Blockchain Technology
Practical Journal

Submitted By

Name: Adiba Mohammed Raza Siddique
Class: MSC CS (Part 2)
Seat No: 31031523033

Department of Computer Science
Somaiya Vidyavihar University
S K Somaiya College

INDEX

Practical 1

Aim: Creating a simple Blockchain to calculate the sum of two numbers.

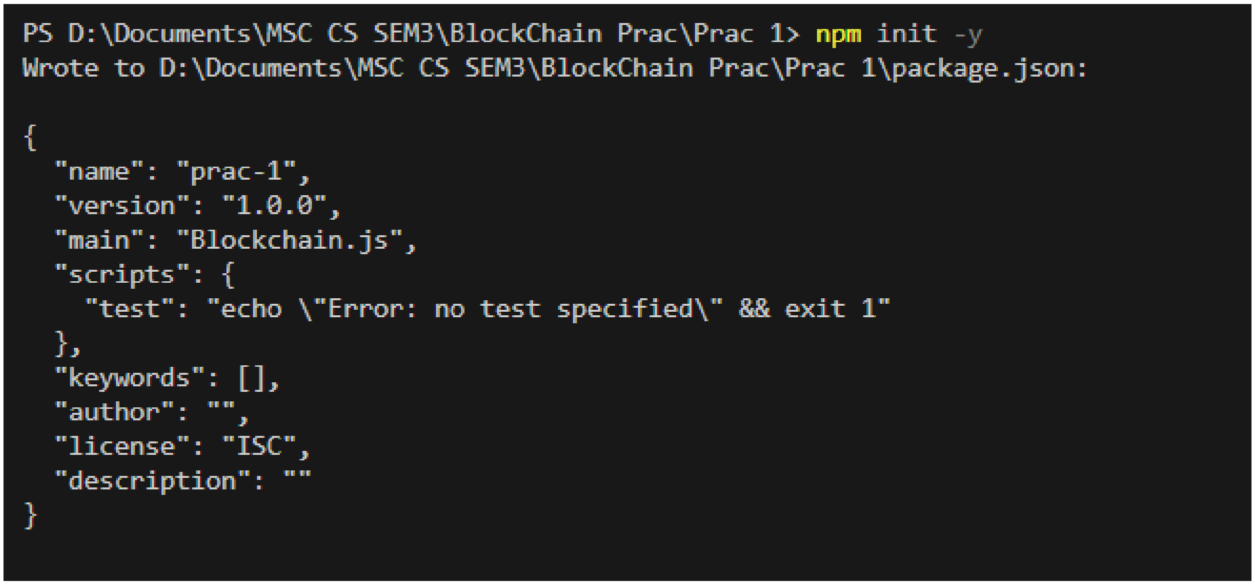
Blockchain.js:

```
const c = require('crypto');

class Block {
    constructor(i, t, n1, n2, ph = '') {
        this.i = i;
        this.t = t;
        this.n1 = n1;
        this.n2 = n2;
        this.sum = n1 + n2;
        this.ph = ph;
        this.h = this.calhash();
    }

    calhash() {
        return c.createHash('sha256').update(this.i + this.t + this.sum +
this.ph).digest('hex');
    }
}

class BlockChain {
    constructor() {
        this.chain = [this.createGBlock()];
    }

    createGBlock() {
        return new Block(0, '01/08/2024', 0, 0, '0');
    }

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

    addBlock(nb) {
        nb.ph = this.getcBlock().h;
        nb.h = nb.calhash();
        this.chain.push(nb);
    }
}

module.exports = { Block, BlockChain };
```

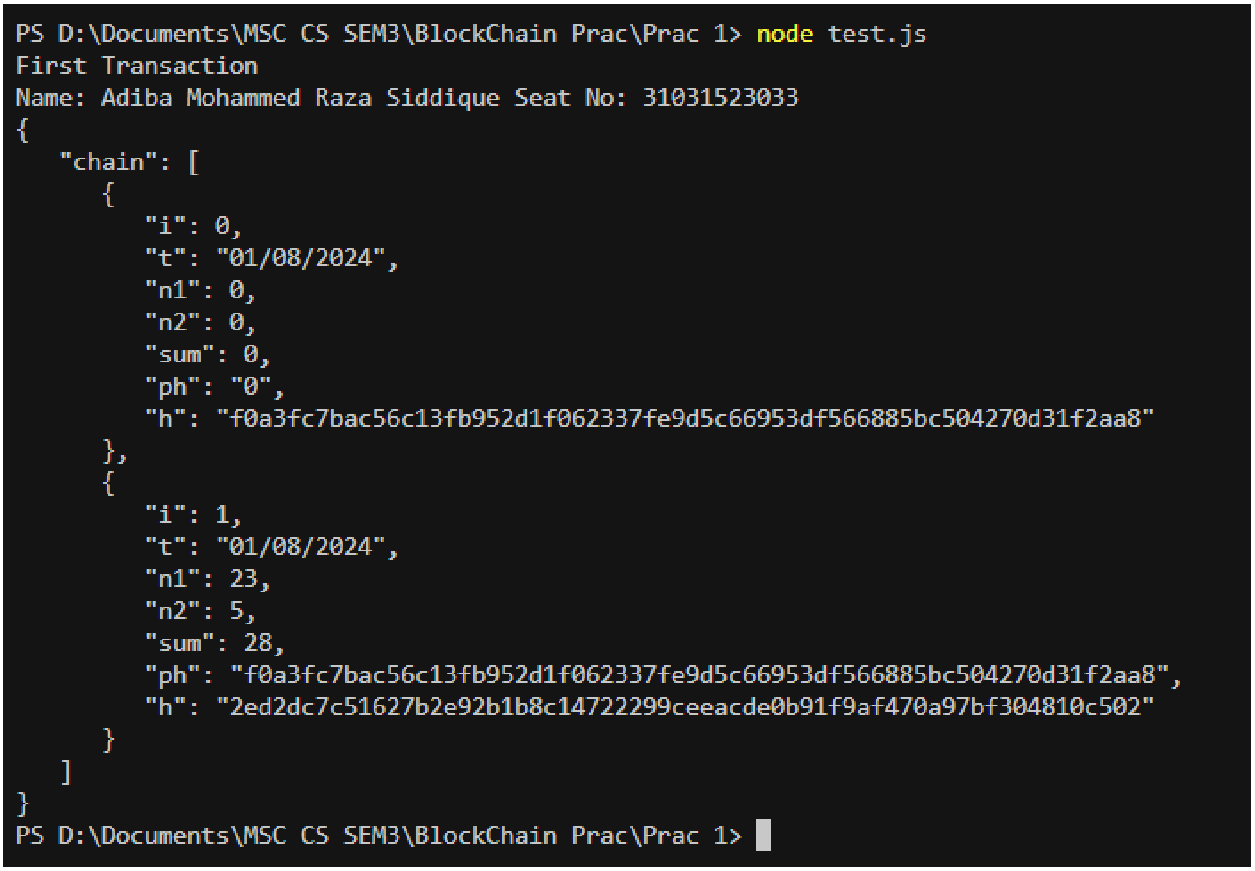
test.js:

```
const { Block, BlockChain } = require('./Blockchain');

let mb = new BlockChain();

console.log("First Transaction");
console.log("Name: Adiba Mohammed Raza Siddique Seat No: 31031523033");
mb.addBlock(new Block(1, '01/08/2024', 23, 5));

console.log(JSON.stringify(mb, null, 3));
```

Output:



```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 1> npm init -y
Wrote to D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 1\package.json:

{
  "name": "prac-1",
  "version": "1.0.0",
  "main": "Blockchain.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```



```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 1> node test.js
First Transaction
Name: Adiba Mohammed Raza Siddique Seat No: 31031523033
{
   "chain": [
      {
         "i": 0,
         "t": "01/08/2024",
         "n1": 0,
         "n2": 0,
         "sum": 0,
         "ph": "0",
         "h": "f0a3fc7bac56c13fb952d1f062337fe9d5c66953df566885bc504270d31f2aa8"
      },
      {
         "i": 1,
         "t": "01/08/2024",
         "n1": 23,
         "n2": 5,
         "sum": 28,
         "ph": "f0a3fc7bac56c13fb952d1f062337fe9d5c66953df566885bc504270d31f2aa8",
         "h": "2ed2dc7c51627b2e92b1b8c14722299ceeacde0b91f9af470a97bf304810c502"
      }
   ]
}
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 1>
```

Practical 2

A] Aim:- Creating a simple blockchain to calculate the factorial of numbers.

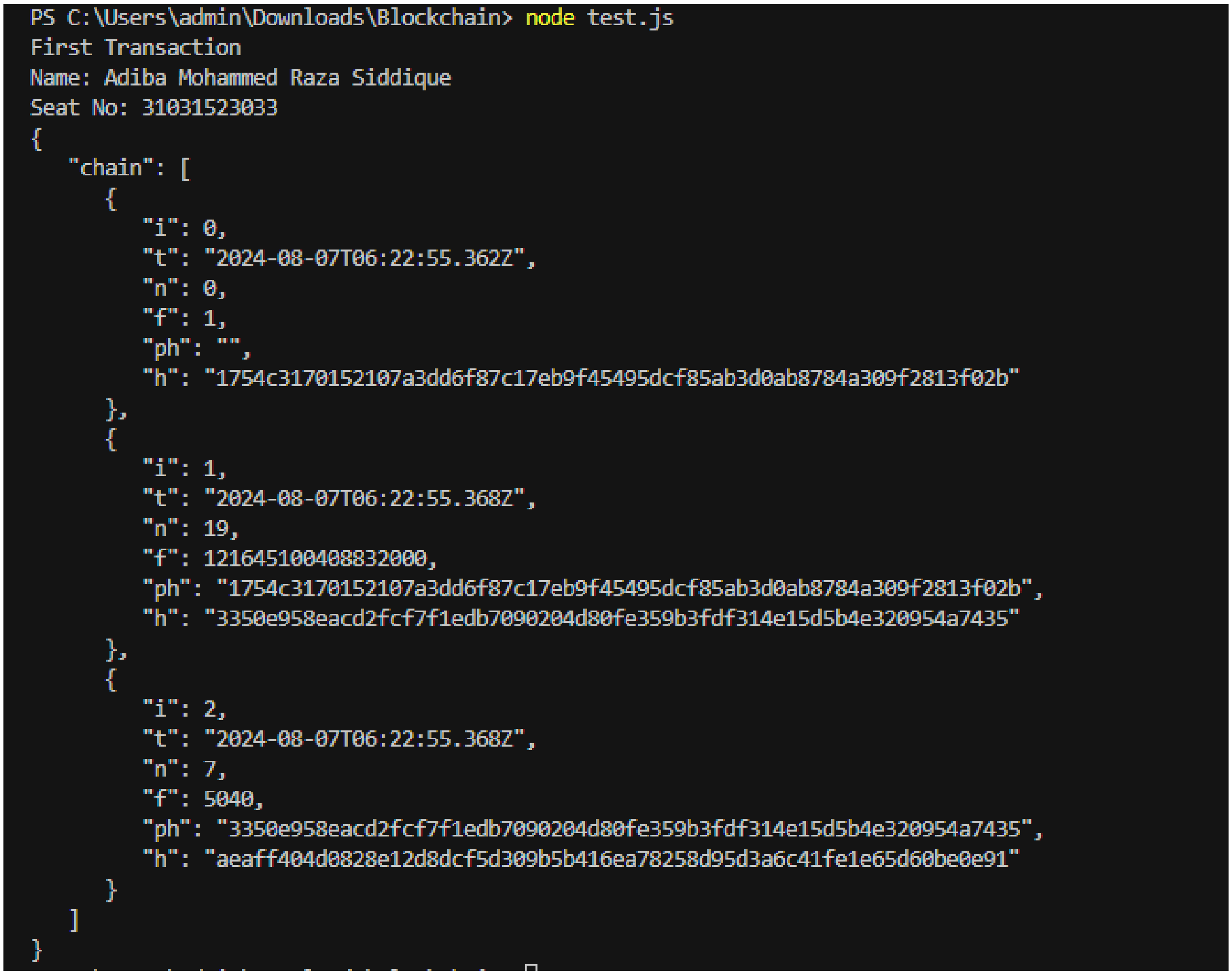
FactChain.js

```
const c = require('crypto');

class Block {
    constructor(i, t, n, f, ph = '') {
        this.i = i;
        this.t = t;
        this.n = n;
        this.f = this.calFact();
        this.ph = ph;
        this.h = this.calhash();
    }


    calFact() {
        let f;
        if (this.n === 0) {
            f = 1;
        } else {
            f = 1;
            for (let i = 1; i <= this.n; i++) {
                f *= i;
            }
        }
        return f;
    }




    calhash() {
        return c.createHash('sha256').update(this.i + this.t + this.n +
this.f + this.ph).digest('hex');


    }
}
    class Factchain {
        constructor(){
    this.chain=[this.genesisBlock()];
        }

        genesisBlock(){
            return new Block(0, new Date(),0)

        }

        getcBlock() {
            return this.chain[this.chain.length - 1];
    }
        addBlock(nb) {
            nb.ph = this.getcBlock().h;
            nb.h = nb.calhash();
            this.chain.push(nb);
    }

