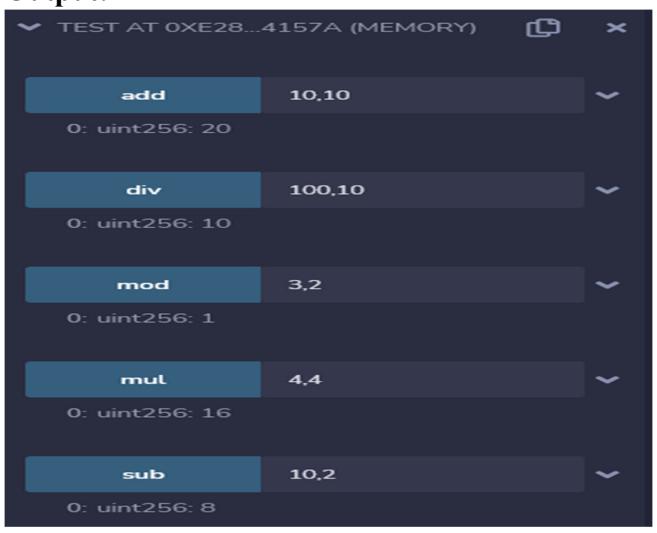


11) Write a solidity smart contract to create arithmetic calculator which includes functions for operations addition, subtraction, multiplication, division etc.

Code:

```
pragma solidity 0.5.0;
contract Test
{
   function add(uint num,uint num2) public view returns(uint)
   {
      num = num + num2;
      return num;
   }
   function sub(uint num,uint num2) public view returns(uint)
   {
      num = num - num2;
      return num;
   }
   function mul(uint num,uint num2) public view returns(uint)
   {
      num = num * num2;
      return num;
   }
   function div(uint num,uint num2) public view returns(uint)
   {
      num = num / num2;
      return num;
   }
   function mod(uint num,uint num2) public view returns(uint)
   {
      num = num % num2;
      return num;  }
}
```

Output:

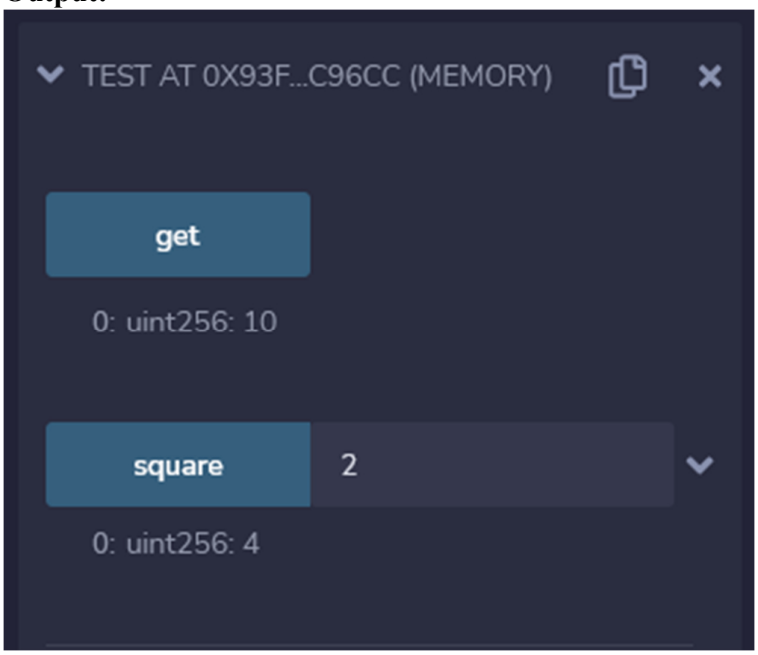


12) Write a solidity smart contract to demonstrate view function and pure function.

Code:

```
pragma solidity 0.5.0;
contract Test
{
   uint a=10;
   function get() public view returns(uint){
   return a;
   }
   function square(uint num) public pure returns(uint)
   {
      num = num * num;
      return num;
   }
}
```

Output:

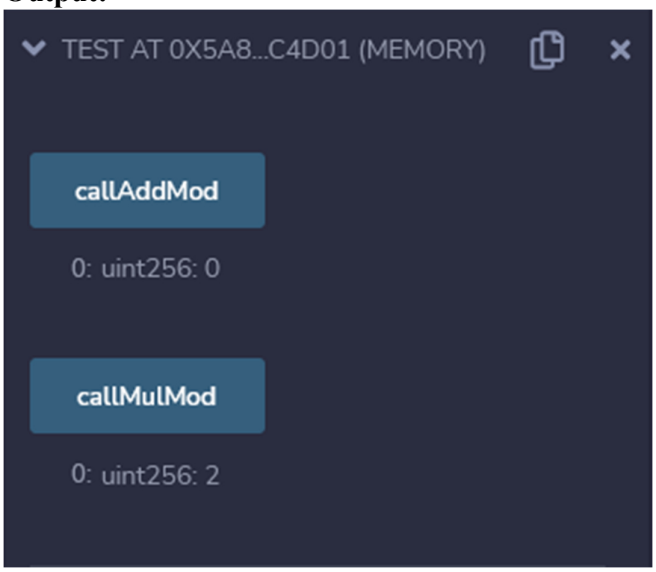


13) Write a solidity smart contract to demonstrate inbuilt mathematical functions.

Code:

```
pragma solidity ^0.5.0;

contract Test {
  function callAddMod() public pure returns(uint){
    return addmod(4, 5, 3);
  }
  function callMulMod() public pure returns(uint){
    return mulmod(4, 5, 3);
  }
}
```

Output:

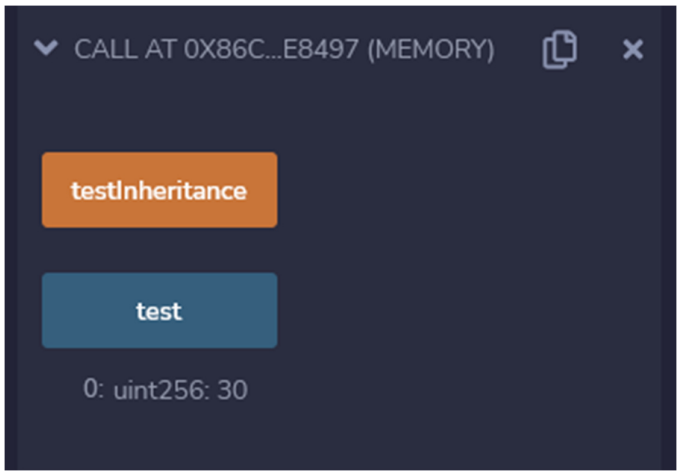


14) Write a solidity smart contract to demonstrate inheritance in contract.

Code:

```
pragma solidity 0.6.0;
contract parent{
        uint internal sum;
        function setValue() external {
            uint a = 10;
            uint b = 20;
            sum = a + b;
        }
}
contract child is parent{
        function getValue() external view returns(uint) {
            return sum;
        }
}
contract Call {

        child cc = new child();
        function testInheritance() public {
            cc.setValue();
        }
   function test() public view returns(uint)
   {
      return cc.getValue();
   }
}
```

Output:

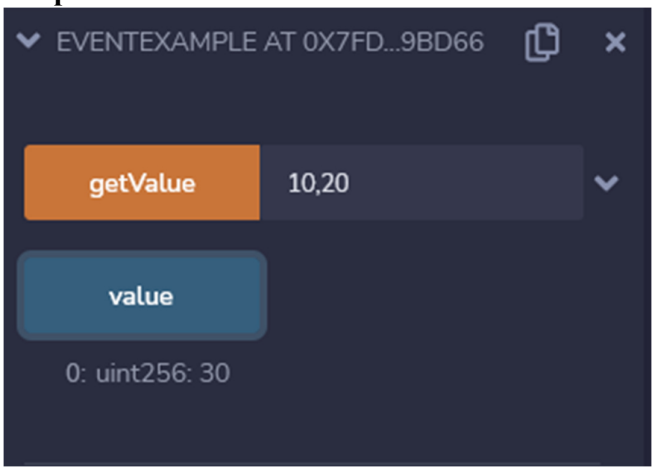


15) Write a solidity smart contract to demonstrate events.

Code:

```
pragma solidity ^0.4.21;

contract eventExample {

    uint256 public value = 0;

    event Increment(address owner);

    function getValue(uint _a, uint _b) public {
        emit Increment(msg.sender);
        value = _a + _b;
    }
}
```

Output:

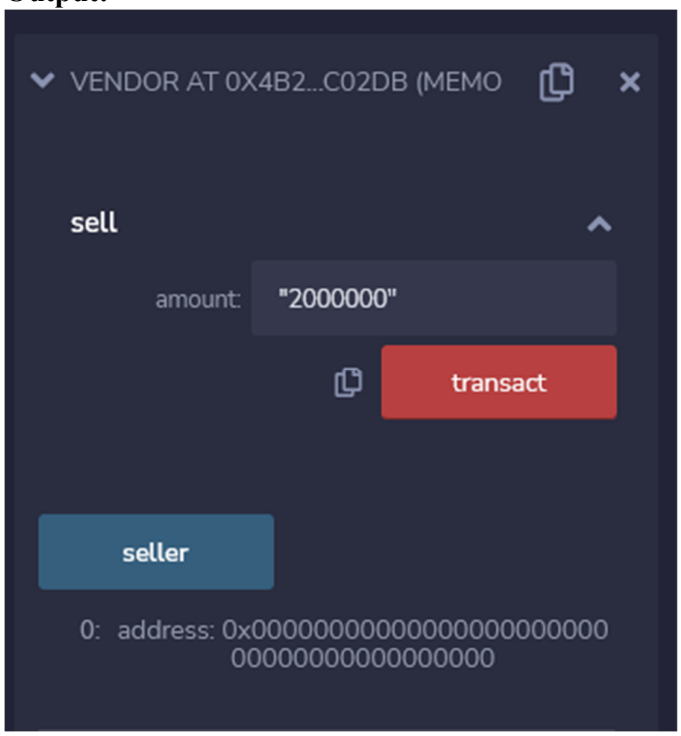


16) Write a solidity smart contract to demonstrate error handling.

Code:

```
pragma solidity ^0.5.0;

contract Vendor {
  address public seller;
  modifier onlySeller() {
    require(
      msg.sender == seller,
      "Only seller can call this."
    );
    _;
  }
  function sell(uint amount) public payable onlySeller {
    if (amount > msg.value / 2 ether)
      revert("Not enough Ether provided.");
  }
}
```

Output:



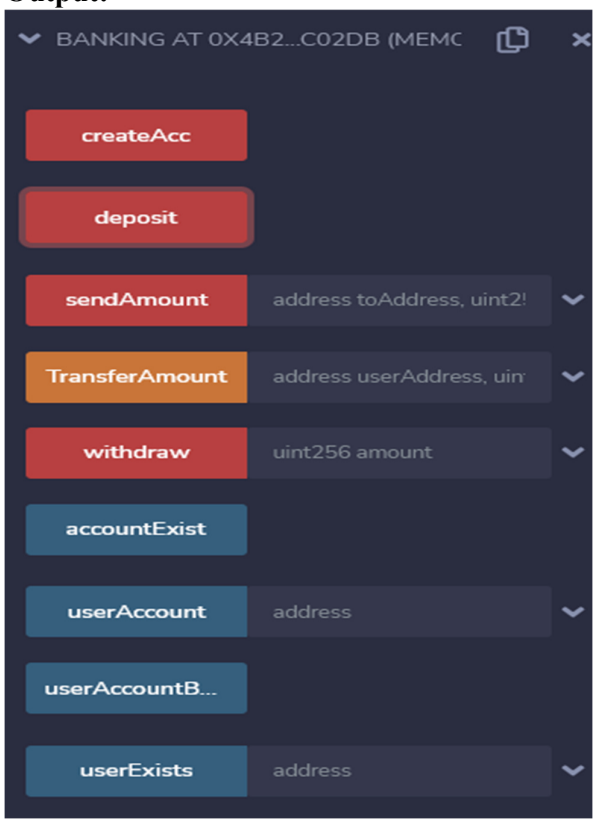
17) Write a solidity smart contract for Bank Account which provides operations such as check account balance, withdraw amount and deposit amount etc.