}
module.exports = { Block, Factchain };
```

Test.js

```
const { Block, Factchain } = require('./FactChain');

let mb = new Factchain();

console.log("First Transaction");
console.log("Name: Adiba Mohammed Raza Siddique\nSeat No: 31031523033");
mb.addBlock(new Block(1, new Date(), 4));
mb.addBlock(new Block(2, new Date(), 7));

console.log(JSON.stringify(mb, null, 3));
```

Output:



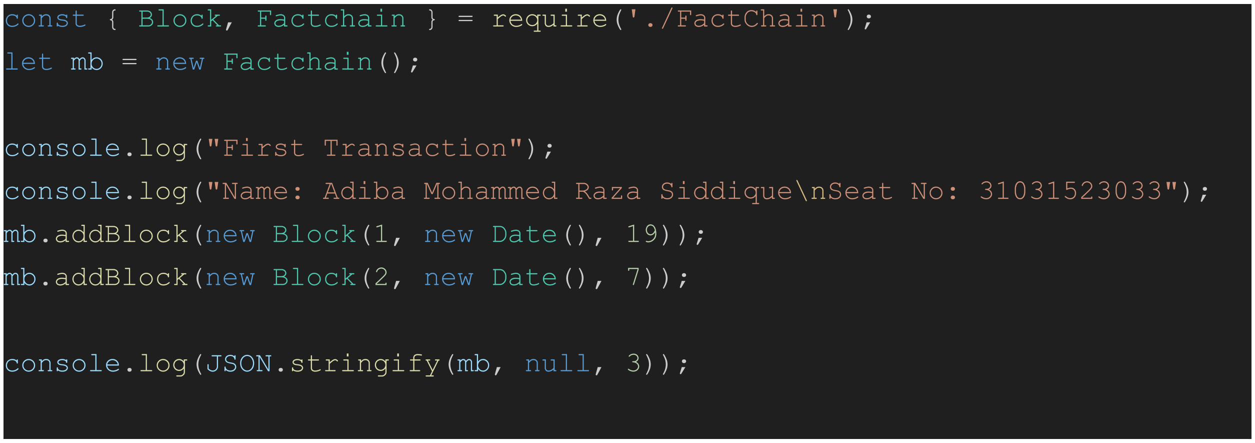
```
PS C:\Users\admin\Downloads\Blockchain> node test.js
First Transaction
Name: Adiba Mohammed Raza Siddique
Seat No: 31031523033
{
   "chain": [
      {
         "i": 0,
         "t": "2024-08-07T06:22:55.362Z",
         "n": 0,
         "f": 1,
         "ph": "",
         "h": "1754c3170152107a3dd6f87c17eb9f45495dcf85ab3d0ab8784a309f2813f02b"
      },
      {
         "i": 1,
         "t": "2024-08-07T06:22:55.368Z",
         "n": 19,
         "f": 121645100408832000,
         "ph": "1754c3170152107a3dd6f87c17eb9f45495dcf85ab3d0ab8784a309f2813f02b",
         "h": "3350e958eacd2fcf7f1edb7090204d80fe359b3fdf314e15d5b4e320954a7435"
      },
      {
         "i": 2,
         "t": "2024-08-07T06:22:55.368Z",
         "n": 7,
         "f": 5040,
         "ph": "3350e958eacd2fcf7f1edb7090204d80fe359b3fdf314e15d5b4e320954a7435",
         "h": "aeaff404d0828e12d8dcf5d309b5b416ea78258d95d3a6c41fe1e65d60be0e91"
      }
   ]
}
```

B] Aim:- Creating a simple blockchain to calculate the happy number.

FactChain.js

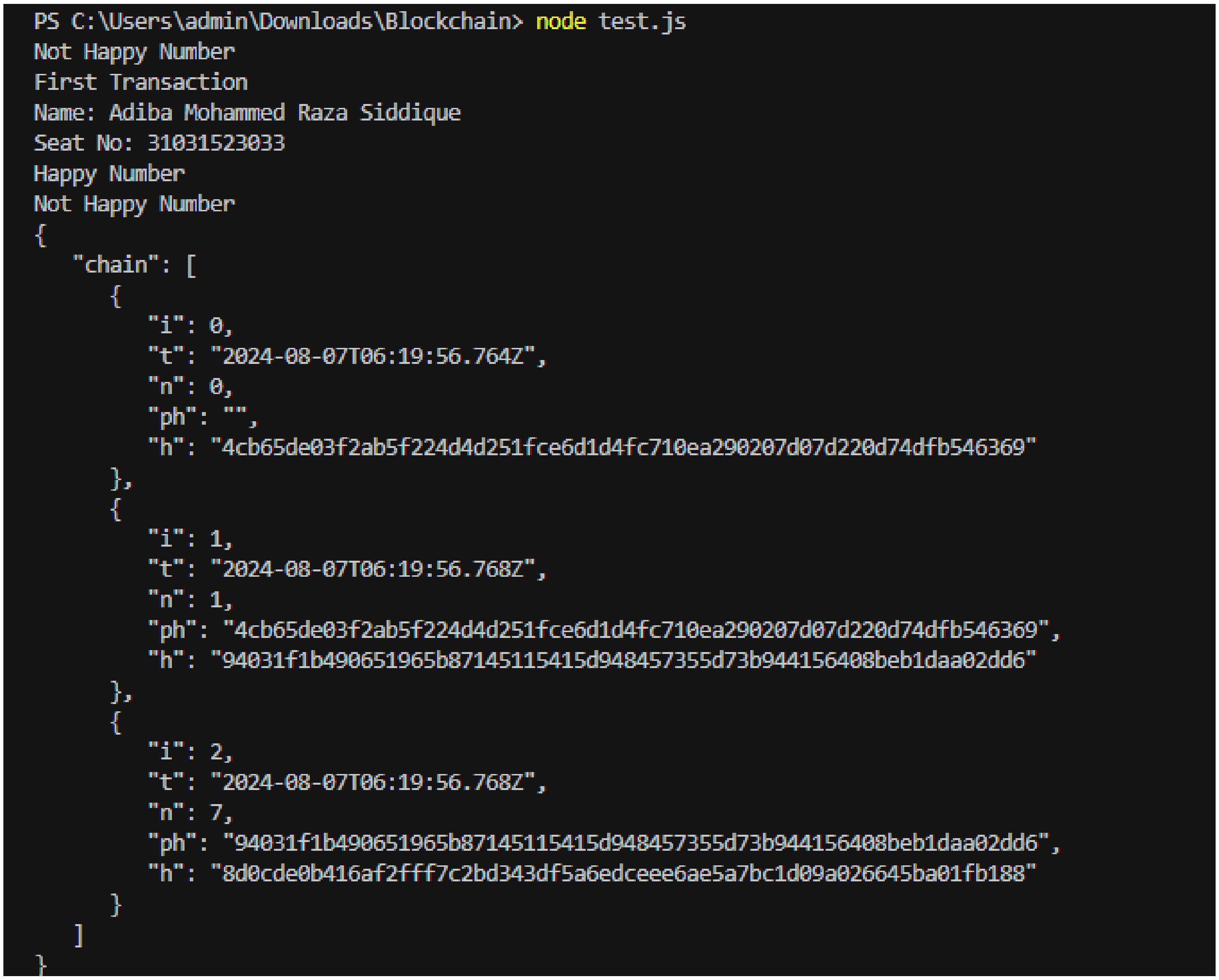
```
const c = require('crypto');


class Block {
    constructor(i, t, n, ph = '') {
        this.i = i;
        this.t = t;
        this.n = n;
        this.ph = ph;
        this.f = this.calHappyNumber();
        this.h = this.calhash();
    }


    calHappyNumber() {
        while(this.n>9)
            {
                let sum = 0;
                while(this.n>0)
                {
                    let reminder = this.n%10;
                    this.n = Math.floor(this.n/10);
                    let sqr = reminder*reminder;
                    sum+=sqr;
                }


                this.n = sum;
            }


            if (this.n==1)
                console.log("Happy Number")


            else
            {
                console.log("Not Happy Number")
            }



    }

    calhash() {
        return
c.createHash('sha256').update(`${this.i}${this.t}${this.n}${this.f}${this.
ph}`).digest('hex');
    }
}

class Factchain {
    constructor() {
        this.chain = [this.genesisBlock()];
    }


    genesisBlock() {
        return new Block(0, new Date().toISOString(), 0);
    }


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

    addBlock(nb) {
        nb.ph = this.getBlock().h;
        nb.h = nb.calhash();
        this.chain.push(nb);
    }
}
module.exports.Block = Block;
module.exports.Factchain = Factchain;
```

Test.js



```
const { Block, Factchain } = require('./FactChain');
let mb = new Factchain();

console.log("First Transaction");
console.log("Name: Adiba Mohammed Raza Siddique\nSeat No: 31031523033");
mb.addBlock(new Block(1, new Date(), 19));
mb.addBlock(new Block(2, new Date(), 7));

console.log(JSON.stringify(mb, null, 3));
```

Output:



```
PS C:\Users\admin\Downloads\Blockchain> node test.js
Not Happy Number
First Transaction
Name: Adiba Mohammed Raza Siddique
Seat No: 31031523033
Happy Number
Not Happy Number
{
   "chain": [
      {
         "i": 0,
         "t": "2024-08-07T06:19:56.764Z",
         "n": 0,
         "ph": "",
         "h": "4cb65de03f2ab5f224d4d251fce6d1d4fc710ea290207d07d220d74dfb546369"
      },
      {
         "i": 1,
         "t": "2024-08-07T06:19:56.768Z",
         "n": 1,
         "ph": "4cb65de03f2ab5f224d4d251fce6d1d4fc710ea290207d07d220d74dfb546369",
         "h": "94031f1b490651965b87145115415d948457355d73b944156408beb1daa02dd6"
      },
      {
         "i": 2,
         "t": "2024-08-07T06:19:56.768Z",
         "n": 7,
         "ph": "94031f1b490651965b87145115415d948457355d73b944156408beb1daa02dd6",
         "h": "8d0cde0b416af2fff7c2bd343df5a6edceee6ae5a7bc1d09a026645ba01fb188"
      }
   ]
}
```

Practical 3

Aim:- To check and validate Kaprekar number.

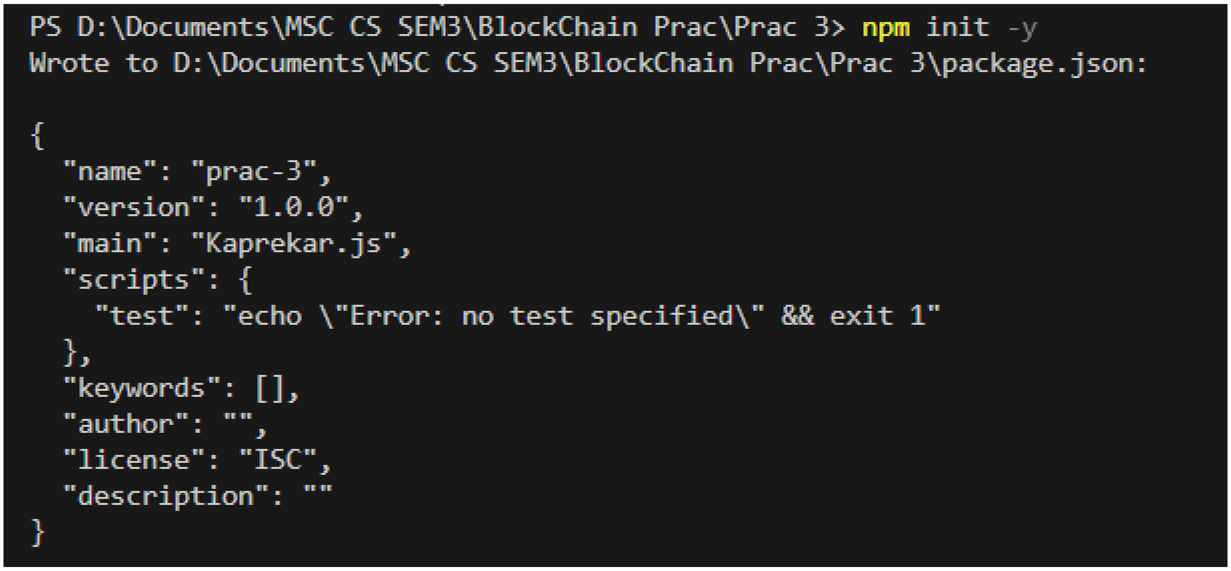
Kaprekar.js

```
const c = require('crypto');
class Block
{
    constructor(i,t,n1,ph='')
    {
        this.i = i;
        this.t = t;
        this.n1 = n1;
        this.ph = ph;
        this.h = this.calhash();

     }
    checkkaps()
    {
        var cnt = 0, x = this.n1,sq = x*x;
        var y = x,r,f,b,sum,sq1,rem;
        while (x!=0)
        {
            rem = x%10;
            cnt++;
            x=parseInt(x/10);
        }
        r = Math.pow(10,cnt);
        r = parseInt(r);
        f = parseInt(sq/r);
        b = parseInt(sq%r);
        sum = f + b;
        if (sum == y)
        {
            return true;
        }
        else
        {
        return false;
        }