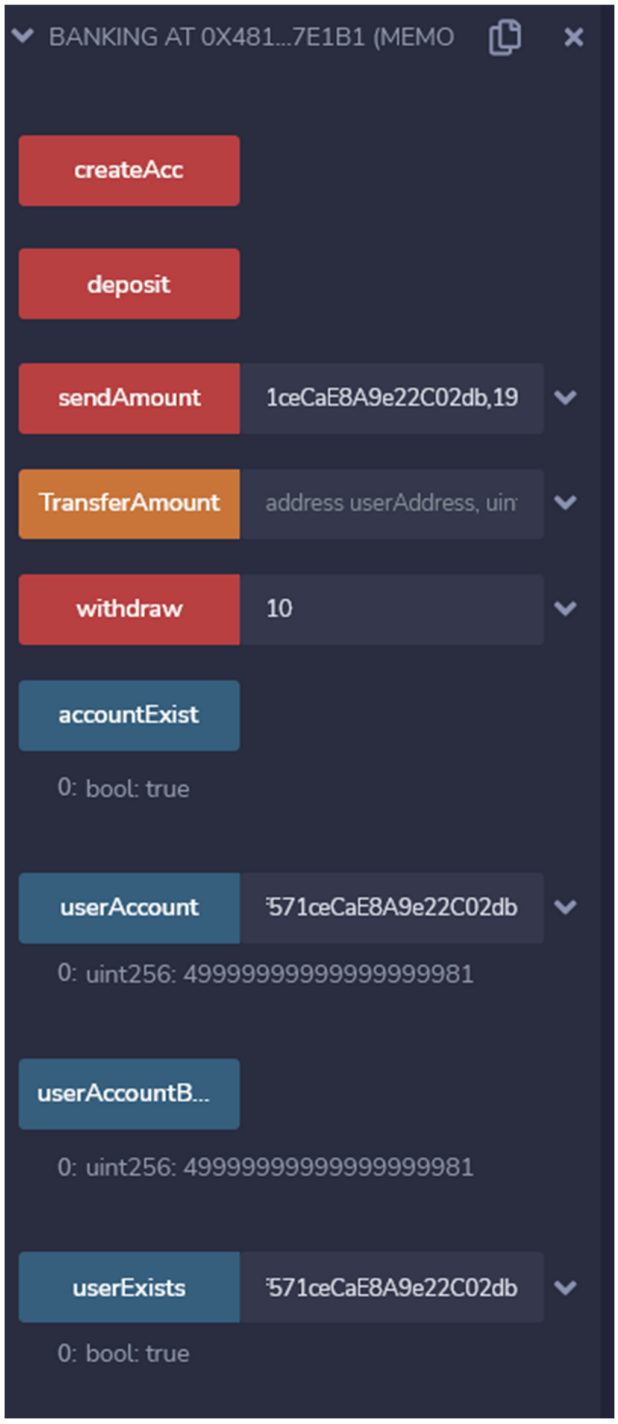
Code:

```
pragma solidity  >=0.4.22 <0.7.0;

contract banking{
  mapping(address=>uint) public userAccount;
  mapping(address=>bool) public userExists;

  function createAcc() public payable returns(string memory){
     require(userExists[msg.sender]==false,'Account Already Created');
     if(msg.value==0){
        userAccount[msg.sender]=0;
        userExists[msg.sender]=true;
        return 'account created';
     }
     require(userExists[msg.sender]==false,'account already created');
     userAccount[msg.sender] = msg.value;
     userExists[msg.sender] = true;
     return 'account created';
  }

  function deposit() public payable returns(string memory){
     require(userExists[msg.sender]==true, 'Account is not created');
     require(msg.value>0, 'Value for deposit is Zero');
     userAccount[msg.sender]=userAccount[msg.sender]+msg.value;
     return 'Deposited Succesfully';
  }

  function withdraw(uint amount) public payable returns(string memory){
     require(userAccount[msg.sender]>amount, 'insufficeint balance in Bank account');
     require(userExists[msg.sender]==true, 'Account is not created');
     require(amount>0, 'Enter non-zero value for withdrawal');
     userAccount[msg.sender]=userAccount[msg.sender]-amount;
     msg.sender.transfer(amount);
     return 'withdrawal Succesful';
  }

  function TransferAmount(address payable userAddress, uint amount) public
returns(string memory){
     require(userAccount[msg.sender]>amount, 'insufficeint balance in Bank account');
     require(userExists[msg.sender]==true, 'Account is not created');
     require(userExists[userAddress]==true, 'to Transfer account does not exists in bank
accounts ');
     require(amount>0, 'Enter non-zero value for sending');
```

```
    userAccount[msg.sender]=userAccount[msg.sender]-amount;
    userAccount[userAddress]=userAccount[userAddress]+amount;
    return 'transfer succesfully';
  }

  function sendAmount(address payable toAddress , uint256 amount) public payable
returns(string memory){
    require(amount>0, 'Enter non-zero value for withdrawal');
    require(userExists[msg.sender]==true, 'Account is not created');
    require(userAccount[msg.sender]>amount, 'insufficeint balance in Bank account');
    userAccount[msg.sender]=userAccount[msg.sender]-amount;
    toAddress.transfer(amount);
    return 'transfer success';
  }

  function userAccountBalance() public view returns(uint){
    return userAccount[msg.sender];
  }

  function accountExist() public view returns(bool){
    return userExists[msg.sender];
  }
}
```

Output:




BANKING AT 0X481...7E1B1 (MEMO
createAcc
deposit
sendAmount
1ceCaE8A9e22C02db,19
TransferAmount
address userAddress, uin
withdraw
10
accountExist
0: bool: true
userAccount
571ceCaE8A9e22C02db
0: uint256: 49999999999999999981
userAccountB...
0: uint256: 49999999999999999981
userExists
571ceCaE8A9e22C02db
0: bool: true

	HOPE Foundation's Finolex Academy Of Management And Technology, Ratnagiri Department of MCA		
Course Code: **MCALE331**	Course Name: **Blockchain Lab**		
Class: **SYMCA**	Semester: **III CBCGS (2 Year MCA)**		Academic Year: **2021-22**
Roll No: **22**	Name of Student: **Shrinath Sunil Padave**		
Assignment/Experiment No: **04**		Quiz Score:	5
Name of Assignment/Experiment: **Ethereum**			
Lab Objective applicable: **LOB4**. Understand the deployment of DApp in Ethereum.			
Lab Outcome applicable: **LO4**. Develop and deploy Dapp in Ethereum.			

Details of Quiz:

Q. No.	Question	Correct Answer	Given Answer
1	Which of the following development environment is/are used to connect Remix IDE to Metamask?	B. Injected Provider	B. Injected Provider
2	What's MetaMask used for exactly?	A. A Browser plugin that generates and stores private keys	D. All of the above
3	Which among the following is not true with respect to Contract Accounts characteristics?	D. Controlled by humans	D. Controlled by humans
4	Ethereum is _____.	B. open source blockchain networks	B. open source blockchain networks
5	Which among the feature does Ethereum ensures to provide?	D. All of the above	D. All of the above
6	Accounts in Ethereum are represented by _____.	A. hexadecimal address	A. hexadecimal address

Assignment/Experiment Evaluation:

Sr. No.	**Parameters**	**Marks Obtained**	**Out of**
1	Technical Understanding (Assessment based on Q and A through online quiz)	5	6
2	Neatness/Presentation	2	2
3	Punctuality	2	2
Date of Execution (DOE)	30/11/2021 Total Marks Obtained	9	10
Date of Assessment (DOA)	13/12/2021 Signature of Teacher		

PAGE NO.:
DATE: / /

practical No. 04.

Ethereum

Theory:

- Ethereum:
Ethereum is smart decentralize, open-source blockchain with smart contract functionality. Smart contract is a executable code that run on ethereum blockchain. Smart contracts are able to execute the contracts without third party. Ether is a cryptocurrency that are used for transactions. Amongest cryptocurrencies, Ether is second only to Bitcoin in market capitalization. Ethereum was invented 2013 by programmer Vitalik buterin.

- Ethereum virtual machine.
Ethereum virtual machine also known as EVM. EVM is the runtime environment for Ethereum virtual smart contract in Ethereum. The Ethereum virtual machine focuses on providing security and executing untrusted code by computer all over the worlds.

- Smart contracts:
A smart contracts is computer protocol intended to digitally facilitate verify or enforce the negotiation or

or performances of contract. Smart contracts allow performances of the contracts transactions that are creditable without any third parties. These transactions, are trackable and irreversible.

- **metamask:**
metamask is a software cryptocurrency wallet used to interact with the ethereum blockchain. It allows users to access their Ethereum wallet through a browser extention or mobile app, which can be used to computer with mobile, or browser Extention. metamask used to interact with dApps (Dapps). Metamask allow user to store and manage account keys broadcast transactions, send & receive Ethereum based crypto currency and tokens and securly connecting to decentralized applications through a compatible web browser or mobile app user.

- **Solidity:**
Solidity is object oriented, high level programming language for writing the smart contracts. Smart contracts are executible programs that run on the ethereum blockchain. Solidity program that run on Ethereum blockchain

PAGE NO.:
DATE: / /

- Geth: (Go Ethereum)
Geth is a command line interface for running Ethereum node implemented in Go language. Using geth you can join Ethereum network, Transfer ethers from one account to another and even mine ethers.

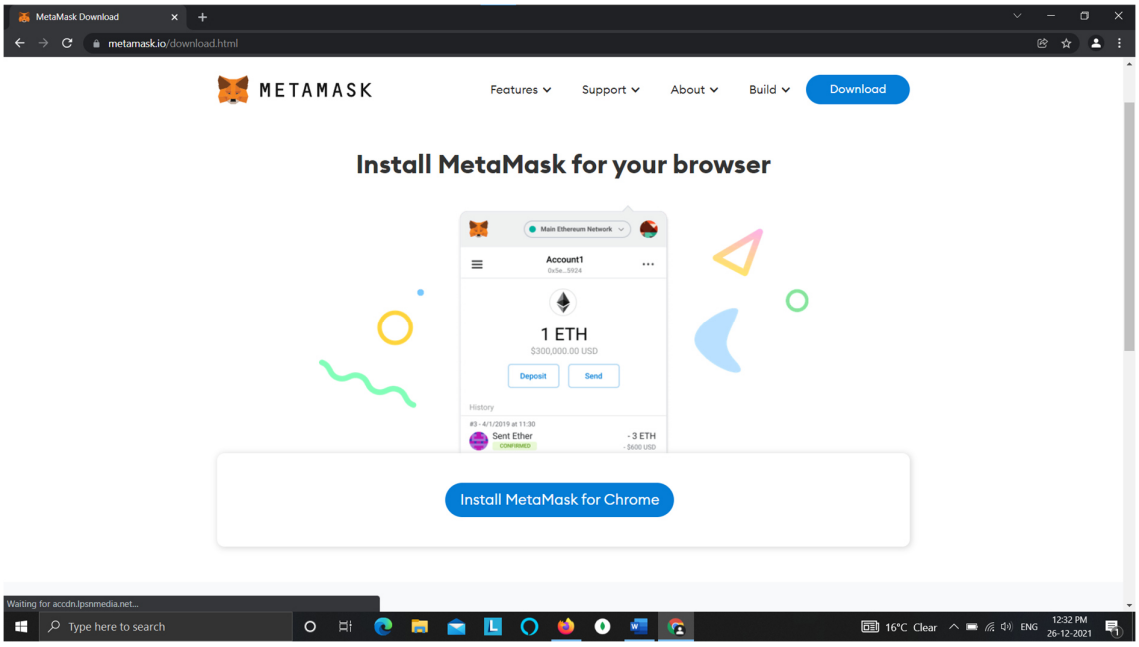
- Ganache:
Ganache is personal for rapid Ethereum and corda distributed application development. You can use ganache in entire development cycle: deploy your dApps and test your dApps in a safe and deterministic environment. Ganache comes in two flavors, UI & CLI.