    }
    calhash()
    {
        return
c.createHash('sha256').update(this.i+this.t+this.n1+this.ph).digest('hex')
;
    }
}

class Blockchain
{
    constructor()
    {
        this.chain = [this.create_GBlock()];
    }
    create_GBlock()
    {
        return new Block(0,'07-08-24',0,'0');
    }
    Getcurr_block()
    {
        return this.chain[this.chain.length - 1];
    }
    Add_Block(nb)
    {
        if(nb.checkkaps() == true)
    {
        nb.ph = this.Getcurr_block().h;
        nb.h = nb.calhash();
        this.chain.push(nb);
    }
}
validate()
{
    for(let i = 1; i < this.chain.length; i++)
    {
        let cb = this.chain[i];
        let pb = this.chain[i-1];
        if(cb.h != cb.calhash())
        {
            return false;
        }
        if(pb.h != cb.ph)
        {
            return false;
        }
    }
    return true;
}
}
module.exports.Block = Block;
module.exports.Blockchain = Blockchain;
```

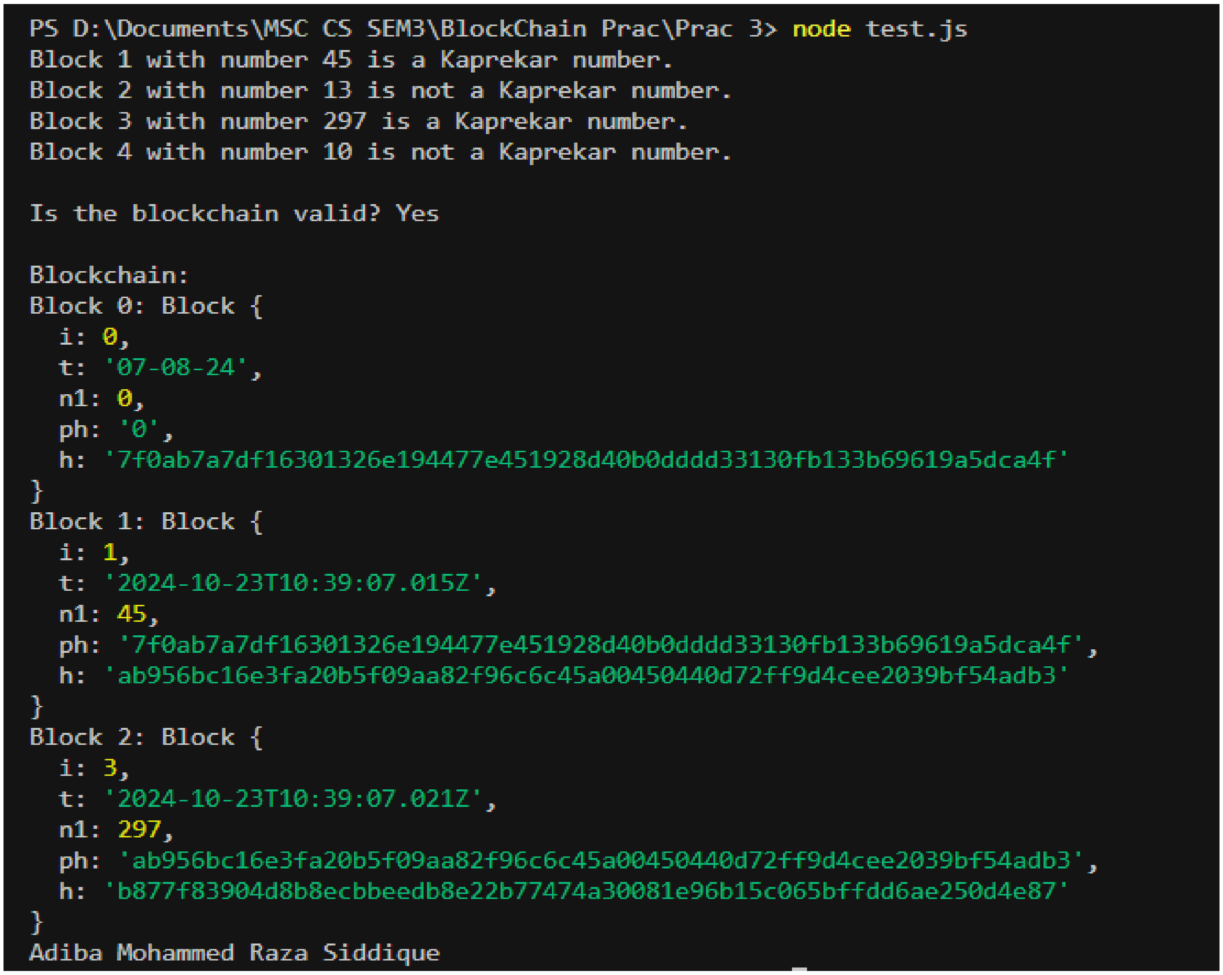
test.js

```
const { Block, Blockchain } = require('./Kaprekar');

const blockchain = new Blockchain();

const testNumbers = [45, 13, 297, 10];

testNumbers.forEach((num, index) => {
    const block = new Block(index + 1, new Date().toISOString(), num);
    blockchain.Add_Block(block);
    console.log(`Block ${index + 1} with number ${num} ${block.checkkaps()
? 'is a Kaprekar number.' : 'is not a Kaprekar number.'}`);
});

const isChainValid = blockchain.validate();
console.log(`\nIs the blockchain valid? ${isChainValid ? 'Yes' : 'No'}`);

console.log("\nBlockchain:");
blockchain.chain.forEach((block, index) => {
    console.log(`Block ${index}:`, block);
});

    console.log('Adiba Mohammed Raza Siddique');
```

Output:



```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 3> npm init -y
Wrote to D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 3\package.json:

{
  "name": "prac-3",
  "version": "1.0.0",
  "main": "Kaprekar.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```



```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 3> node test.js
Block 1 with number 45 is a Kaprekar number.
Block 2 with number 13 is not a Kaprekar number.
Block 3 with number 297 is a Kaprekar number.
Block 4 with number 10 is not a Kaprekar number.

Is the blockchain valid? Yes

Blockchain:
Block 0: Block {
  i: 0,
  t: '07-08-24',
  n1: 0,
  ph: '0',
  h: '7f0ab7a7df16301326e194477e451928d40b0dddd33130fb133b69619a5dca4f'
}
Block 1: Block {
  i: 1,
  t: '2024-10-23T10:39:07.015Z',
  n1: 45,
  ph: '7f0ab7a7df16301326e194477e451928d40b0dddd33130fb133b69619a5dca4f',
  h: 'ab956bc16e3fa20b5f09aa82f96c6c45a00450440d72ff9d4cee2039bf54adb3'
}
Block 2: Block {
  i: 3,
  t: '2024-10-23T10:39:07.021Z',
  n1: 297,
  ph: 'ab956bc16e3fa20b5f09aa82f96c6c45a00450440d72ff9d4cee2039bf54adb3',
  h: 'b877f83904d8b8ecbbeedb8e22b77474a30081e96b15c065bffdd6ae250d4e87'
}
Adiba Mohammed Raza Siddique
```

Practical 4

Aim:- Create a simple blockchain to store only automorphic numbers, also secure your Automorpohic number by applying DES Algorithm and also validate the block before adding it to the blockchain.

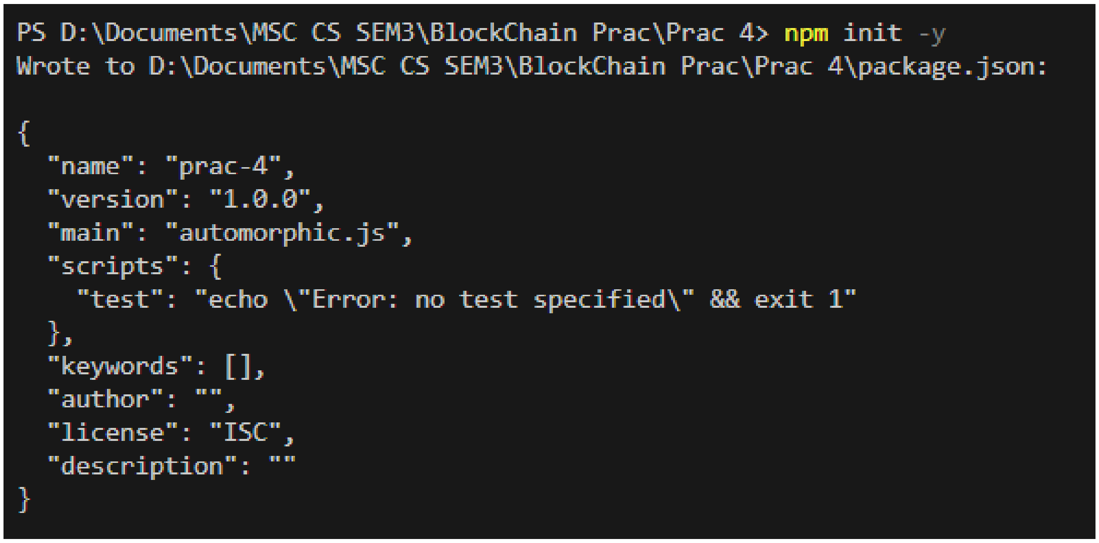
Filename: automorphic.js

```
const crypto = require('crypto');


class Block {
    constructor(i, t, n, ph = '') {
        this.i = i;
        this.t = t;
        this.n = n;
        this.ph = ph;
        this.f = this.isAutomorphic();
        this.h = this.calhash();
    }


    isAutomorphic() {
        let n1 = this.n;
        let sq = n1 * n1;
        let c = 0;
        let flag = false;


        if (sq !== 0) {
            while (sq !== 0) {
                c = c + 1;
                sq = Math.floor(sq / 10);
            }


            for (let i = 1; i < c; i++) {
                let sq1 = n1 * n1;
                let r = sq1 % Math.pow(10, i);
                if (this.n === r) {
                    flag = true;
                    break;
                } else {
                    flag = false;
                }
            }


            this.f = flag ? "Automorphic" : "Not Automorphic";
        } else {
            this.f = "Automorphic";
        }


        return this.f;
    }


    calhash() {
        return crypto.createHash('sha256').update(this.i + this.t + this.n + this.ph).digest('hex');
    }
}


class Blockchain {
    constructor() {
        this.chain = [this.createGenesisBlock()];
    }


    createGenesisBlock() {
        return new Block(0, '07-08-24', 0, '0');
    }


    getCurrentBlock() {
        return this.chain[this.chain.length - 1];
    }
    addBlock(newBlock) {
        if (newBlock.isAutomorphic() === "Automorphic") {
            newBlock.ph = this.getCurrentBlock().h;
            newBlock.h = newBlock.calhash();
            this.chain.push(newBlock);
        }
    }


    validate() {
        for (let i = 1; i < this.chain.length; i++) {
            let cb = this.chain[i];
            let pb = this.chain[i - 1];
            if (cb.h !== cb.calhash()) {
                return false;
            }
            if (pb.h !== cb.ph) {
                return false;
            }
        }
        return true;
    }
}


module.exports.Block = Block;
module.exports.Blockchain = Blockchain;
```

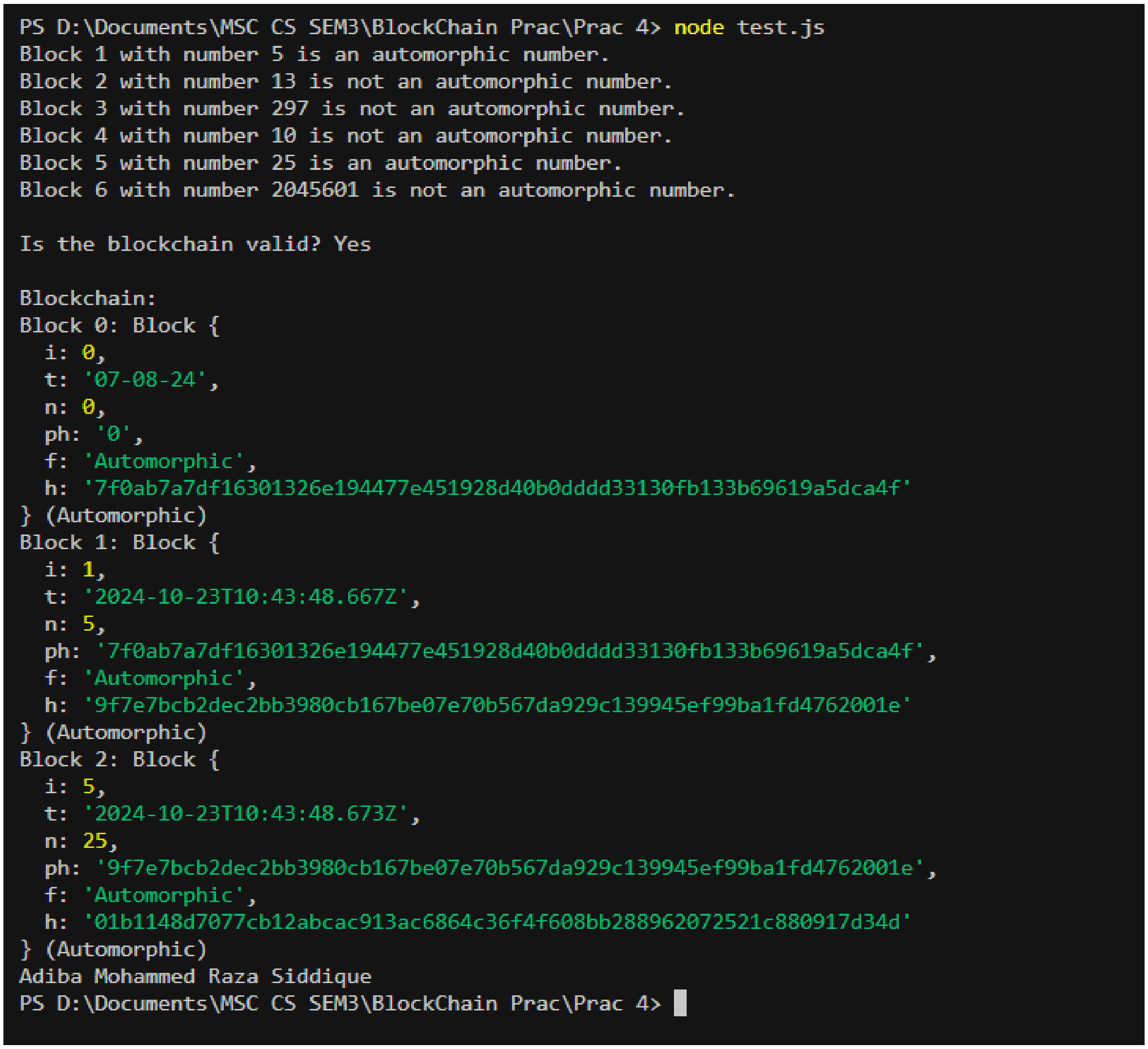
test.js

```
const { Block, Blockchain } = require('./automorphic');


const blockchain = new Blockchain();


const testNumbers = [5, 13, 297, 10, 25,2045601];


testNumbers.forEach((num, index) => {
    const block = new Block(index + 1, new Date().toISOString(), num);
    blockchain.addBlock(block);
    console.log(`Block ${index + 1} with number ${num} ${block.f ===
'Automorphic' ? 'is an automorphic number.' : 'is not an automorphic
number.'}`);
});


const isChainValid = blockchain.validate();
console.log(`\nIs the blockchain valid? ${isChainValid ? 'Yes' : 'No'}`);


console.log("\nBlockchain:");
blockchain.chain.forEach((block, index) => {
    console.log(`Block ${index}:`, block, `(${block.f === 'Automorphic' ?
'Automorphic' : 'Not Automorphic'})`);
});


console.log('Adiba Mohammed Raza Siddique');
```

Output:



```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 4> npm init -y
Wrote to D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 4\package.json:

{
  "name": "prac-4",
  "version": "1.0.0",
  "main": "automorphic.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

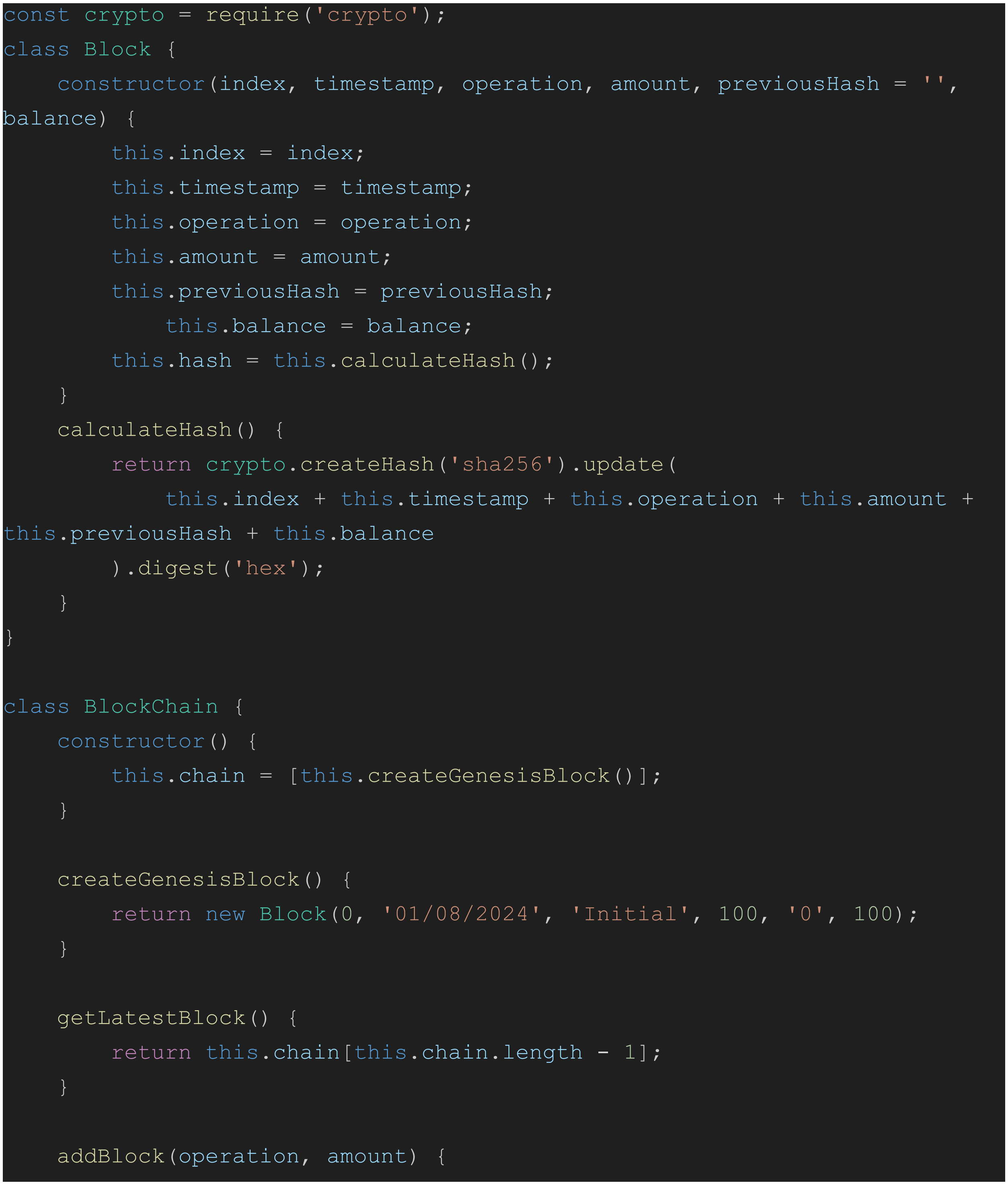


```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 4> node test.js
Block 1 with number 5 is an automorphic number.
Block 2 with number 13 is not an automorphic number.
Block 3 with number 297 is not an automorphic number.
Block 4 with number 10 is not an automorphic number.
Block 5 with number 25 is an automorphic number.
Block 6 with number 2045601 is not an automorphic number.

Is the blockchain valid? Yes

Blockchain:
Block 0: Block {
  i: 0,
  t: '07-08-24',
  n: 0,
  ph: '0',
  f: 'Automorphic',
  h: '7f0ab7a7df16301326e194477e451928d40b0dddd33130fb133b69619a5dca4f'
} (Automorphic)
Block 1: Block {
  i: 1,
  t: '2024-10-23T10:43:48.667Z',
  n: 5,
  ph: '7f0ab7a7df16301326e194477e451928d40b0dddd33130fb133b69619a5dca4f',
  f: 'Automorphic',
  h: '9f7e7bcb2dec2bb3980cb167be07e70b567da929c139945ef99ba1fd4762001e'
} (Automorphic)
Block 2: Block {
  i: 5,
  t: '2024-10-23T10:43:48.673Z',
  n: 25,
  ph: '9f7e7bcb2dec2bb3980cb167be07e70b567da929c139945ef99ba1fd4762001e',
  f: 'Automorphic',
  h: '01b1148d7077cb12abcac913ac6864c36f4f608bb288962072521c880917d34d'
} (Automorphic)
Adiba Mohammed Raza Siddique
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 4>
```

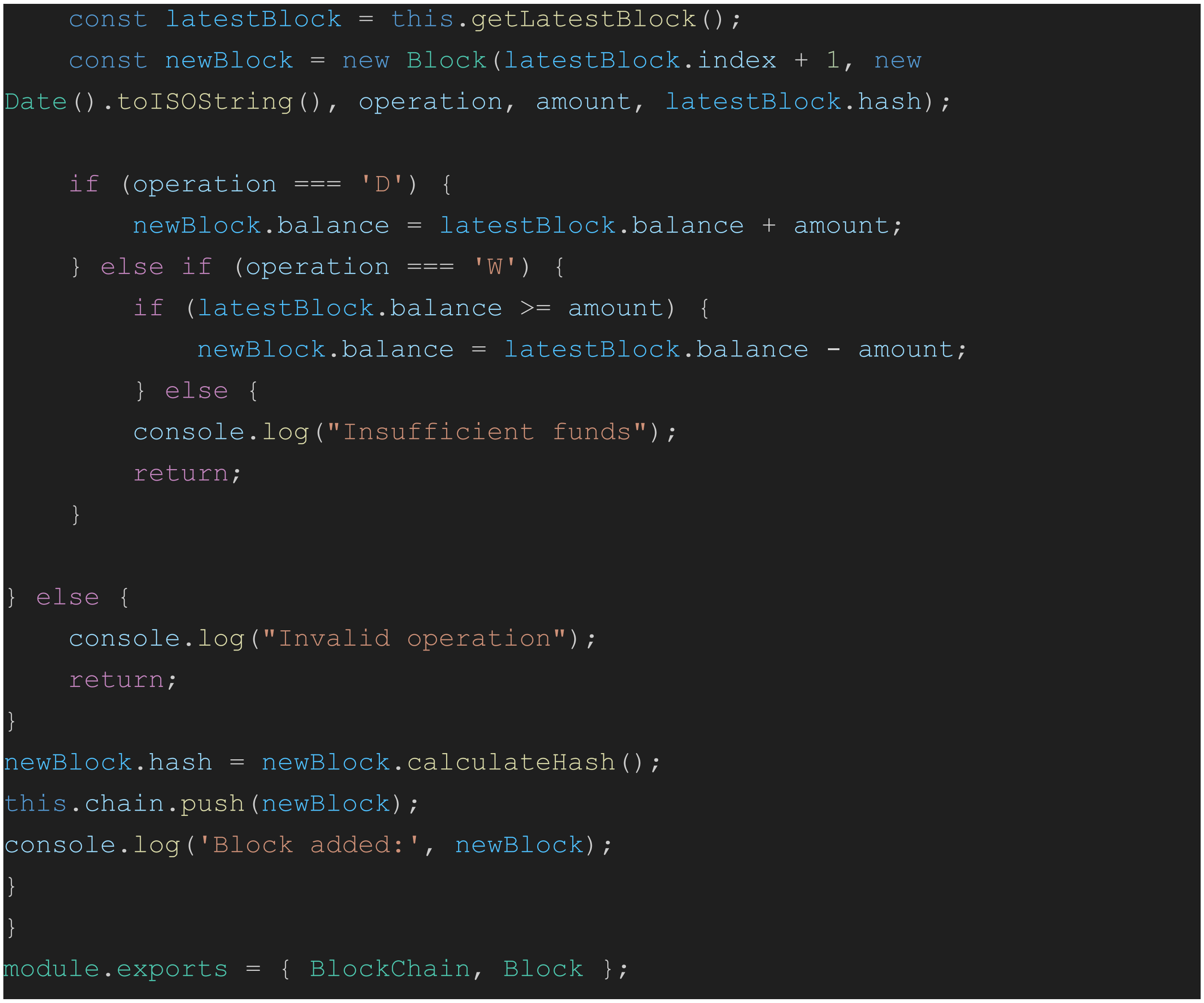
Practical 5

Aim:- Create a blockchain to deposit and withdraw money from the Block.

Filename: blockhain.js



```
const crypto = require('crypto');
class Block {
    constructor(index, timestamp, operation, amount, previousHash = '',
balance) {
        this.index = index;
        this.timestamp = timestamp;
        this.operation = operation;
        this.amount = amount;
        this.previousHash = previousHash;
            this.balance = balance;
        this.hash = this.calculateHash();
    }
    calculateHash() {
        return crypto.createHash('sha256').update(
            this.index + this.timestamp + this.operation + this.amount +
this.previousHash + this.balance
        ).digest('hex');
    }
}

class BlockChain {
    constructor() {
        this.chain = [this.createGenesisBlock()];
    }

    createGenesisBlock() {
        return new Block(0, '01/08/2024', 'Initial', 100, '0', 100);
    }

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

    addBlock(operation, amount) {
```

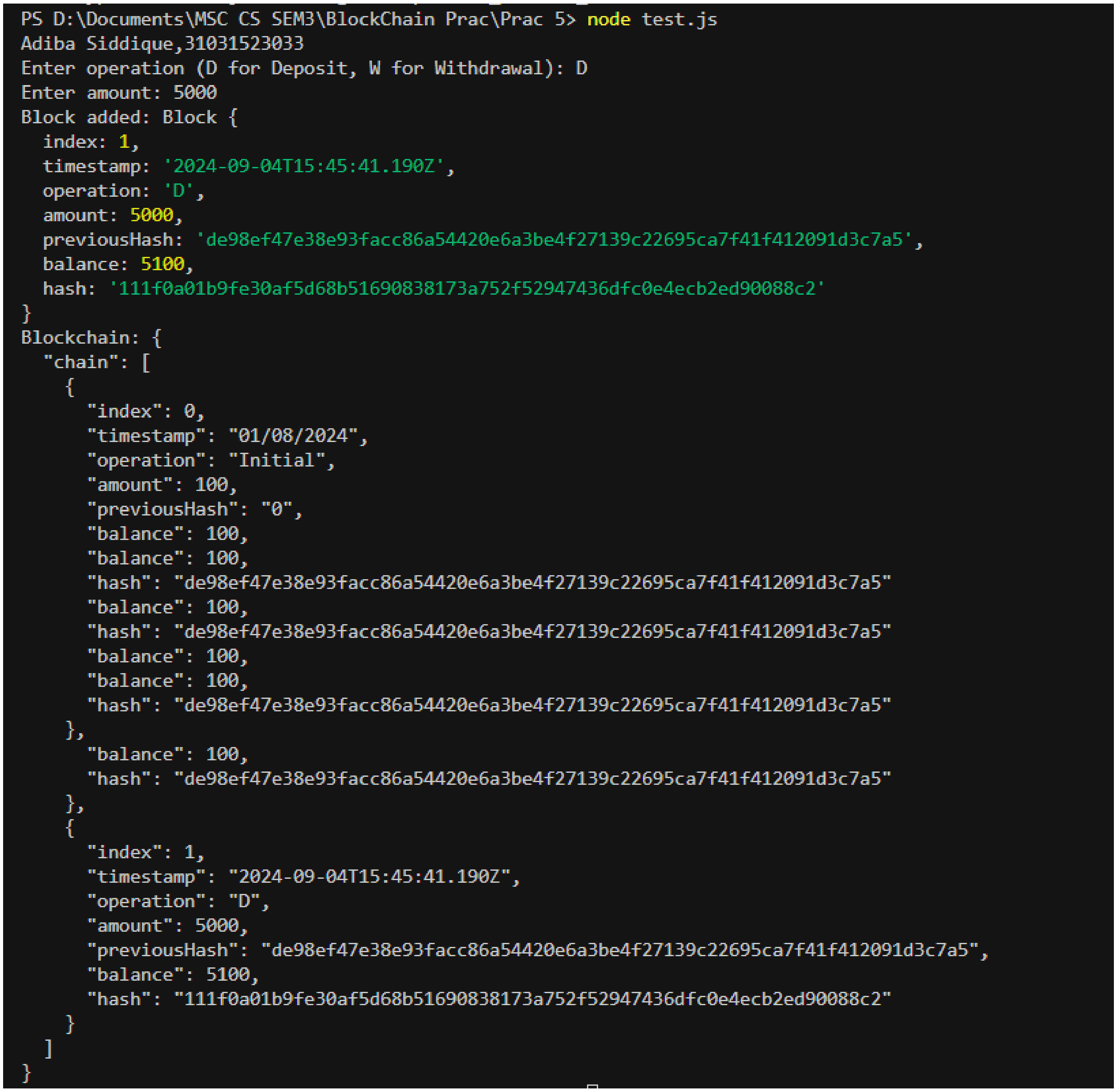


```
    const latestBlock = this.getLatestBlock();
    const newBlock = new Block(latestBlock.index + 1, new
Date().toISOString(), operation, amount, latestBlock.hash);

    if (operation === 'D') {
        newBlock.balance = latestBlock.balance + amount;
    } else if (operation === 'W') {
        if (latestBlock.balance >= amount) {
            newBlock.balance = latestBlock.balance - amount;
        } else {
        console.log("Insufficient funds");
        return;
    }