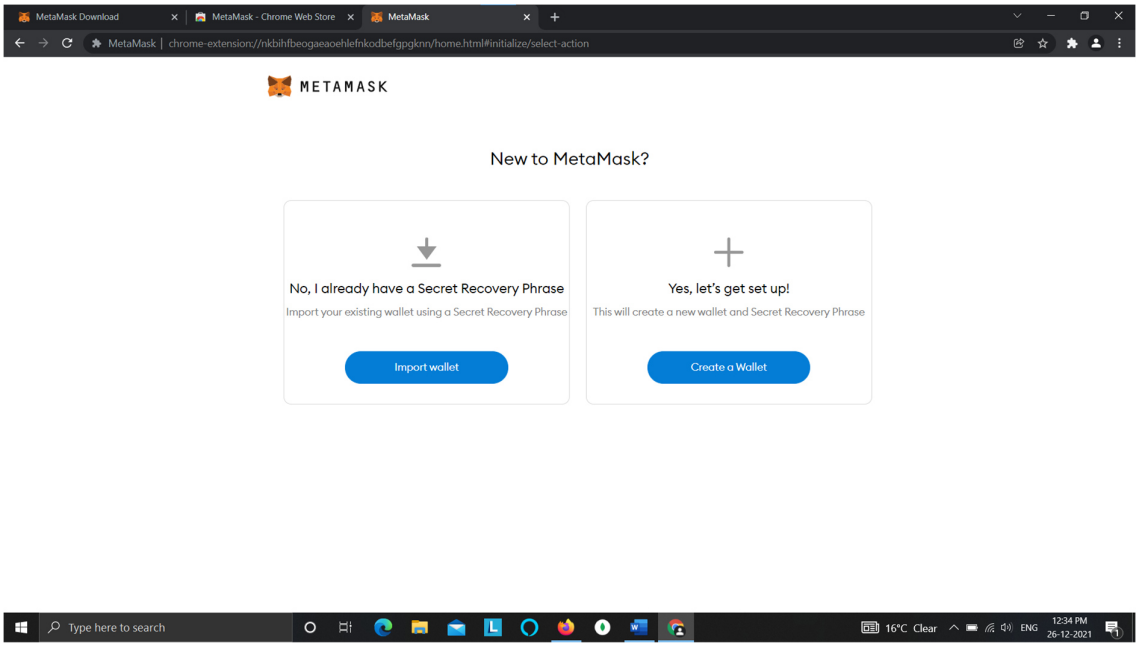
- Truffle:
Truffle is world-class development environment testing framework & asset pipeline for blockchains using the Ethereum-Virtual-machine, aming to makes blockchain develop life easier. Truffle is widly use most popular blockchain application development tool with over 1.5 million lifetime downloads.

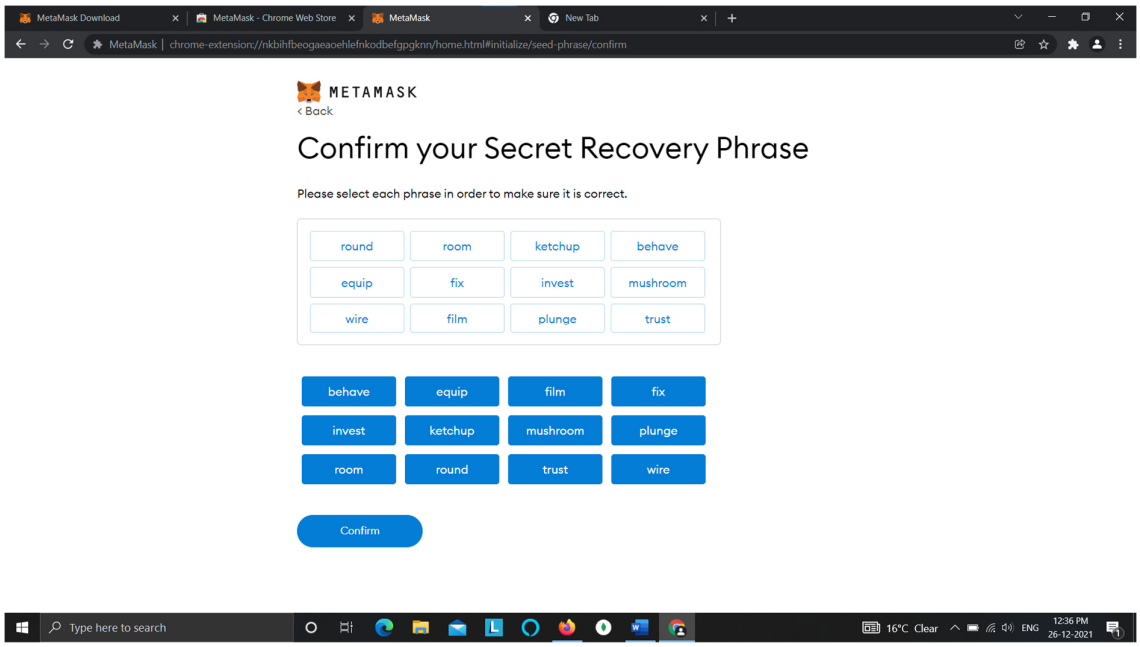
Practical No. 04

1) **Install the metamask in browser. Setup the metamask digital cryptocurrency wallet. Create multiple accounts in metamask and connect with one of the etherum blockchain test network. Perform the task buy ethers and send ethers from one account to another. Take the screenshots of created accounts, account assets and account transactions which showing the details of transaction.**

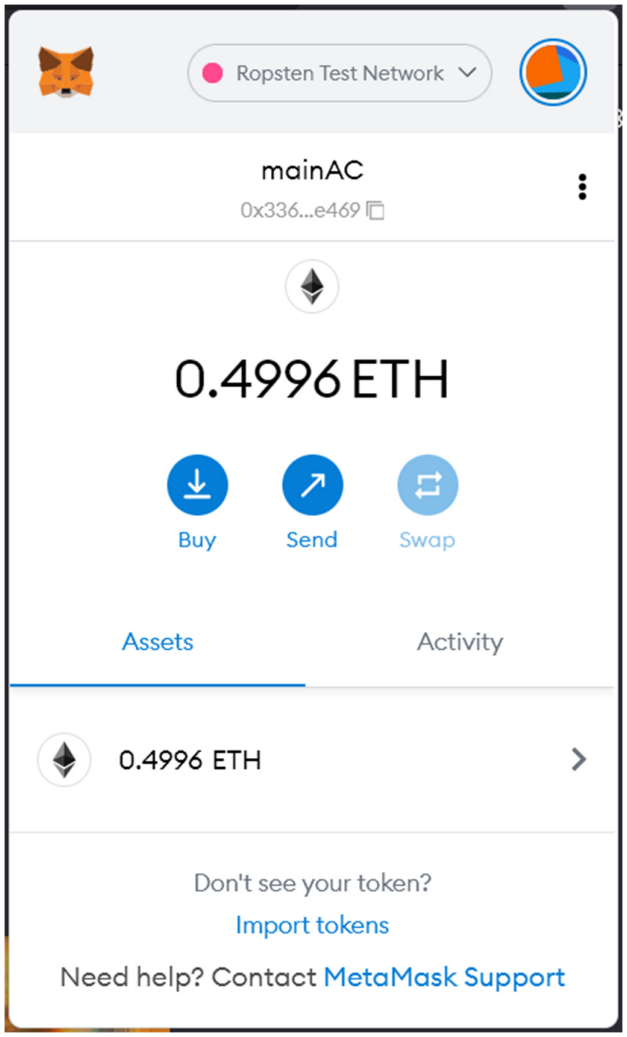
Outputs:

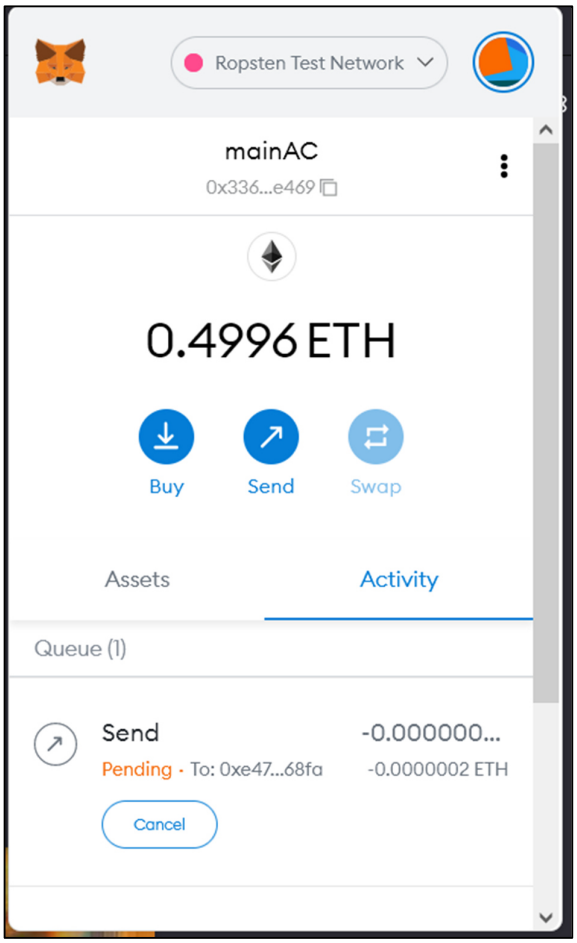




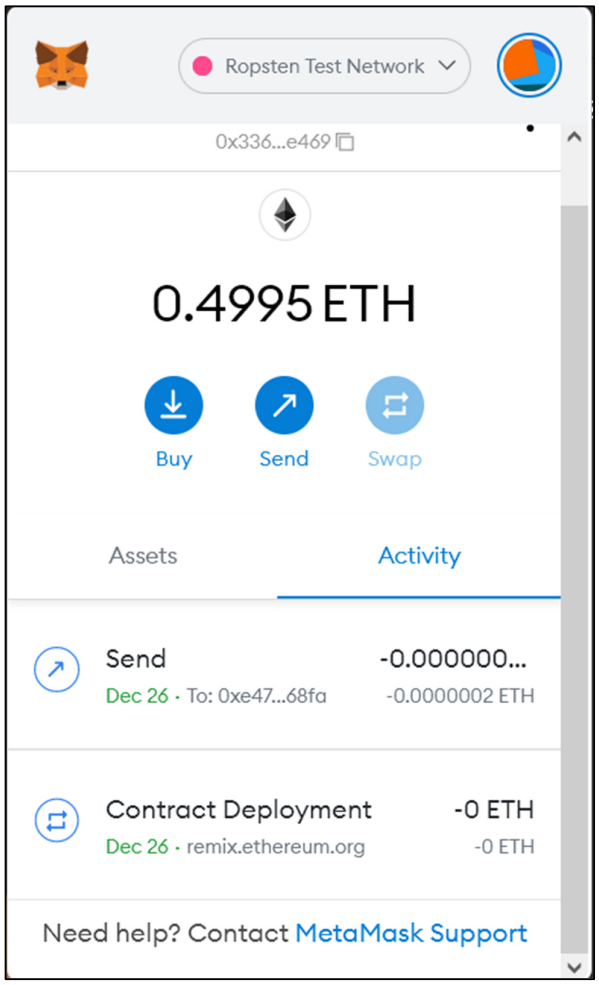

MetaMask Download
MetaMask - Chrome Web Store
MetaMask
New Tab
MetaMask | chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#initialize/seed-phrase/confirm
METAMASK
< Back
Confirm your Secret Recovery Phrase
Please select each phrase in order to make sure it is correct.
round
room
ketchup
behave
equip
fix
invest
mushroom
wire
film
plunge
trust
behave
equip
film
fix
invest
ketchup
mushroom
plunge
room
round
trust
wire
Confirm