} else {
    console.log("Invalid operation");
    return;
}
newBlock.hash = newBlock.calculateHash();
this.chain.push(newBlock);
console.log('Block added:', newBlock);
}
}
module.exports = { BlockChain, Block };
```

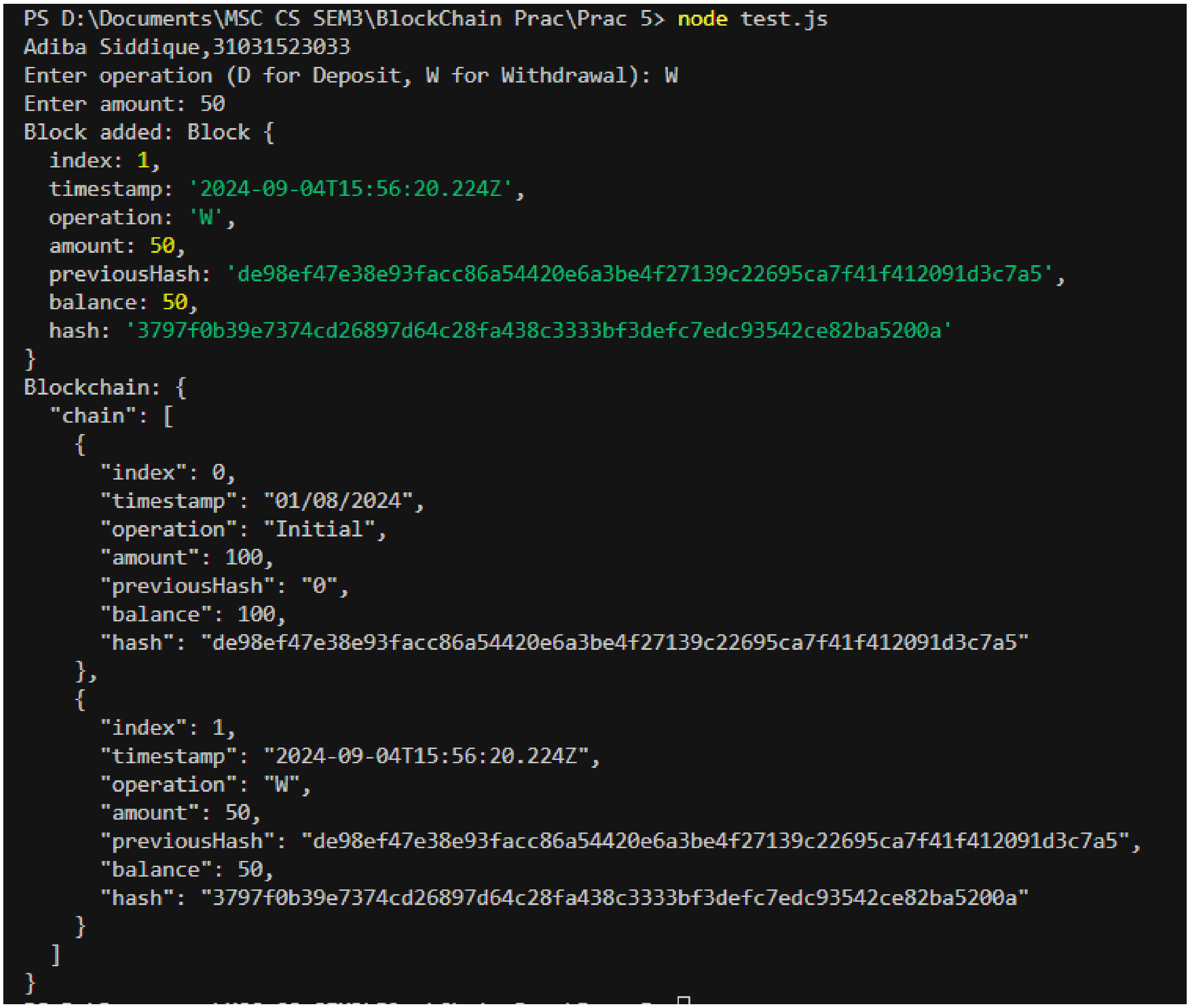
Filename: test.js

```
console.log("Adiba Siddique, 31031523033")
const { BlockChain } = require('./blockchain');
const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

const myChain = new BlockChain();

rl.question('Enter operation (D for Deposit, W for Withdrawal): ',
(operation) => {
    rl.question('Enter amount: ', (amount) => {
        amount = parseFloat(amount);
    if (operation !== 'D' && operation !== 'W') {
        console.log('Invalid operation.');
    } else if (isNaN(amount) || amount <= 0) {
        console.log('Invalid amount.');
    } else {
        myChain.addBlock(operation, amount);
    }

    console.log('Blockchain:', JSON.stringify(myChain, null, 2));
    rl.close();
});
});
```

Output:



```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 5> node test.js
Adiba Siddique,31031523033
Enter operation (D for Deposit, W for Withdrawal): D
Enter amount: 5000
Block added: Block {
  index: 1,
  timestamp: '2024-09-04T15:45:41.190Z',
  operation: 'D',
  amount: 5000,
  previousHash: 'de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5',
  balance: 5100,
  hash: '111f0a01b9fe30af5d68b51690838173a752f52947436dfc0e4ecb2ed90088c2'
}
Blockchain: {
  "chain": [
    {
      "index": 0,
      "timestamp": "01/08/2024",
      "operation": "Initial",
      "amount": 100,
      "previousHash": "0",
      "balance": 100,
      "balance": 100,
      "hash": "de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5"
      "balance": 100,
      "hash": "de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5"
      "balance": 100,
      "balance": 100,
      "hash": "de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5"
    },
      "balance": 100,
      "hash": "de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5"
    },
    {
      "index": 1,
      "timestamp": "2024-09-04T15:45:41.190Z",
      "operation": "D",
      "amount": 5000,
      "previousHash": "de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5",
      "balance": 5100,
      "hash": "111f0a01b9fe30af5d68b51690838173a752f52947436dfc0e4ecb2ed90088c2"
    }
  ]
}
```



```
PS D:\Documents\MSC CS SEM3\BlockChain Prac\Prac 5> node test.js
Adiba Siddique,31031523033
Enter operation (D for Deposit, W for Withdrawal): W
Enter amount: 50
Block added: Block {
  index: 1,
  timestamp: '2024-09-04T15:56:20.224Z',
  operation: 'W',
  amount: 50,
  previousHash: 'de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5',
  balance: 50,
  hash: '3797f0b39e7374cd26897d64c28fa438c3333bf3defc7edc93542ce82ba5200a'
}
Blockchain: {
  "chain": [
    {
      "index": 0,
      "timestamp": "01/08/2024",
      "operation": "Initial",
      "amount": 100,
      "previousHash": "0",
      "balance": 100,
      "hash": "de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5"
    },
    {
      "index": 1,
      "timestamp": "2024-09-04T15:56:20.224Z",
      "operation": "W",
      "amount": 50,
      "previousHash": "de98ef47e38e93facc86a54420e6a3be4f27139c22695ca7f41f412091d3c7a5",
      "balance": 50,
      "hash": "3797f0b39e7374cd26897d64c28fa438c3333bf3defc7edc93542ce82ba5200a"
    }
  ]
}
```

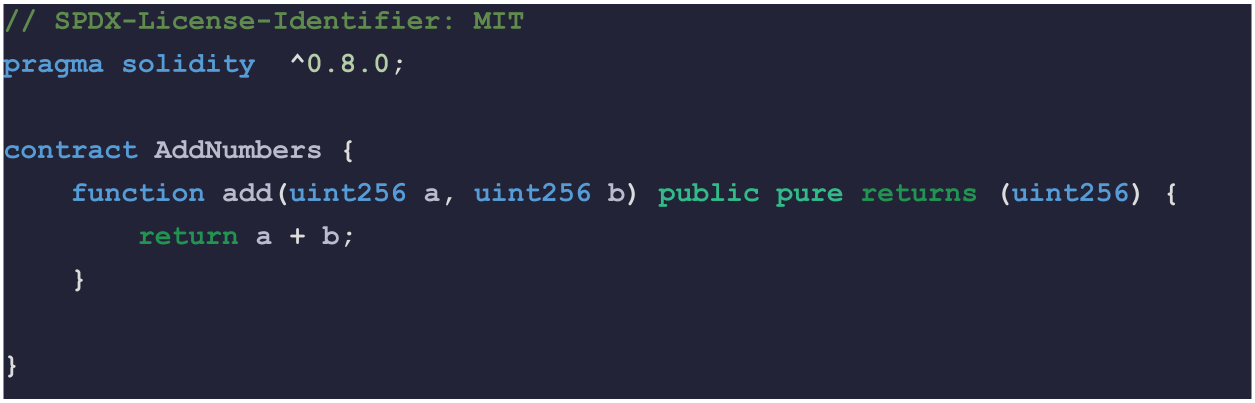
Practical 6

A] Aim:- Creating Smart Contract in Solidity.

Step 1: Open Remix-Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js

Step 2: Create a workspace > Create a blank project > click on OK

Step 3: Create a new file Named as mynew.sol



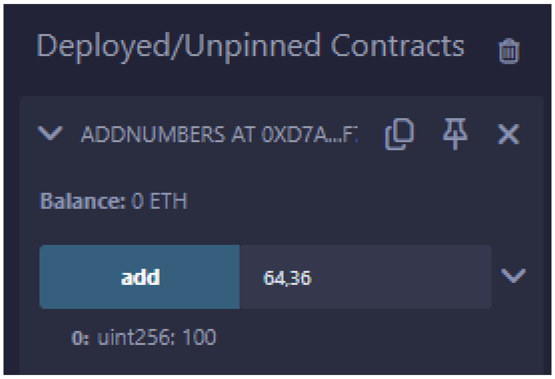
```
// SPDX-License-Identifier: MIT
pragma solidity  ^0.8.0;

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


}
```

Compile using CTRL+S

Step 4: Now go to left side panel and click on deploy and run Transaction

Step 5: Click on deploy

Step 6: Expand panel Deployed Contracts and the 2 values using comma

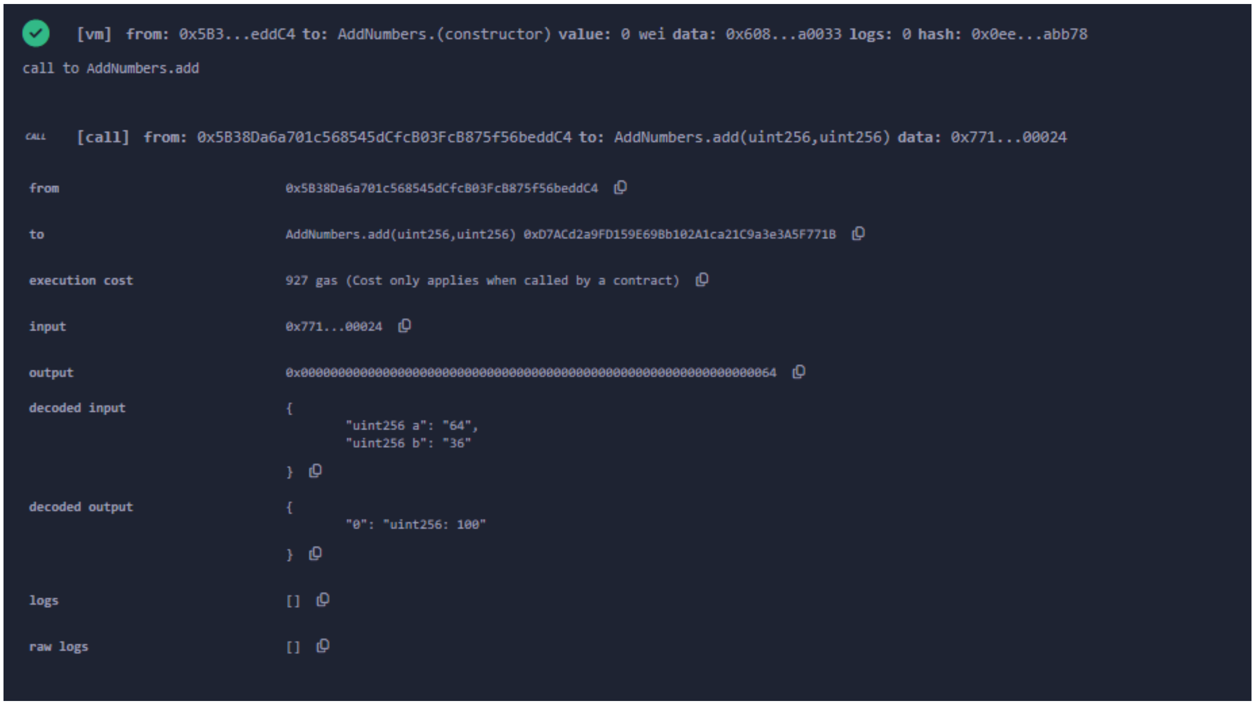


Balance: 0 ETH

64,36

0: uint256: 100

Output:



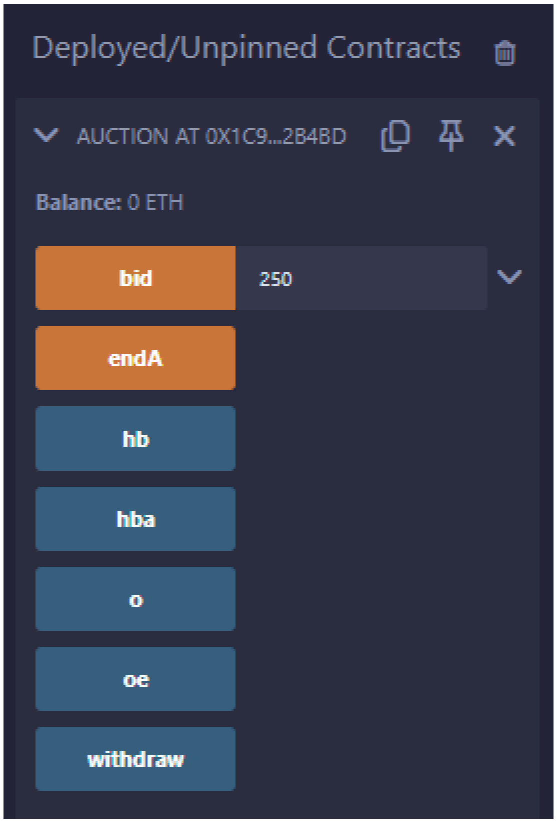
```
[vm] from: 0x5B3...eddC4 to: AddNumbers.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x0ee...abb78
call to AddNumbers.add