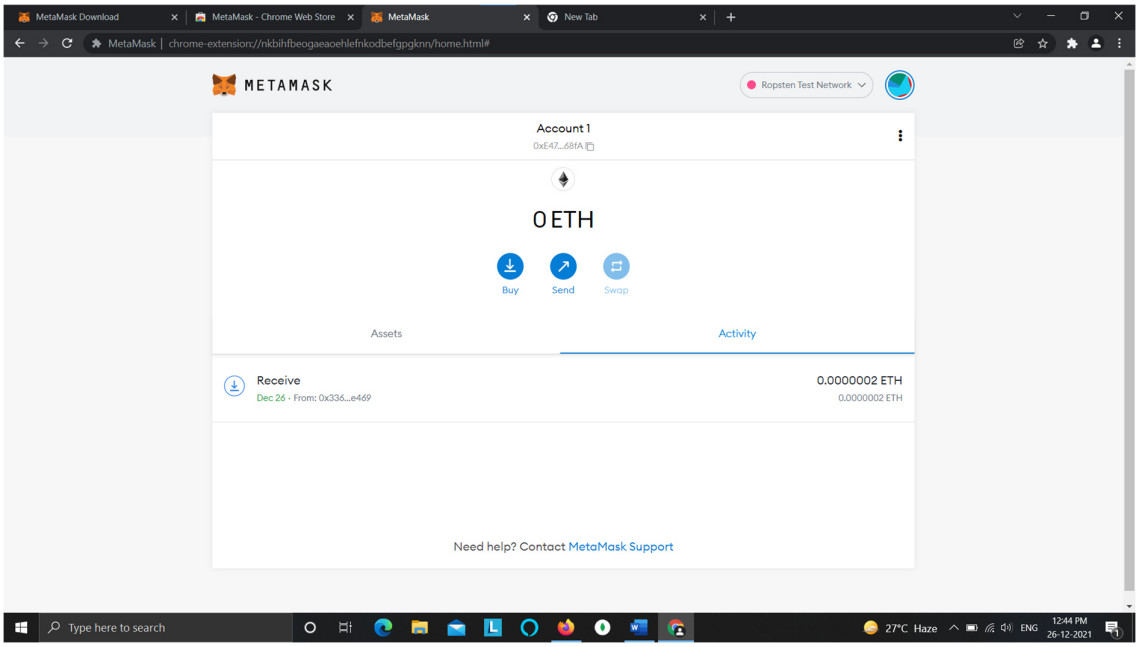
Type here to search
16°C Clear
ENG
12:36 PM
26-12-2021


Ropsten Test Network
mainAC
0x336...e469
0.4996 ETH
Buy
Send
Swap
Assets
Activity
0.4996 ETH
Don't see your token?
Import tokens
Need help? Contact MetaMask Support


Ropsten Test Network
mainAC
0x336...e469
0.4996 ETH
Buy
Send
Swap
Assets
Activity
Queue (1)
Send
-0.000000...
Pending · To: 0xe47...68fa
-0.0000002 ETH
Cancel


Confirmed transaction
Transaction 1 confirmed! View on Etherscan


Ropsten Test Network
0x336...e469
0.4995 ETH
Buy
Send
Swap
Assets
Activity
Send
-0.000000...
Dec 26 · To: 0xe47...68fa
-0.0000002 ETH
Contract Deployment
-0 ETH
Dec 26 · remix.ethereum.org
-0 ETH
Need help? Contact MetaMask Support

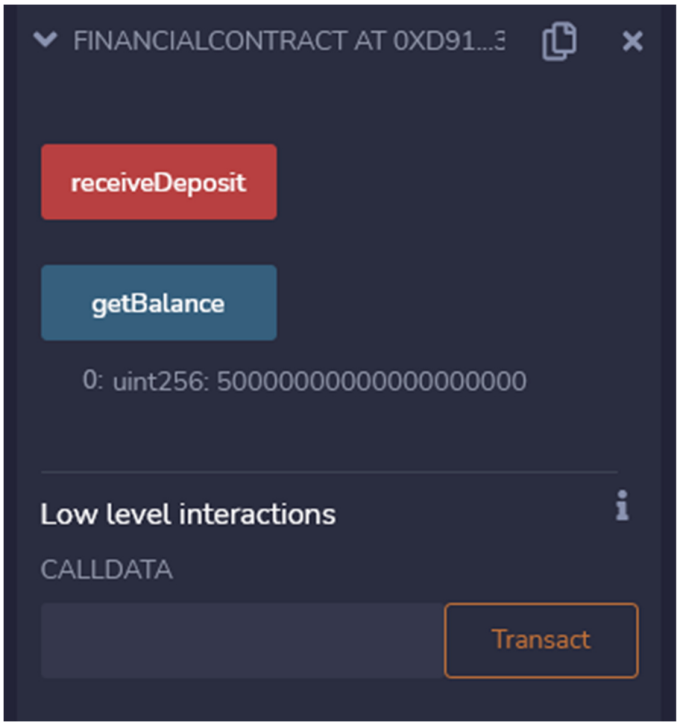

MetaMask Download
MetaMask - Chrome Web Store
MetaMask
New Tab
METAMASK
Ropsten Test Network
Account 1
0xE47...68fA
0 ETH
Buy
Send
Swap
Assets
Activity
Receive
Dec 26 · From: 0x336...e469
0.0000002 ETH
0.0000002 ETH
Need help? Contact MetaMask Support
Type here to search
27°C Haze
ENG
12:44 PM
26-12-2021

2) **Write a solidity smart contract to transfer funds (ethers) from user account to contract account using remixIDE and JavaScriptVM environment.**

Code:

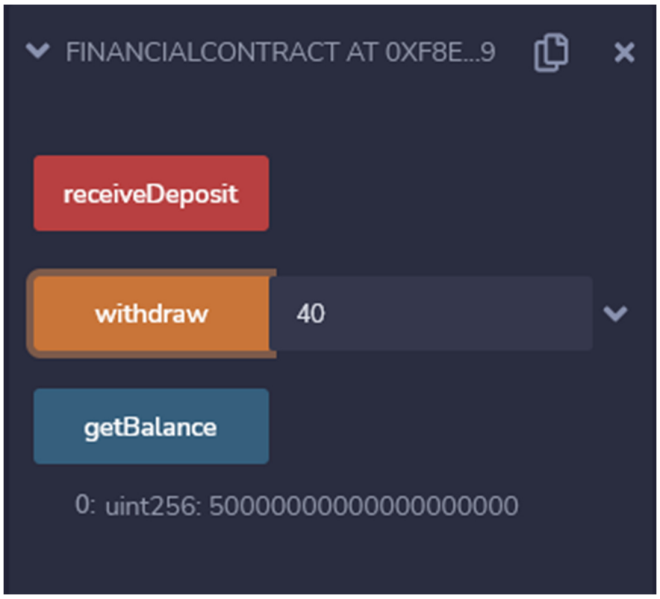
```
pragma solidity ^0.5.0;
contract financialContract{
        function receiveDeposit() payable public{

        }
        function getBalance() public view returns(uint){
           return address(this).balance;
        }
}
```

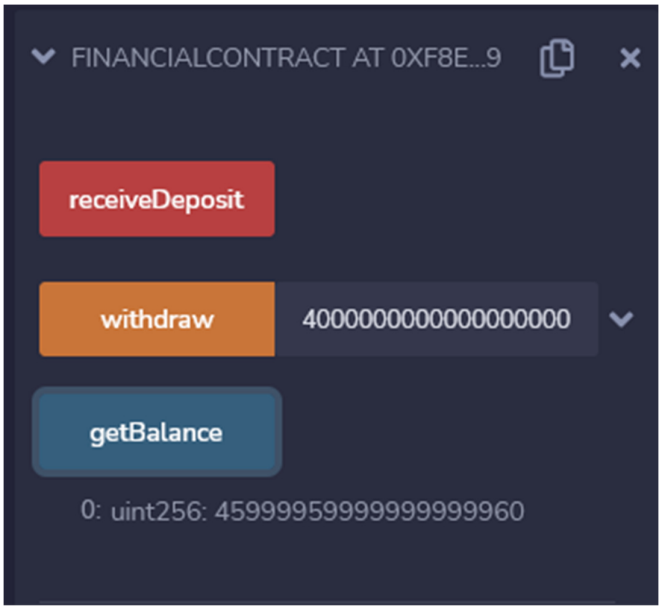
Output:



3) **Write a solidity smart contract to withdraw funds (ethers) from contract account to user account using remixIDE and JavaScriptVM environment.**

Code:





4) **Write a solidity smart contract to apply restriction that only owner of the contract can withdraw funds (ethers) from contract account to his/her user account using remixIDE and JavaScriptVM environment.**

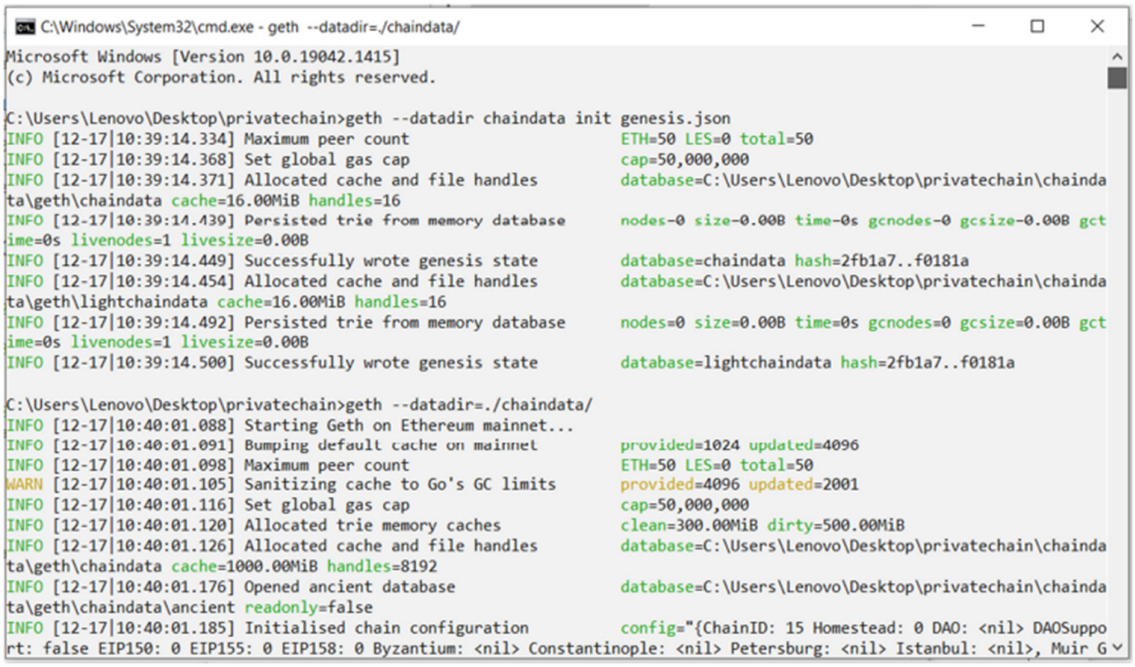
Code:

```
pragma solidity ^0.4.0;
contract financialContract{
    address owner;
    constructor() public{
    owner=msg.sender;
    }
    modifier ifOwner(){
      if(owner!=msg.sender){
        revert();
      }
      _;
    }
    function receiveDeposit() payable public{
    }
    function getBalance() public view returns(uint){
      return address(this).balance;
    }
    function withdraw(uint funds) public ifOwner{
      msg.sender.transfer(funds);
    }
}
```

Output:

5) **Create Ethereum node using Geth and create genesis block and create your personal private Ethereum blockchain. And using IPC is used to interact with Geth node to perform following task: create account, transfer funds using send transaction, mine the block, show the account balance before after the mining the block, show the specific block details and access chain details.**

Outputs:

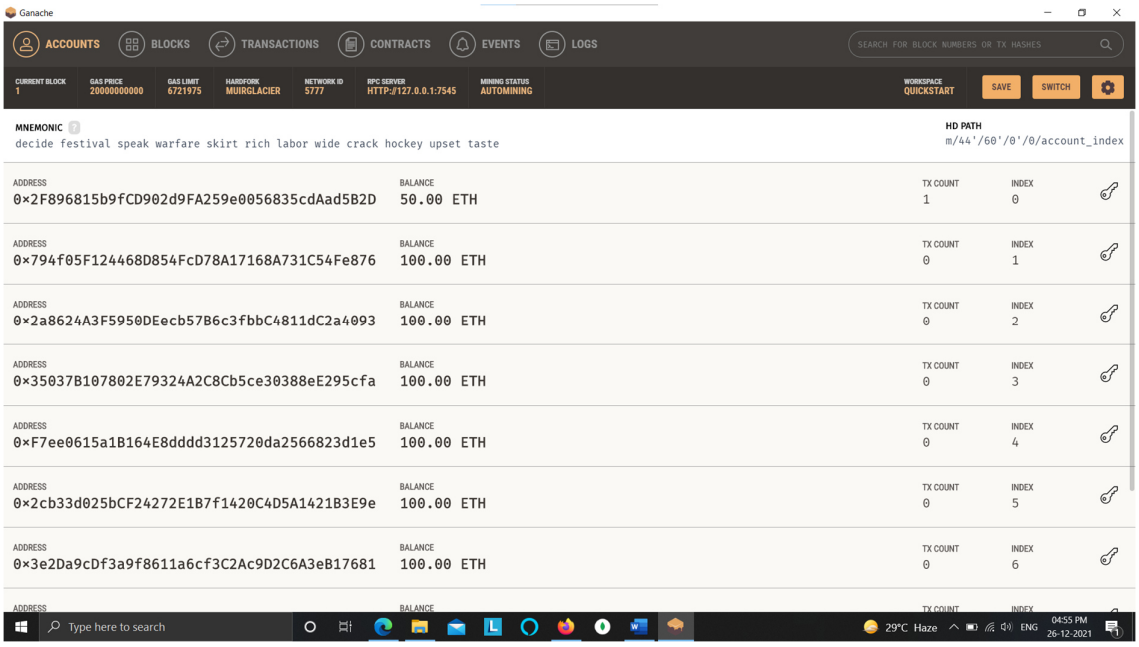


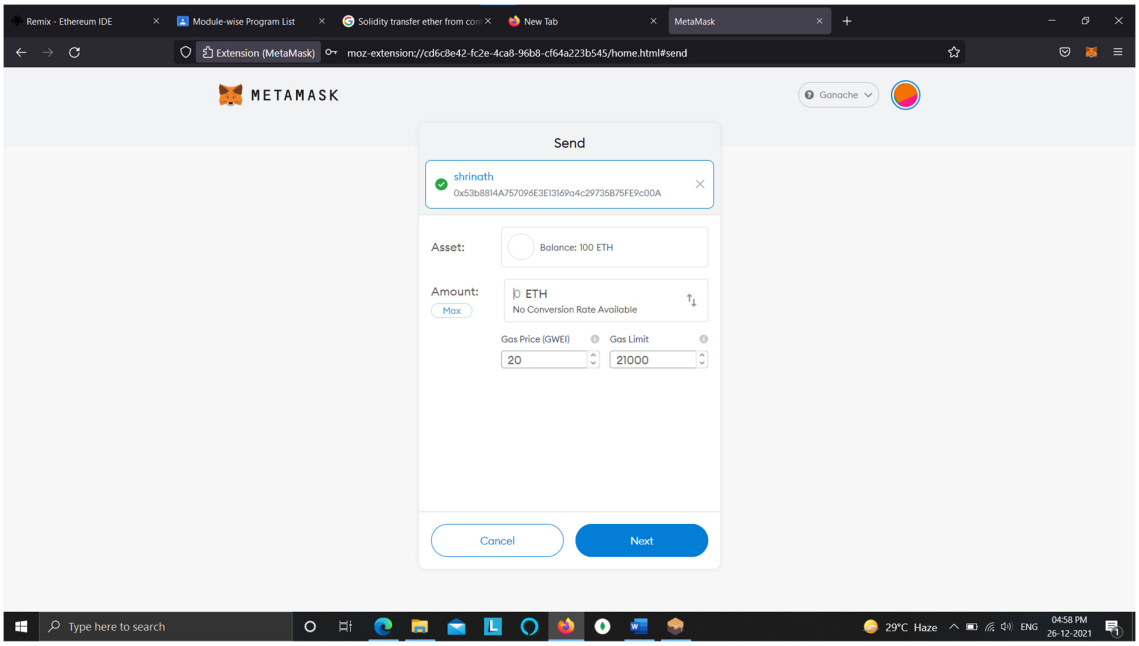

Command Prompt - geth attach ipc:\\.\pipe\geth.ipc
> eth.getBalance(eth.accounts[1])
1000000000000000000
> eth.getBlock("latest")
{
difficulty: 136916,
extraData: "0xda83010a0d846765746888676f312e31372e328777696e646f7773",
gasLimit: 3453424,
gasUsed: 0,
hash: "0x85bd27a7349dc7259bd47c8012a7189705ee4e29f177abd135cfa6fcdfa8f36d",
logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000",
miner: "0x1c11b49c2956222c6380c60ec0ec35a260237a4e",
mixHash: "0x92948cc65b85acfb2694dc6764083b7bee7a5145efccc14e3a93366e5d362473",
nonce: "0x3e50e09c8d67a28e",
number: 97,
parentHash: "0xd3f4e189dfa6ae5e4c610c79faa608bfdfea02e6a817147efb5b602ca9a65b63",
receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
size: 538,
stateRoot: "0x0e9b1d0a5b67863242c37f569e731e6a99c82969e29e1db38ca92faf950958e4",
timestamp: 1639718820,
totalDifficulty: 13109838,
transactions: [],
transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
uncles: []
}
>

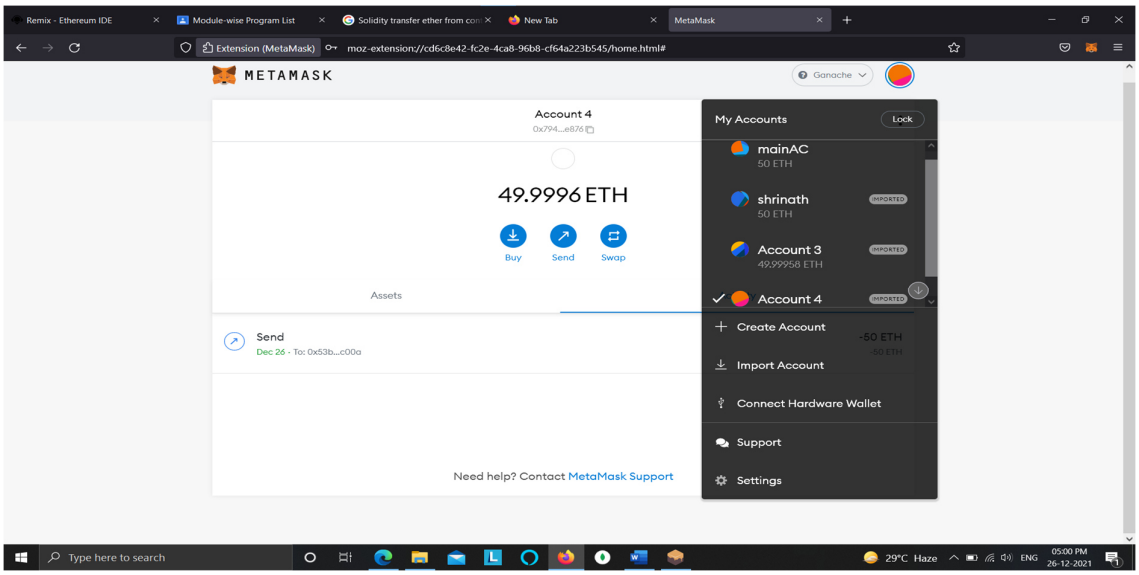

Command Prompt - geth attach ipc:\\.\pipe\geth.ipc
uncles: []
}
> eth.getBlock(30)
{
difficulty: 132544,
extraData: "0xda83010a0d846765746888676f312e31372e328777696e646f7773",
gasLimit: 3234899,
gasUsed: 0,
hash: "0x3b8ec437c7e3fd68590a128d44c2da70c889bd697cb45a7c9bf30a79d57ccd58",
logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000",
miner: "0x1c11b49c2956222c6380c60ec0ec35a260237a4e",
mixHash: "0x585968206c07fa6ae0fb3fd32354e5ba9d7515763efb131e0fe0dfee7e89efc1",
nonce: "0x39b7eb88afbf6493",
number: 30,
parentHash: "0x7179e0ecf47d45c93bccadf2e92c91064701622d7a663e330482251d374e8c52",
receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
size: 538,
stateRoot: "0x43aa4cf6474f32ec45ba3bb097084b558c1c3907dd1b3491957583cdf59960b5",
timestamp: 1639718753,
totalDifficulty: 4081536,
transactions: [],
transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
uncles: []
}
>

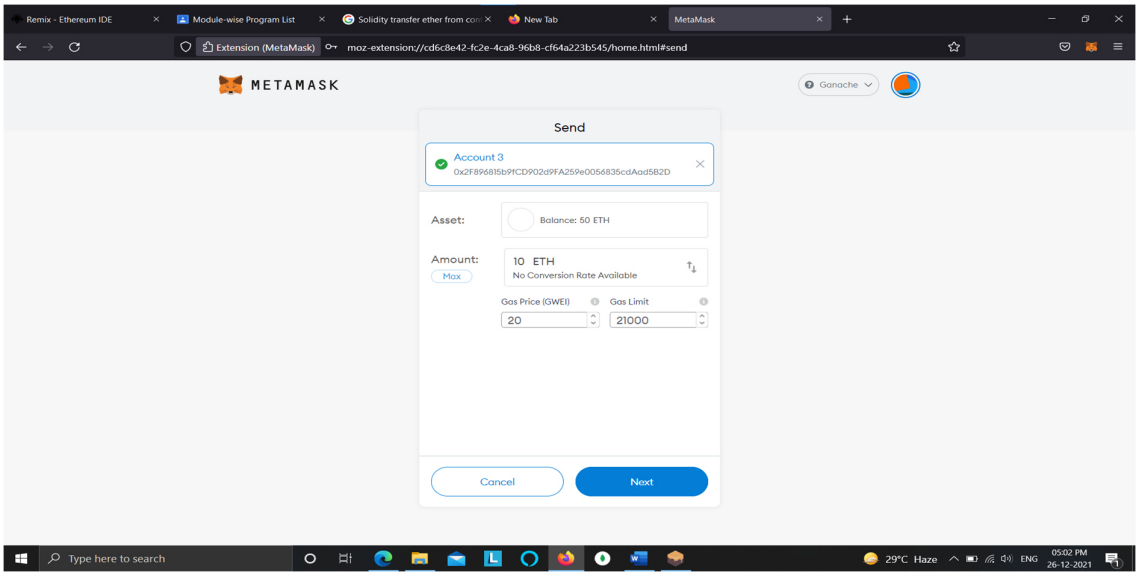
6) **Start Ganache your personal private blockchain network. Connect Ganache with MetaMask and import the account from Ganache to MetaMask. Transfer funds from imported account to other account of MetaMask. Take the screenshots of created accounts, account assets and account transactions which showing the details of transaction from MetaMask and Ganache interface.**