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: AddNumbers.add(uint256,uint256) data: 0x771...00024

from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to                      AddNumbers.add(uint256,uint256) 0xD7ACd2a9FD159E69Bb102A1ca21C9a3e3A5F771B

execution cost          927 gas (Cost only applies when called by a contract)

input                   0x771...00024

output                  0x0000000000000000000000000000000000000000000000000000000000000064
decoded input           {
                                "uint256 a": "64",
                                "uint256 b": "36"
                        }

decoded output          {
                                "0": "uint256: 100"
                        }

logs                    []

raw logs                []
```

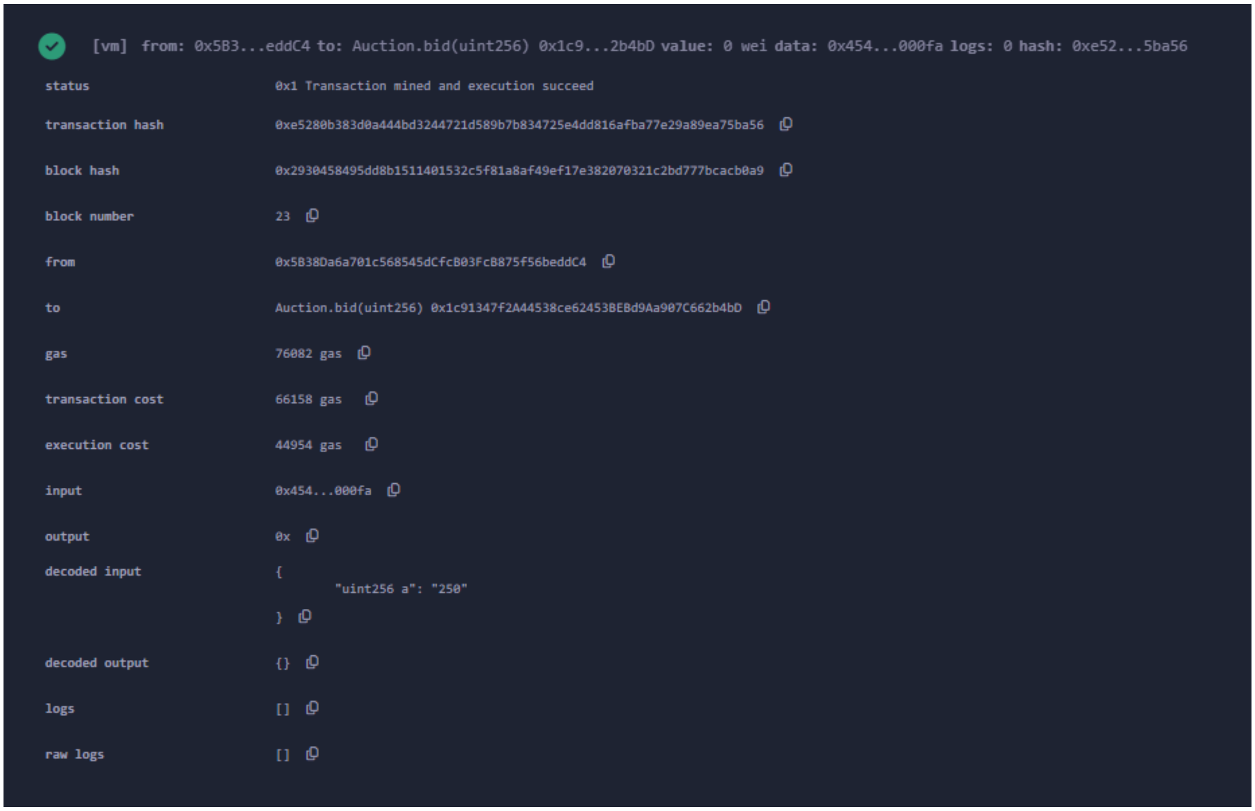
B] Aim: Write a simple auction contract when a user can bid on an item and the highest bidder wins.

Follow the same steps

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

contract Auction {
        address public o;
        uint public hb;
        address public hba;
        bool public oe;

        constructor(){
            o=msg.sender;
            hb=0;
            oe = false;

        }

    function bid(uint a) public {
        require(!oe, "Auction has already ended");
        require(a>hb, "Bid must be greater than current value");
        hb=a;
        hba=msg.sender;

    }
    function endA() public {
        require(msg.sender==o, "Only owner can send");
        oe=true;
    }
    function withdraw() public view returns (uint256){
        require(msg.sender==hba, "Only highest bidder can withdraw");
        return hb;
    }

}
```

Output:





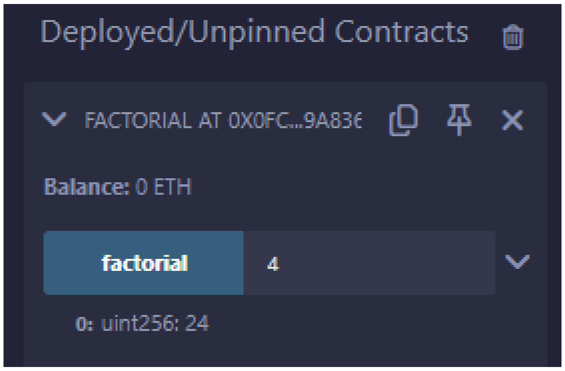
```
[vm] from: 0x5B3...eddC4 to: Auction.bid(uint256) 0x1c9...2b4bD value: 0 wei data: 0x454...000fa logs: 0 hash: 0xe52...5ba56
status              0x1 Transaction mined and execution succeed
transaction hash    0xe5280b383d0a444bd3244721d589b7b834725e4dd816afba77e29a89ea75ba56
block hash          0x2930458495dd8b1511401532c5f81a8af49ef17e382070321c2bd777bcacb0a9
block number        23
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Auction.bid(uint256) 0x1c91347f2A44538ce62453BEBd9Aa907C662b4bD
gas                 76082 gas
transaction cost    66158 gas
execution cost      44954 gas
input               0x454...000fa
output              0x
decoded input       {
                            "uint256 a": "250"
                    }
decoded output      {}
logs                []
raw logs            []
```

Practical 7

A] Aim: Write a Smart Contract for Factorial Number in Blockchain.

```
pragma solidity ^0.8.0;

contract Factorial {

    function factorial(uint256 n) public pure returns (uint256) {
        require(n >= 0, "Input must be a non-negative integer");

        if (n == 0 || n == 1) {
            return 1;
        }

        uint256 result = 1;
        for (uint256 i = 2; i <= n; i++) {
            result *= i;
        }

        return result;
    }
}
```

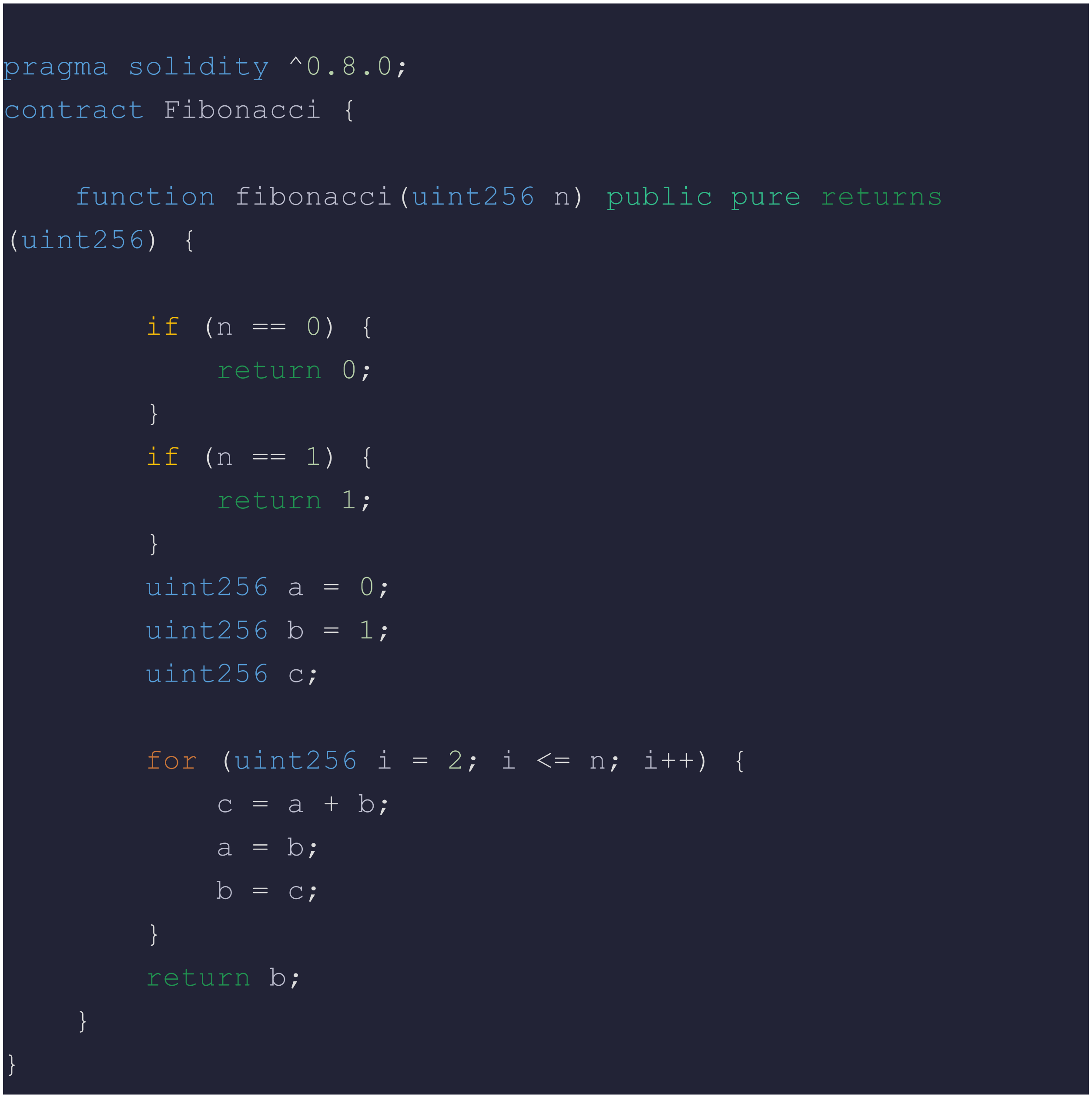
Output:



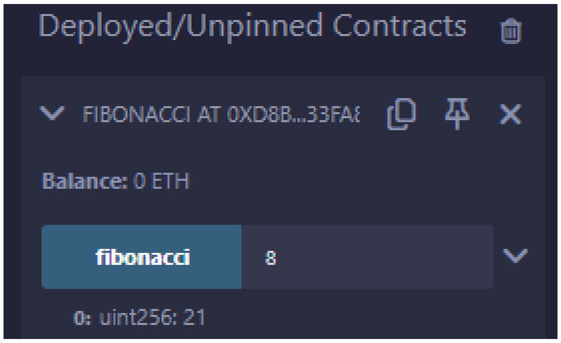
```
Balance: 0 ETH
factorial    4
0: uint256: 24
```

Name: Adiba Mohammed Raza Siddique
Seat No: 31031523033

B] Aim: Write a smart contract for the nth term of fibonacci in Blockchain.



```
pragma solidity ^0.8.0;
contract Fibonacci {

    function fibonacci(uint256 n) public pure returns
(uint256) {

        if (n == 0) {
            return 0;
        }
        if (n == 1) {
            return 1;
        }
        uint256 a = 0;
        uint256 b = 1;
        uint256 c;

        for (uint256 i = 2; i <= n; i++) {
            c = a + b;
            a = b;
            b = c;
        }
        return b;
    }
}
```



Deployed/Unpinned Contracts

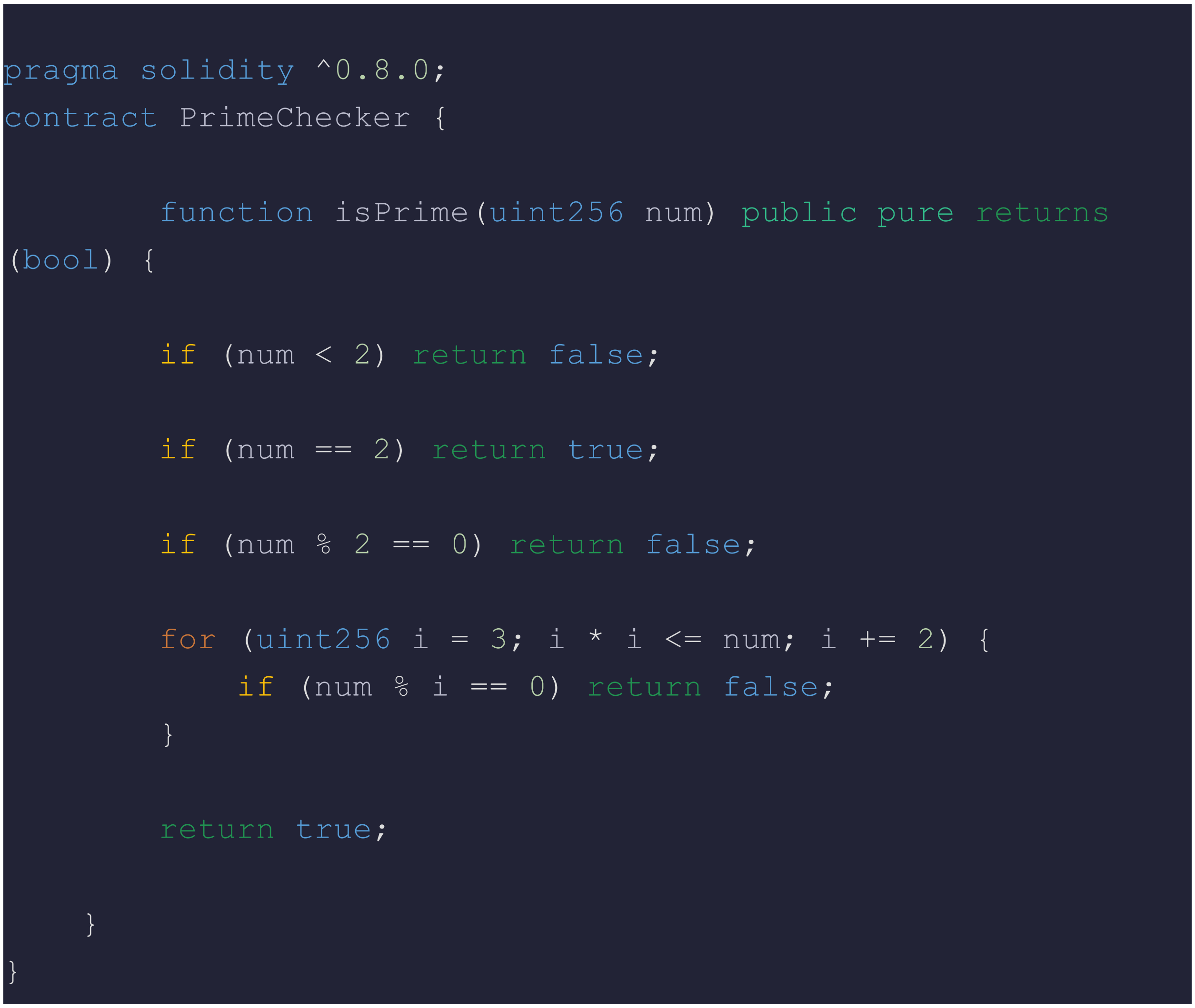
FIBONACCI AT 0XD8B...33FA8

Balance: 0 ETH

fibonacci 8

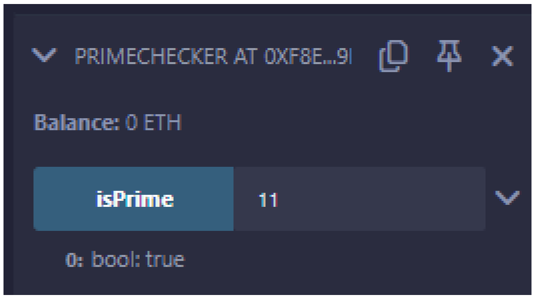
0: uint256: 21

C] Aim: Write a Smart Contract for Prime Numbers in blockchain.

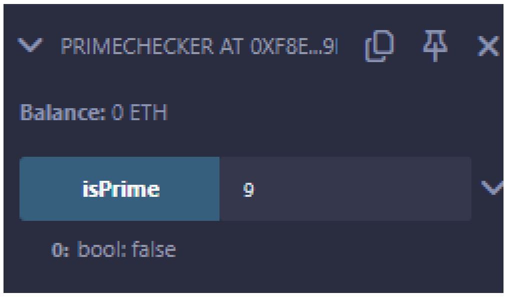


```
pragma solidity ^0.8.0;
contract PrimeChecker {

        function isPrime(uint256 num) public pure returns
(bool) {

        if (num < 2) return false;

        if (num == 2) return true;

        if (num % 2 == 0) return false;

        for (uint256 i = 3; i * i <= num; i += 2) {
            if (num % i == 0) return false;
        }

        return true;

    }
}
```

Output:

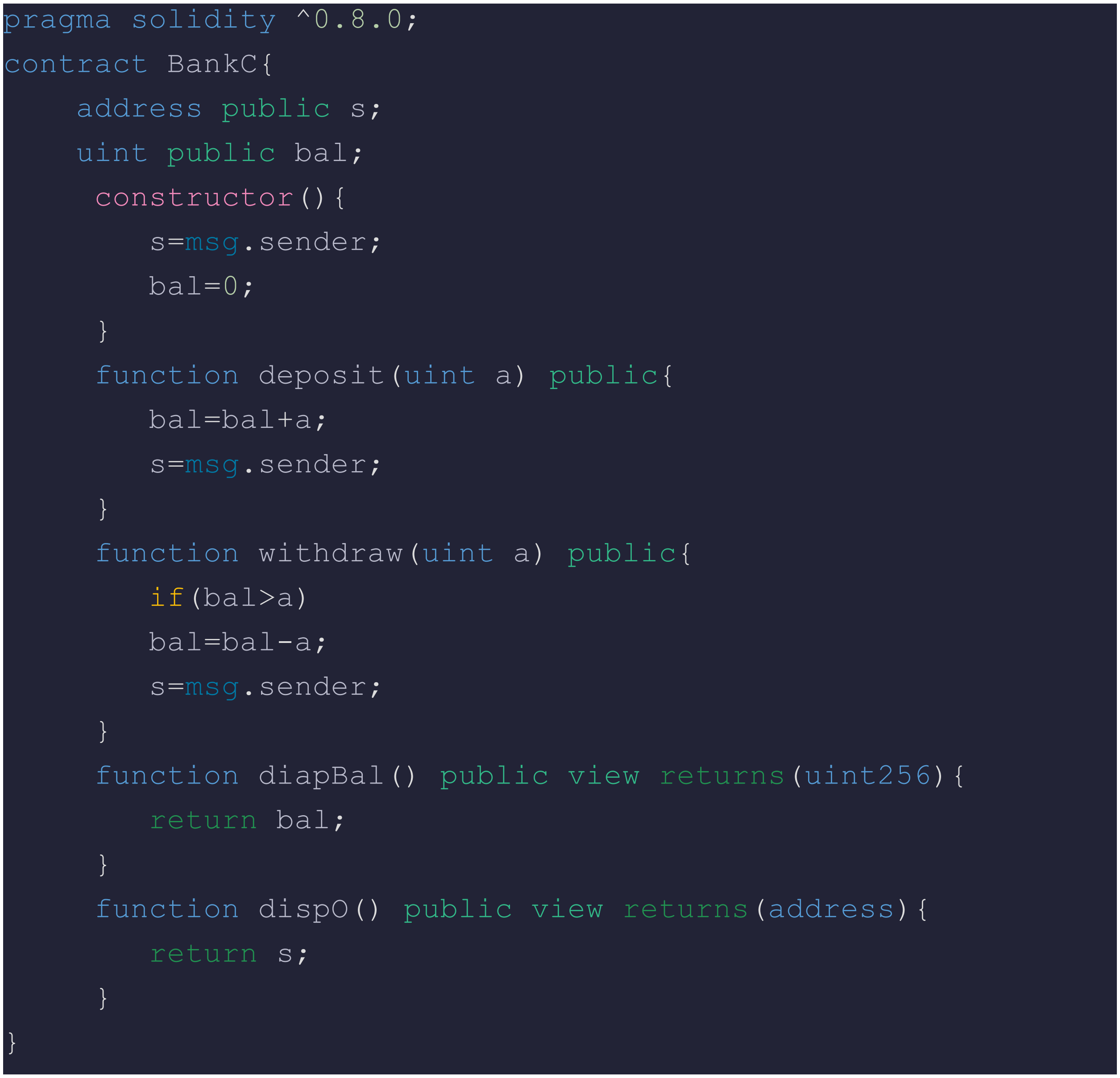


```
Balance: 0 ETH
isPrime  11
0: bool: true
```



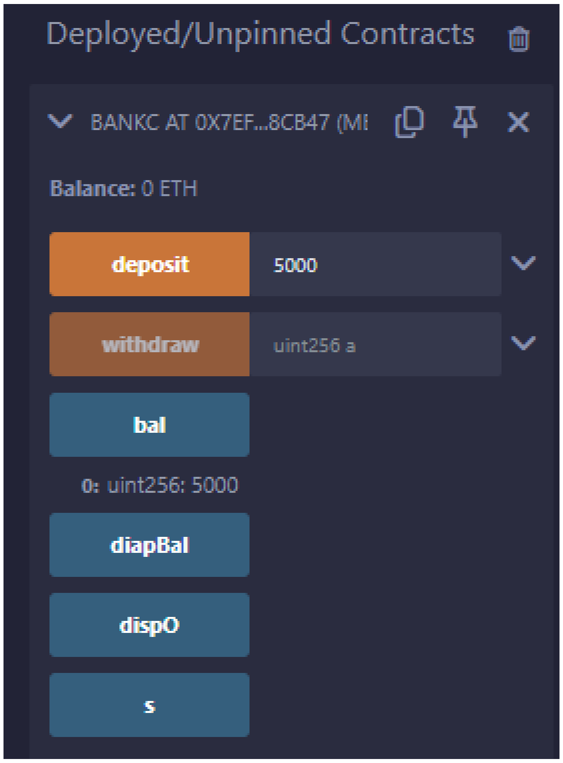
```
Balance: 0 ETH
isPrime  9
0: bool: false
```

D] Aim: Create a Smart Contract for deposit and withdrawal of money.



```
pragma solidity ^0.8.0;
contract BankC{
    address public s;
    uint public bal;
     constructor(){
        s=msg.sender;
        bal=0;
     }
     function deposit(uint a) public{
        bal=bal+a;
        s=msg.sender;
     }
     function withdraw(uint a) public{
        if(bal>a)
        bal=bal-a;
        s=msg.sender;
     }
     function diapBal() public view returns(uint256){
        return bal;
     }
     function dispO() public view returns(address){
        return s;
     }
}
```

Output:

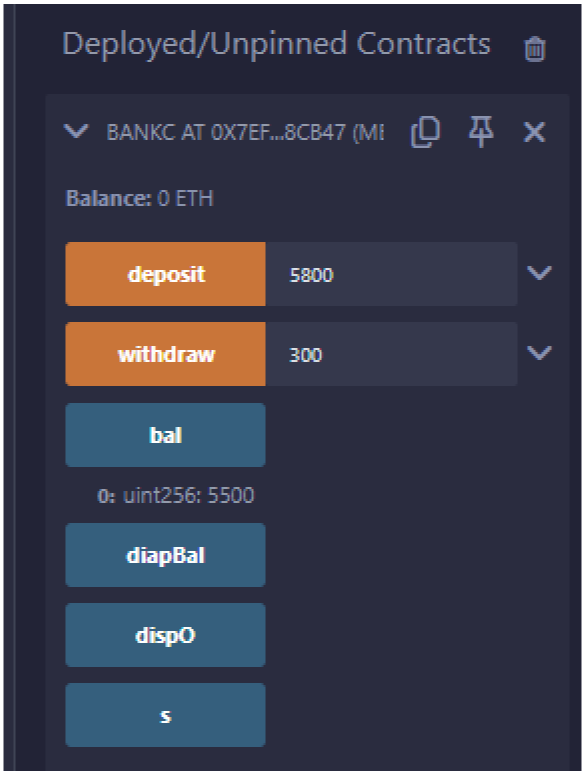


Balance: 0 ETH

deposit: 5000

withdraw: uint256 a

```
0: uint256: 5000
```



Balance: 0 ETH

deposit: 5800

withdraw: 300

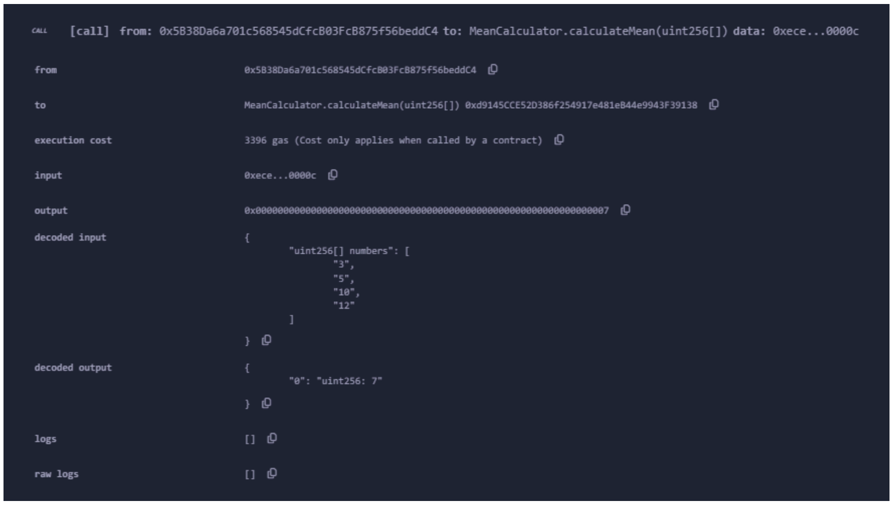
```
0: uint256: 5500
```

Practical 8

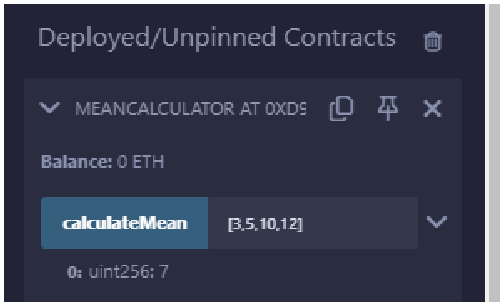
A] Aim: Create a Smart Contract to calculate the mean of n numbers.

```
pragma solidity ^0.8.0;

contract MeanCalculator {
    function calculateMean(uint[] memory numbers) public pure returns (uint) {
        require(numbers.length > 0, "The array must contain at least one number.");

        uint sum = 0;
        for (uint i = 0; i < numbers.length; i++) {
            sum += numbers[i];
        }
        uint mean = sum / numbers.length;
        return mean;
    }
}
```