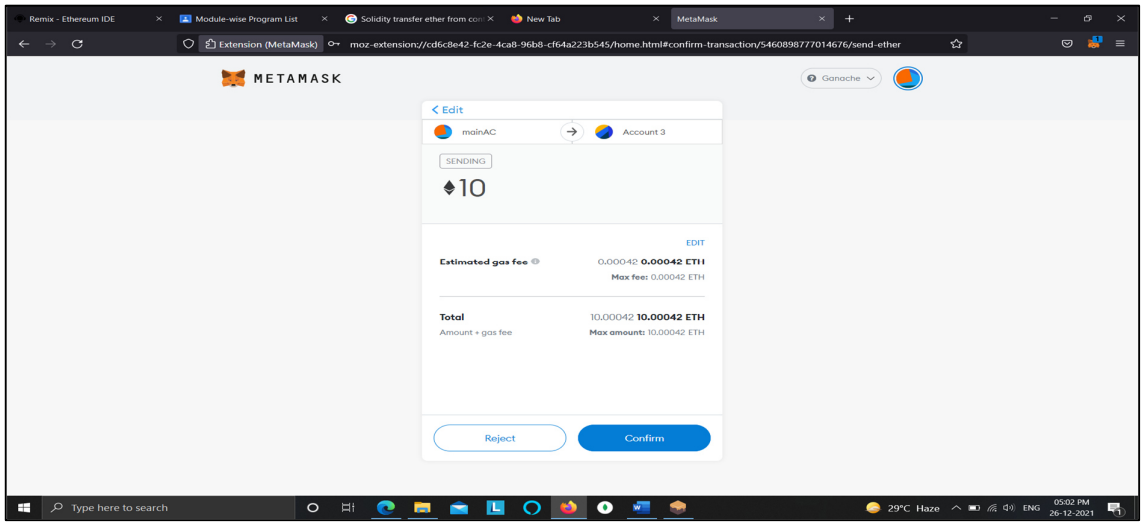
Screenshots:

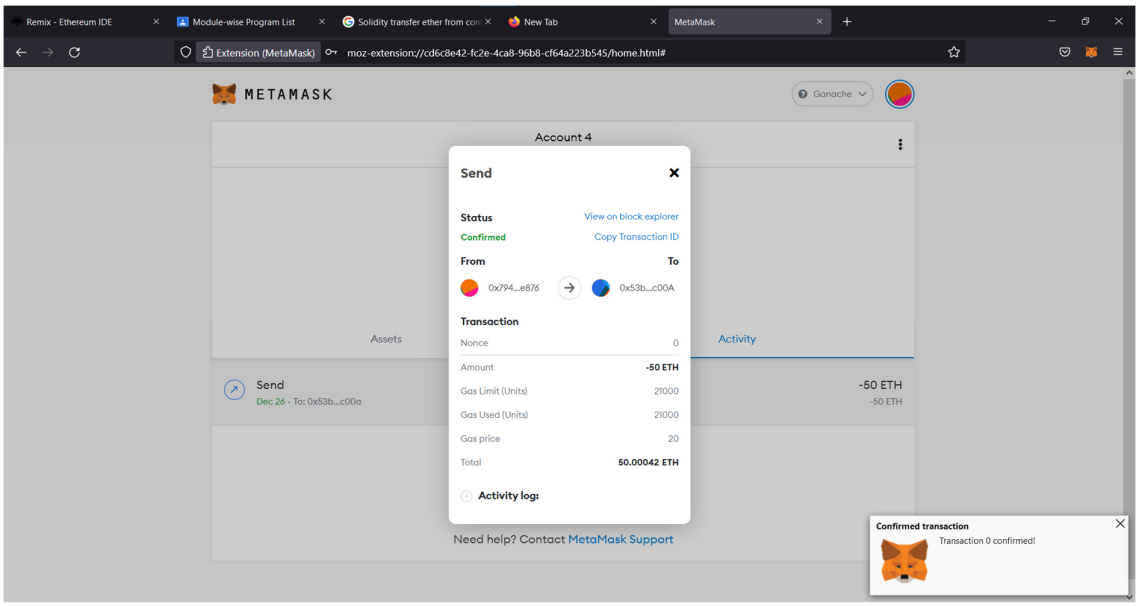



Remix - Ethereum IDE
Module-wise Program List
Solidity transfer ether from con
New Tab
MetaMask
Extension (MetaMask)
moz-extension://cd6c8e42-fc2e-4ca8-96b8-cf64a223b545/home.html#send
METAMASK
Ganache
Send
shrinath
0x53b8814A757096E3E13169a4c29735B75FE9c00A
Asset:
Balance: 100 ETH
Amount:
Max
0 ETH
No Conversion Rate Available
Gas Price (GWEI)
Gas Limit
20
21000
Cancel
Next
Type here to search
29°C Haze
ENG
04:58 PM
26-12-2021


Remix - Ethereum IDE
Module-wise Program List
Solidity transfer ether from con
New Tab
MetaMask
Extension (MetaMask)
moz-extension://cd6c8e42-fc2e-4ca8-96b8-cf64a223b545/home.html#
METAMASK
Ganache
Account 4
0x794...e876
49.9996 ETH
Buy
Send
Swap
Assets
Send
Dec 26 - To: 0x53b...c00a
Need help? Contact MetaMask Support
My Accounts
Lock
mainAC
50 ETH
shrinath
50 ETH
IMPORTED
Account 3
49.99958 ETH
IMPORTED
Account 4
IMPORTED
Create Account
Import Account
Connect Hardware Wallet
Support
Settings
-50 ETH
-50 ETH
Type here to search
29°C Haze
ENG
05:00 PM
26-12-2021

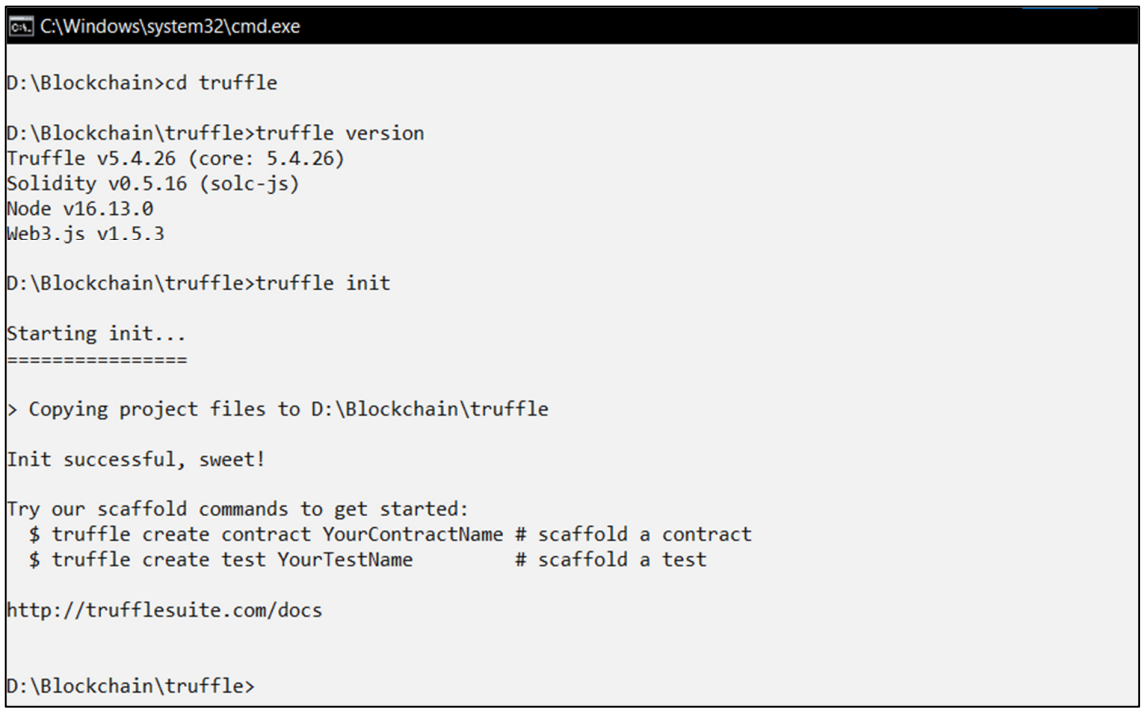

Remix - Ethereum IDE
Module-wise Program List
Solidity transfer ether from con
New Tab
MetaMask
Extension (MetaMask)
moz-extension://cd6c8e42-fc2e-4ca8-96b8-cf64a223b545/home.html#send
METAMASK
Ganache
Send
Account 3
0x2F896815b9fCD902d9FA259e0056835cdAad5B2D
Asset:
Balance: 50 ETH
Amount:
Max
10 ETH
No Conversion Rate Available
Gas Price (GWEI)
Gas Limit
20
21000
Cancel
Next
Type here to search
29°C Haze
ENG
05:02 PM
26-12-2021


Remix - Ethereum IDE
Module-wise Program List
Solidity transfer ether from con
New Tab
MetaMask
Extension (MetaMask)
moz-extension://cd6c8e42-fc2e-4ca8-96b8-cf64a223b545/home.html#confirm-transaction/546089777014676/send-ether
METAMASK
Ganache
Edit
mainAC
Account 3
SENDING
10
EDIT
Estimated gas fee
0.00042 ETH
Max fee: 0.00042 ETH
Total
10.00042 ETH
Amount + gas fee
Max amount: 10.00042 ETH
Reject
Confirm
Type here to search
29°C Haze
ENG
05:02 PM
26-12-2021


Remix - Ethereum IDE
Module-wise Program List
Solidity transfer ether from con
New Tab
MetaMask
Extension (MetaMask)
moz-extension://cd6c8e42-fc2e-4ca8-96b8-cf64a223b545/home.html#
METAMASK
Ganache
Account 4
Send
Status
Confirmed
View on block explorer
Copy Transaction ID
From
To
0x794...e876
0x53b...c00A
Transaction
Nonce
0
Amount
-50 ETH
Gas Limit (Units)
21000
Gas Used (Units)
21000
Gas price
20
Total
50.00042 ETH
Activity log:
Assets
Activity
Send
Dec 26 - To: 0x53b...c00a
-50 ETH
-50 ETH
Need help? Contact MetaMask Support
Confirmed transaction
Transaction 0 confirmed!

7) **Create new truffle project with migration script, smart contract and configuration file. First compile it using truffle suite. Then connect it with personal private blockchain i.e. Ganache and deploy (migrate) smart contract on deploy. Open truffle console and create instance of deployed (migrated) contract of Ganache. Then interact with smart contract using created instance. Take screenshots of all transaction's details and block details from Ganache.**

Outputs:



Code:

Demo.sol

```
//SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.7;
contract demo {
   uint num;
   function set(uint _num) public{
      num = _num;
   }
   function get() public view returns(uint){
      return num;
   }
}
```

Compiling contracts:

```
PS D:\Blockchain\truffle\web> truffle compile

==========================
√ Fetching solc version list from solc-bin. Attempt #1
√ Downloading compiler. Attempt #1.
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\demo.sol
> Compiling .\contracts\demo.sol
> Compilation warnings encountered:

    Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" t
h source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> project:/contracts/demo.sol


> Artifacts written to D:\Blockchain\truffle\build\contracts
> Compiled successfully using:
   - solc: 0.8.7+commit.e28d00a7.Emscripten.clang


Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\demo.sol
> Compiling .\contracts\demo.sol
> Compilation warnings encountered:

    Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" t
h source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> project:/contracts/demo.sol


> Artifacts written to D:\Blockchain\truffle\build\contracts
> Compiled successfully using:
   - solc: 0.8.7+commit.e28d00a7.Emscripten.clang
```

Migrating contracts:

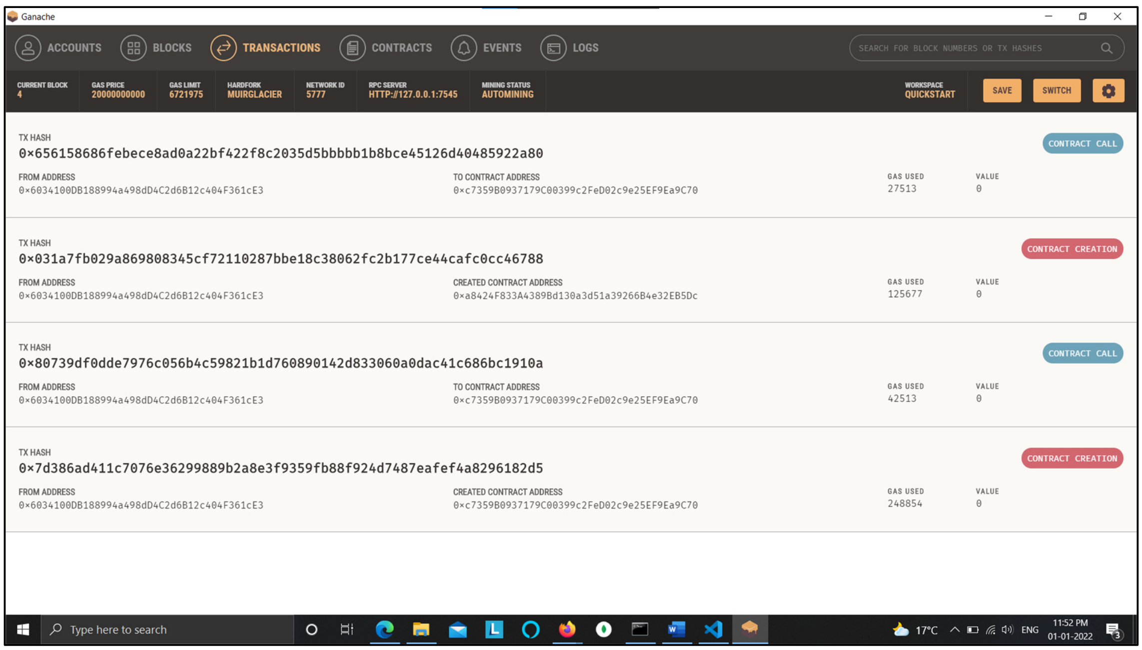
```
PS D:\Blockchain\truffle\web> truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\demo.sol
> Compilation warnings encountered:

    Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" t
h source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> project:/contracts/demo.sol


> Artifacts written to D:\Blockchain\truffle\build\contracts
> Compiled successfully using:
   - solc: 0.8.7+commit.e28d00a7.Emscripten.clang



Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)
```

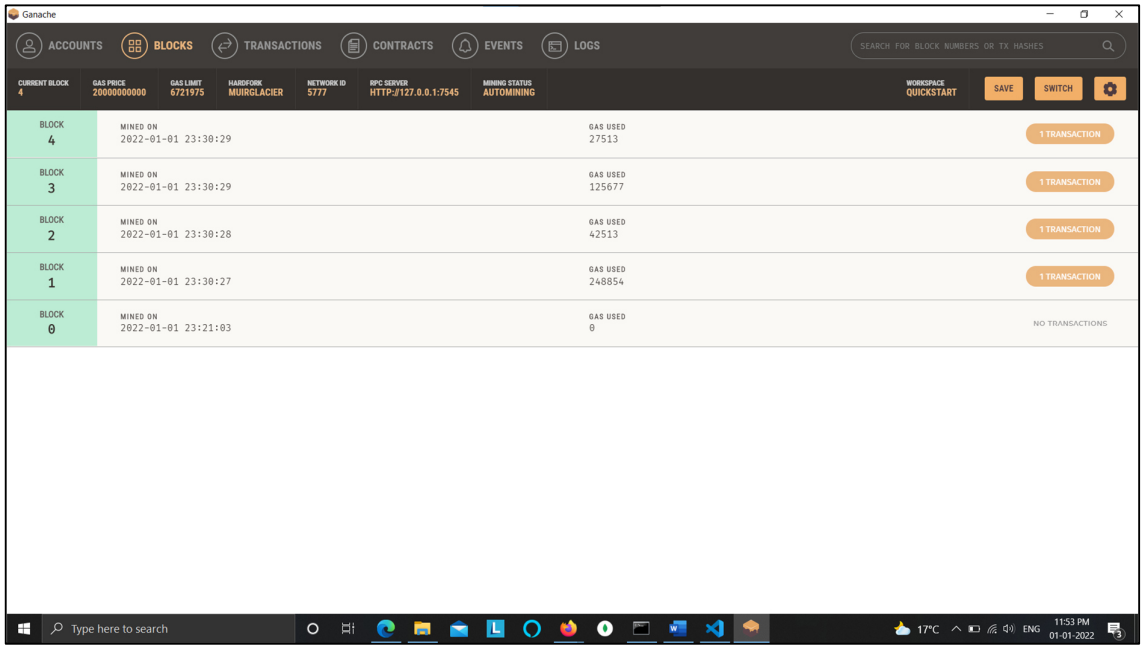
```
1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x7d386ad411c7076e36299889b2a8e3f9359fb88f924d7487eafef4a8296182d5
   > Blocks: 0            Seconds: 0
   > contract address:    0xc7359B0937179C00399c2FeD02c9e25EF9Ea9C70
   > block number:        1
   > block timestamp:     1641060027
   > account:             0x6034100DB188994a498dD4C2d6B12c404F361cE3
   > balance:             99.99502292
   > gas used:            248854 (0x3cc16)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00497708 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00497708 ETH


2_demo_migration.js
===================

   Deploying 'demo'
   ----------------
   > transaction hash:    0x031a7fb029a869808345cf72110287bbe18c38062fc2b177ce44cafc0cc46788
   > Blocks: 0            Seconds: 0
   > contract address:    0xa8424F833A4389Bd130a3d51a39266B4e32EB5Dc
   > block number:        3
   > block timestamp:     1641060029
   > account:             0x6034100DB188994a498dD4C2d6B12c404F361cE3
   > balance:             99.99165912
   > gas used:            125677 (0x1eaed)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00251354 ETH
```

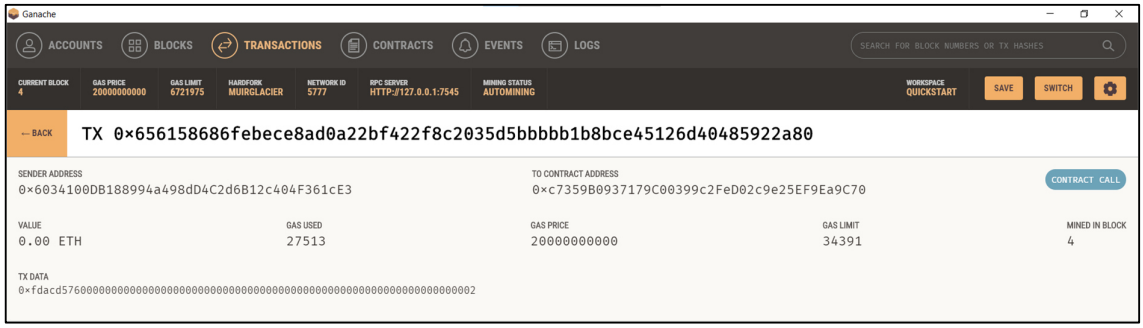
```
   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00251354 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.00749062 ETH
```

```
truffle(development)> demo.deployed().then(function(instance) {app=instance;})
undefined
truffle(development)> app.temp()
'hello'
truffle(development)> app.get()
BN { negative: 0, words: [ 0, <1 empty item> ], length: 1, red: null }
truffle(development)> app.set(10,from:accounts[0])
evalmachine.<anonymous>:0
```

```
truffle(development)> app.set(20,{from:accounts[0]})
{
  tx: '0xce363a73a4f07eb228607109aab78afbc28d7afb9785ed4518f581570ffe4d30',
  receipt: {
    transactionHash: '0xce363a73a4f07eb228607109aab78afbc28d7afb9785ed4518f581570ffe4d30',
    transactionIndex: 0,
    blockHash: '0x517018a37f6ba744b797fb760c8076ec5510c06ea02543b7ac1f524cdda93a97',
    blockNumber: 7,
    from: '0x8412153ab32cc784fc43ad5d64c1fe9c83bf9599',
    to: '0x87223e3e003a40d64d0254d0504dee175a8aa5d1',
    gasUsed: 41602,
    cumulativeGasUsed: 41602,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
truffle(development)> app.get()
BN { negative: 0, words: [ 20, <1 empty item> ], length: 1, red: null }
truffle(development)>
```




Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 4
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MUIRGLACIER
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE QUICKSTART
SAVE
SWITCH
BLOCK 4
MINED ON 2022-01-01 23:30:29
GAS USED 27513
1 TRANSACTION
BLOCK 3
MINED ON 2022-01-01 23:30:29
GAS USED 125677
1 TRANSACTION
BLOCK 2
MINED ON 2022-01-01 23:30:28
GAS USED 42513
1 TRANSACTION
BLOCK 1
MINED ON 2022-01-01 23:30:27
GAS USED 248854
1 TRANSACTION
BLOCK 0
MINED ON 2022-01-01 23:21:03
GAS USED 0
NO TRANSACTIONS
Type here to search
17°C
ENG
11:53 PM
01-01-2022

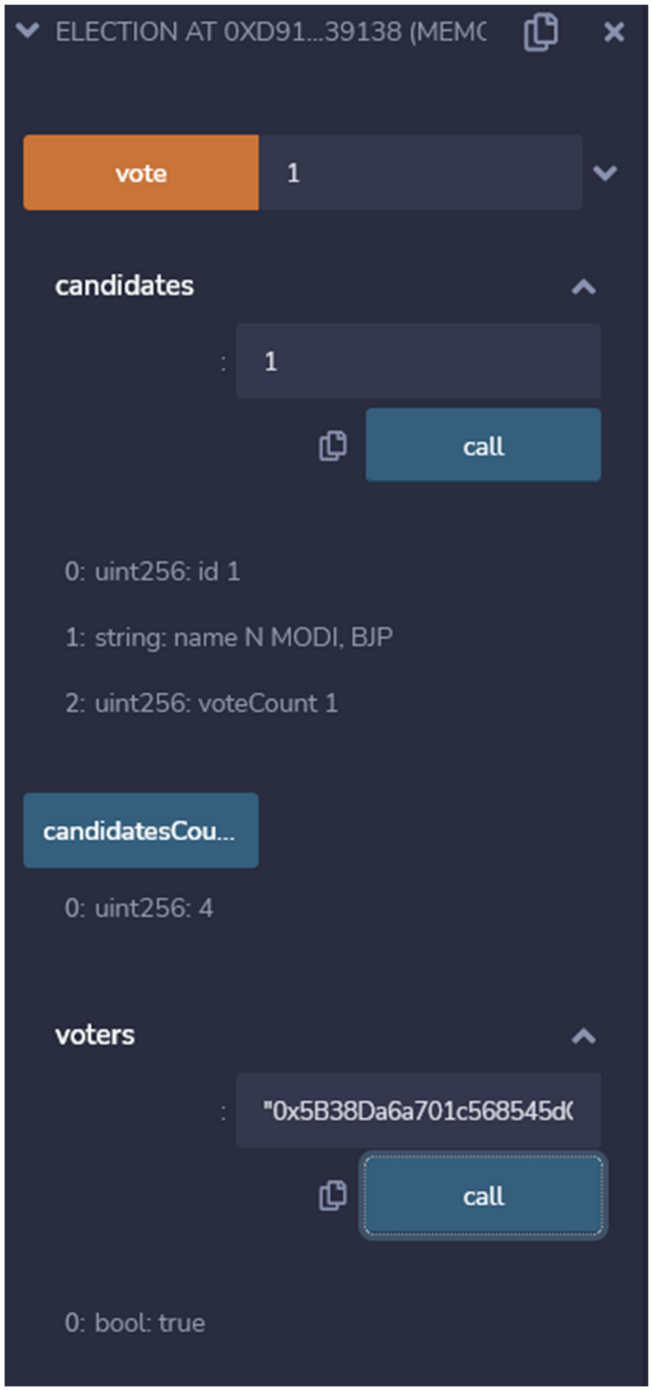

Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 4
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MUIRGLACIER
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE QUICKSTART
SAVE
SWITCH
← BACK
TX 0×656158686febece8ad0a22bf422f8c2035d5bbbbb1b8bce45126d40485922a80
SENDER ADDRESS
0×6034100DB188994a498dD4C2d6B12c404F361cE3
TO CONTRACT ADDRESS
0×c7359B0937179C00399c2FeD02c9e25EF9Ea9C70
CONTRACT CALL
VALUE
0.00 ETH
GAS USED
27513
GAS PRICE
20000000000
GAS LIMIT
34391
MINED IN BLOCK
4
TX DATA
0×fdacd576000000000000000000000000000000000000000000000000000000000000002

8) Create a election contract using solidty.

Code:

```
pragma solidity ^0.4.2;
contract Election {
struct Candidate {
uint id;
string name;
uint voteCount;
}
mapping(address => bool) public voters;
mapping(uint => Candidate) public candidates;
uint public candidatesCount;

event votedEvent (
uint indexed _candidateId
);
function Election () public {
addCandidate("N MODI, BJP");
addCandidate("A kejriwal, AAP");
addCandidate("Rahul G, Congress");
addCandidate("Nikhil, JDS");

}

function addCandidate (string _name) private {
candidatesCount ++;
candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}
function vote (uint _candidateId) public {
require(!voters[msg.sender]);
require(_candidateId > 0 && _candidateId <= candidatesCount);
voters[msg.sender] = true;
candidates[_candidateId].voteCount ++;
votedEvent(_candidateId);
}
}
```

Ouput:


ELECTION AT 0XD91...39138 (MEMC
vote
1
candidates
1
call
0: uint256: id 1
1: string: name N MODI, BJP
2: uint256: voteCount 1
candidatesCou...
0: uint256: 4
voters
"0x5B38Da6a701c568545d(
call
0: bool: true

Hyperledger case study

Challenge

When an outbreak of a food-borne disease happens, it can take days, if not weeks, to find its source. Better traceability could help save lives by allowing companies to act faster and protect the livelihoods of farmers by only discarding produce from the affected farms.

Approach

Walmart thought that blockchain technology might be a good fit for the decentralized food supply ecosystem. To test this hypothesis, the company created a food traceability system based on Hyperledger Fabric. Walmart, together with its technology partner IBM, ran two proof of concept projects to test the system. One project was about tracing mangos sold in Walmart's US stores and the other aimed to trace pork sold in its China stores. The Hyperledger Fabric blockchain-based food traceability system built for the two products worked. For pork in China, it allowed uploading certificates of authenticity to the blockchain, bringing more trust to a system where that used to be a serious issue. And for mangoes in the US, the time needed to trace their provenance went from 7 days to... 2.2 seconds!

Results

Walmart can now trace the origin of over 25 products from 5 different suppliers using a system powered by Hyperledger Fabric. The company plans to roll out the system to more products and categories in the near future. In fact, it has recently announced that it will start requiring all of its suppliers of fresh leafy greens (like salad and spinach) to trace their products using the system.

Tracking food for better safety

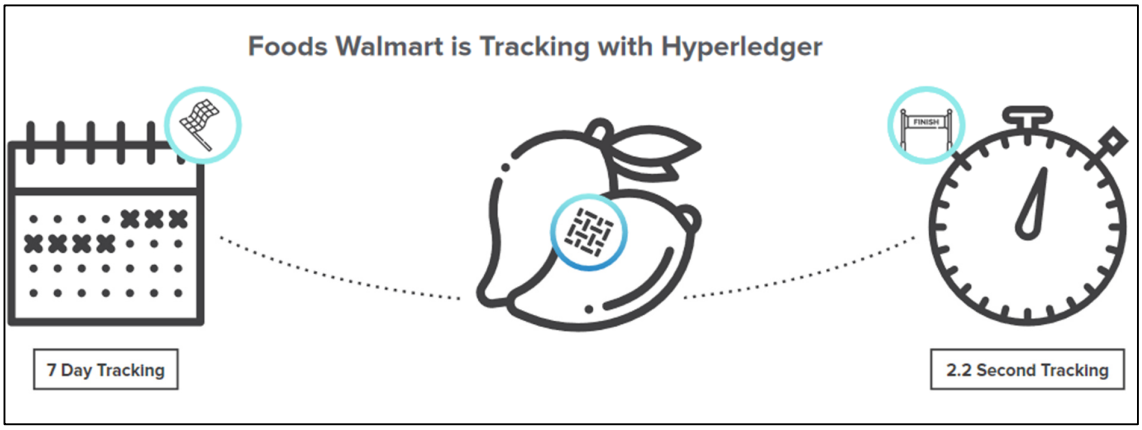
We rarely think about it, but the modern food system is a marvel. We have access to fresh produce all year round, buy exotic food from all around the world, and have more variety than our ancestors could ever dream of. Our food is generally safe to eat. Still, occasionally it can make us sick. Last year in 2018 there were at least 18 reported outbreaks of foodborne illnesses in the USA, including the E. coli found in romaine lettuce. “People talk about the food supply chain,” says Frank Yiannas, former Vice President of Food Safety at Walmart. “But it is not actually a chain, it’s complex network.” When an outbreak of a food-borne disease does happen, it can take days, if not weeks, to find its source. If investigators cannot point to a specific farm or farms, the government usually advises consumers to avoid products grown in a certain area (as happened with romaine lettuce from Yuma, Arizona), or even to avoid the type of product altogether. According to Walmart, millions of bags or heads of lettuce had to be removed, and consumers lost confidence in romaine lettuce altogether. Better traceability could help save lives by allowing companies to act faster and protect the livelihoods of farmers by only discarding produce from the affected farms. For this reason, Walmart has always been interested in enhancing transparency and traceability in the food system. Mr. Yiannas explains that the company has tried many systems and approaches to solving this problem over the years; none had brought them the kind of results they were after. When Yiannas first heard about blockchain and the idea of using it to trace food in the supply chain, he was skeptical.

From blockchain skeptic to believer

Karl Bedwell, Senior Director at Walmart Technology, explains, “Creating a (traceability) system for the entire food supply ecosystem has been a challenge for years, and no one had figured it out. We thought that blockchain technology might be a good fit for this problem, because of its focus on trust, immutability, and transparency.” Bedwell and his team introduced Yiannas to the possibilities of blockchain technology for enterprise solutions. Says Yiannas, “I really had an “aha” moment once I deeply understood the technology. I had been hesitant about creating yet another traceability system – the ones we had tried in the past never scaled. Now I understand that was because they were centralized databases. Blockchain, with its decentralized, shared ledger felt like it was made for the food system!” With the business interest in blockchain technology confirmed, Walmart started working on two proof of concept (POC) projects with their technology partner IBM

Choosing the Blockchain

Walmart Technology considered several blockchain technologies but ultimately decided to go for Hyperledger Fabric. “IBM brought Hyperledger Fabric to us. We looked into Ethereum, Burrow project and others. Ultimately, we decided to go with Hyperledger Fabric because it met most of our needs for a blockchain technology,” Bedwell said. “We felt that it best met our needs. It is an enterprise-grade blockchain technology, and it is permissioned.” The team also found it important to work with an open-source, vendor-neutral blockchain. Since the food traceability system was meant to be used by many parties, including Walmart’s suppliers and even direct competitors, the technology ecosystem underlying it needed to be open. Hyperledger Fabric is a blockchain framework implementation and one of the Hyperledger projects hosted by The Linux Foundation. Intended as a foundation for developing applications or solutions with a modular architecture, Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and- play. Hyperledger Fabric leverages container technology to host smart contracts called “chaincode” that comprise the application logic of the system



The POCs

In October 2016, Walmart, together with its technology partner IBM, announced the two projects: one was about tracing the origin of mangos sold in Walmart's US stores and the other aimed to trace pork sold in its China stores. For the mango POC, Yiannas started by creating a benchmark. He bought a packet of sliced mangoes at a nearby Walmart store and asked his team to identify which farm they had come from – as fast as possible. The team started calling and emailing distributors and suppliers, and eventually had an answer almost seven days later. This was not bad by industry standards, but Walmart wanted to do much better. So together with IBM, they got to work building a blockchain-based food traceability system.

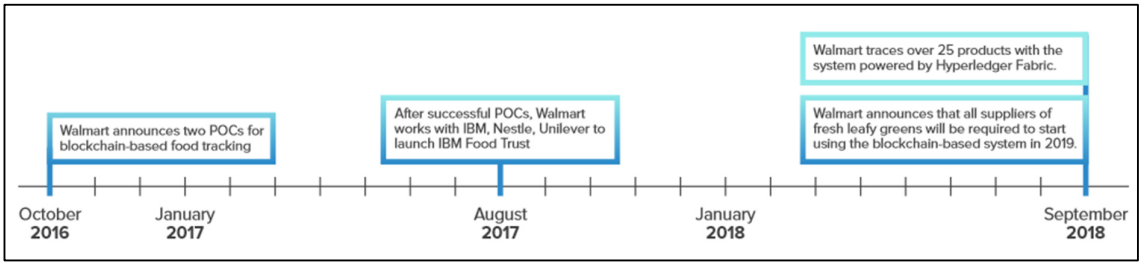
The Walmart Technology team looked at their own processes as well as those of their suppliers to design the application. Archana Sristy, Director of Engineering at Walmart, explains, "[Our team at Walmart Technology] co-led the core design and setup of the application (with IBM), as well as built the integration with the enterprise systems. We worked with GS1 (the standards authority in barcodes and labeling) to define the data attributes for upload to the blockchain. IBM wrote the chaincode. Suppliers used new labels and uploaded their data through a web-based interface. The Hyperledger Fabric blockchain-based food traceability system built for the two products worked. For pork in China, it allowed uploading certificates of authenticity to the blockchain, bringing more trust to a system where that used to be a serious issue. And for mangoes in the US, the time needed to trace their provenance went from 7 days to... 2.2 seconds!

> Says Yiannas, "I really had an 'aha' moment once I deeply understood the technology. I had been hesitant about creating yet another traceability system — the ones we had tried in the past never scaled. Now I understand that was because they were centralized databases. Blockchain, with its decentralized, shared ledger felt like it was made for the food system!"

From POC to production, from Walmart to IBM Food Trust

Once Walmart saw that the system worked, they wanted to expand it – and not just within Walmart. Given the interconnected nature of the food system and the company's negative experience with closed systems, Walmart wanted to make sure that this time, many players were involved. Says Yiannas, "(Walmart's) CEO was reaching out to other food companies the next day, including other retailers!" Wal-Mart collaborated with IBM and others to set up IBM Food Trust, involving prominent players in the food industry, like Nestle and Unilever.

The Walmart team had a positive experience working with Hyperledger. “Every question that we had, it looked like the Hyperledger community had already been working on addressing that,” says Bedwell. For example, in building a truly open system, the Walmart team worried about interoperability with other blockchain-based traceability systems. And as if in answer to their concern, Hyperledger recently announced its collaboration with Ethereum. He adds, “It seems that the Hyperledger community is addressing everything that enterprises would be concerned about.”



Looking forward:

Walmart now traces over 25 products from 5 different suppliers using IBM Blockchain which is built atop Hyperledger Fabric. The products include produce such as mangoes, strawberries and leafy greens; meat and poultry such as chicken and pork; dairy such as yogurt and almond milk; and even multi-ingredient products such as packaged salads and baby foods. Yiannas says of the impact, “This solution allows us to see the whole chain in seconds! We can take a jar of baby food and see where it was manufactured and trace back all the ingredients to the farms!” Walmart plans to roll out the system to more products and categories in the near future in cooperation with IBM Food Trust.. In fact, the company recently announced that it will start requiring all of its suppliers of fresh leafy greens (like salad and spinach) to trace their products using the system. “Using the IBM Food Trust network that relies on blockchain technology, we have shown that we can reduce the amount of time it takes to track a food item from a Walmart Store back to source in seconds, as compared to days or sometimes weeks,” Walmart wrote in a letter to suppliers. Beyond tracing the products’ journey, the company might start tracing other data, like sustainability.

About Walmart

Walmart Inc. (NYSE: WMT) helps people around the world save money and live better – anytime and anywhere – in retail stores, online, and through their mobile devices. Each week, nearly 265 million customers and members visit our more than 11,200 stores under 55 banners in 27 countries and eCommerce websites. With fiscal year 2018 revenue of $500.3 billion, Walmart employs over 2.2 million associates worldwide. Walmart continues to be a leader in sustainability, corporate philanthropy and employment opportunity. Additional information about Walmart can be found by visiting http://corporate.walmart.com.

About Hyperledger

Hyperledger is an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration including leaders in banking, finance, Internet of Things, manufacturing, supply chain, and technology. The Linux Foundation hosts Hyperledger under the foundation. To learn more, visit https://www.hyperledger.org/