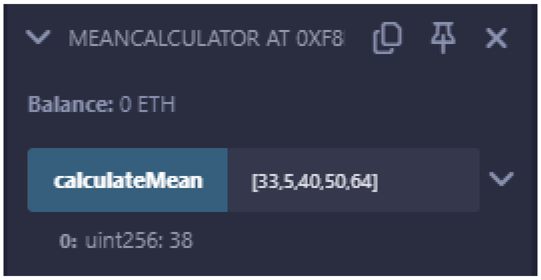
Output:



```
CALL   [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MeanCalculator.calculateMean(uint256[]) data: 0xece...0000c

from                 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to                   MeanCalculator.calculateMean(uint256[]) 0xd9145CCE52D386f254917e481eB44e9943F39138

execution cost       3396 gas (Cost only applies when called by a contract)

input                0xece...0000c

output               0x0000000000000000000000000000000000000000000000000000000000000007

decoded input        {
                             "uint256[] numbers": [
                                     "3",
                                     "5",
                                     "10",
                                     "12"
                             ]
                     }

decoded output       {
                             "0": "uint256: 7"
                     }

logs                 []

raw logs             []
```

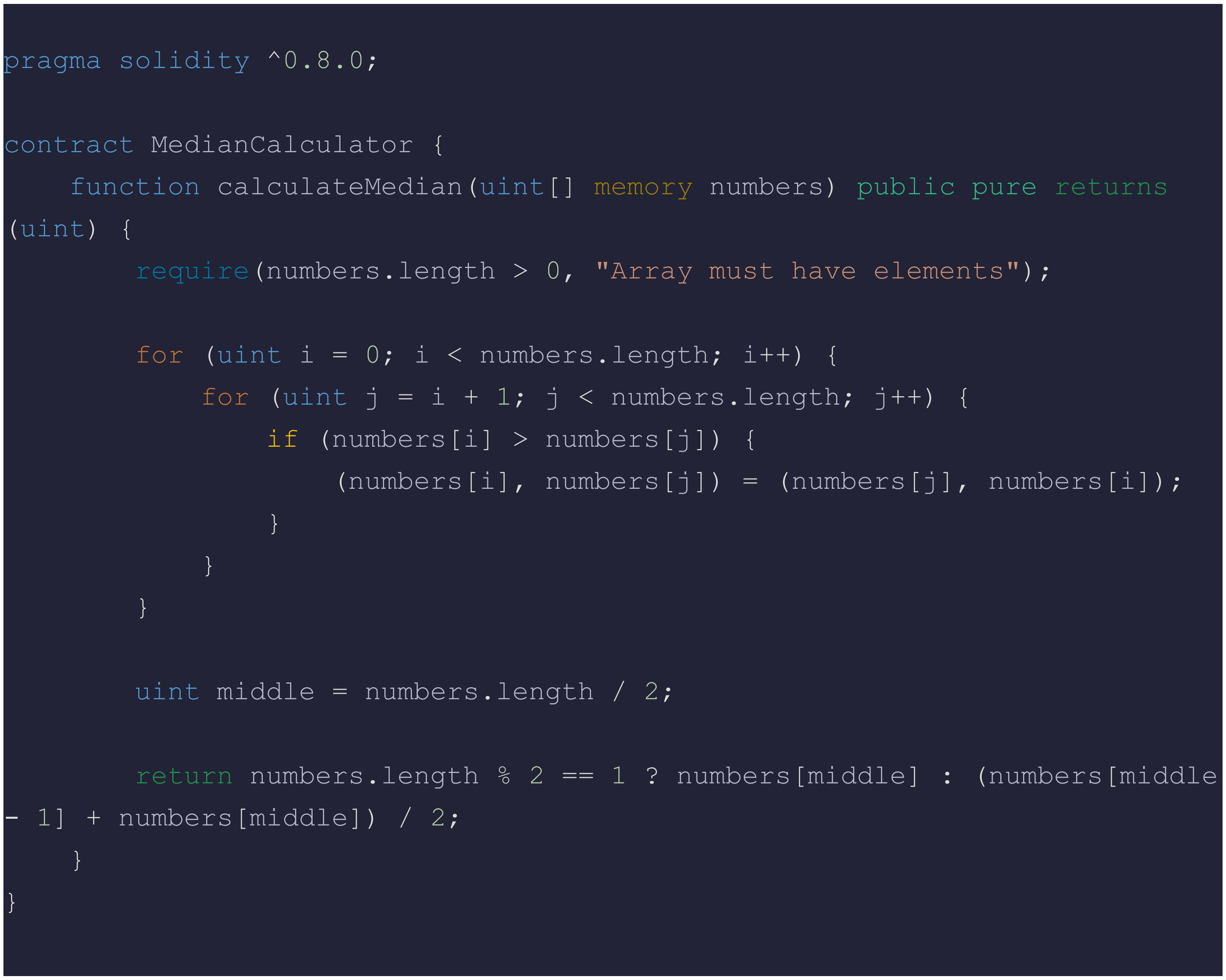


```
[3,5,10,12]
0: uint256: 7
```



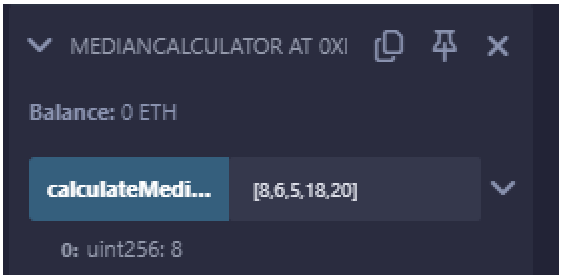
```
[33,5,40,50,64]
0: uint256: 38
```

B] Aim: Create a Smart Contract to calculate the median of n numbers.



```
pragma solidity ^0.8.0;

contract MedianCalculator {
    function calculateMedian(uint[] memory numbers) public pure returns
(uint) {
        require(numbers.length > 0, "Array must have elements");

        for (uint i = 0; i < numbers.length; i++) {
            for (uint j = i + 1; j < numbers.length; j++) {
                if (numbers[i] > numbers[j]) {
                    (numbers[i], numbers[j]) = (numbers[j], numbers[i]);
                }
            }
        }

        uint middle = numbers.length / 2;

        return numbers.length % 2 == 1 ? numbers[middle] : (numbers[middle
- 1] + numbers[middle]) / 2;
    }
}
```

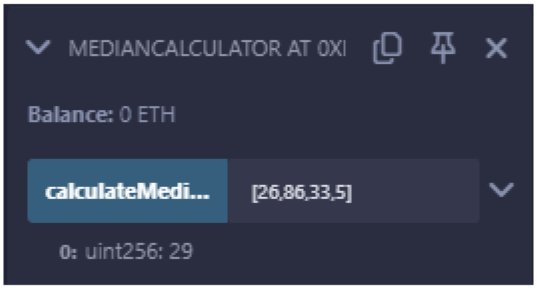
Output:



Balance: 0 ETH

[8,6,5,18,20]

0: uint256: 8



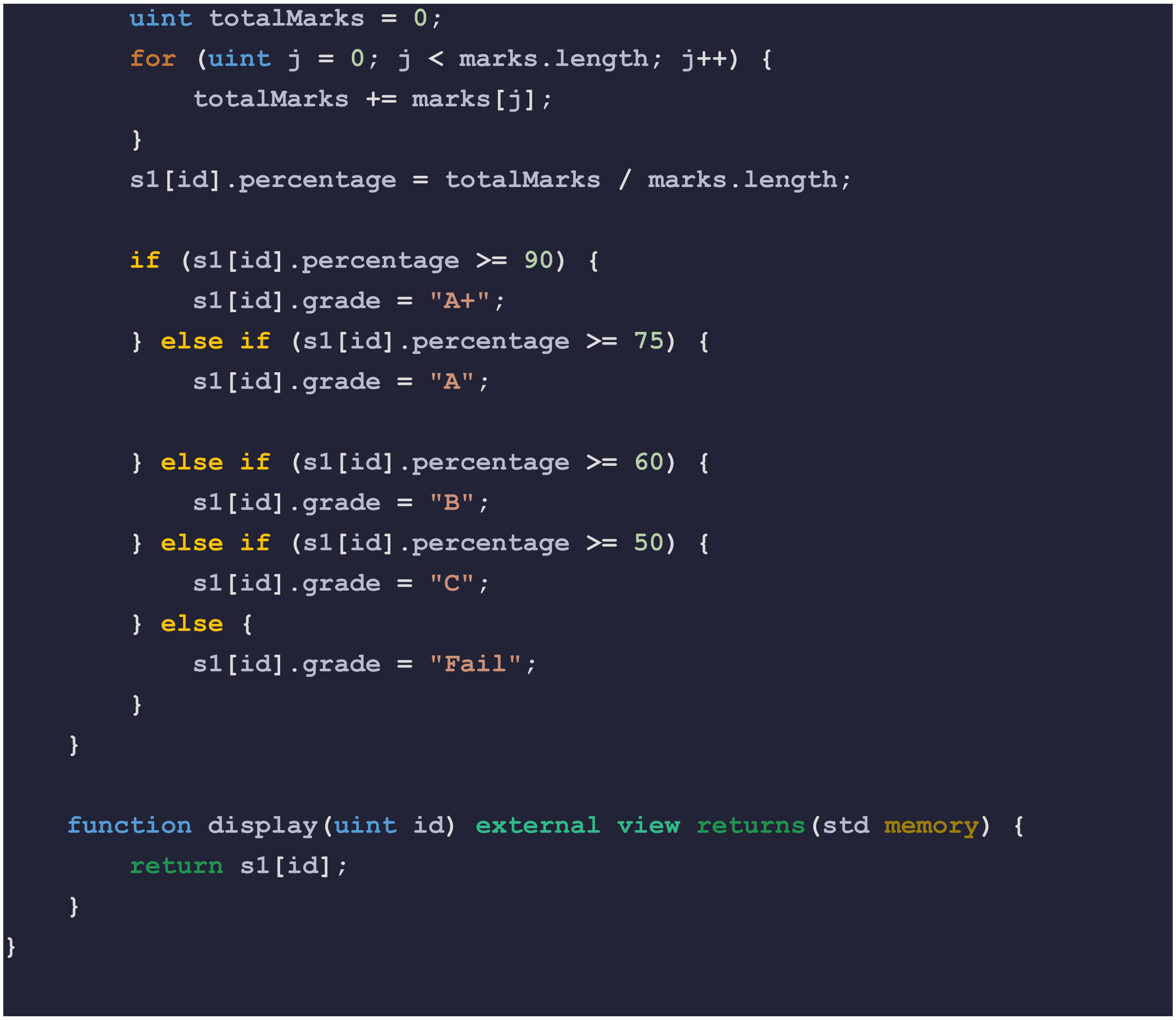
Balance: 0 ETH

[26,86,33,5]

0: uint256: 29

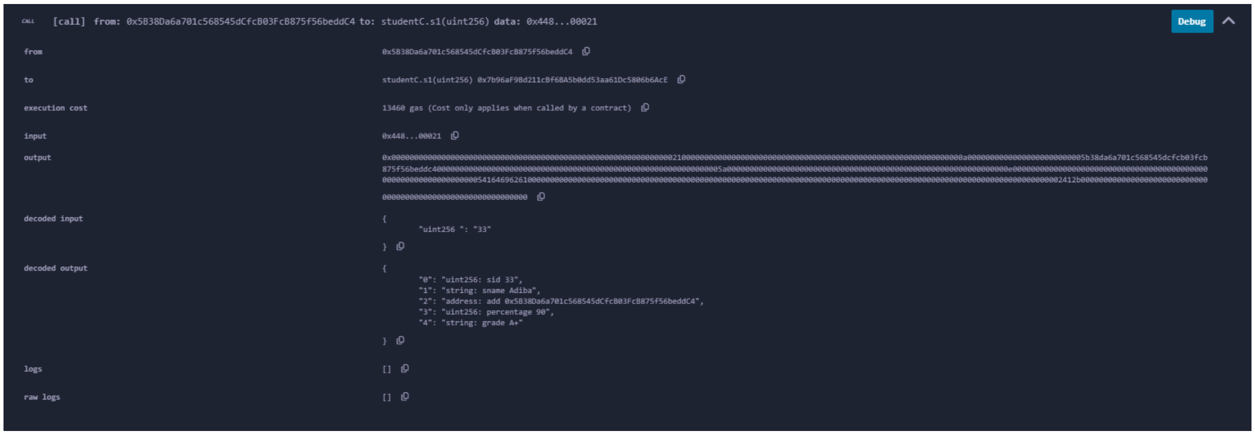
C] Aim: Create a Student Blockchain to register a new student and display the same.

```
pragma solidity ^0.8.0;

contract studentC {

    struct std {
        uint sid;
        string sname;
        address add;
        uint[5] marks;
        uint percentage;
        string grade;
    }

    mapping(uint => std) public s1;

    function registerS(uint i, string memory n) public {
        s1[i].sname = n;
        s1[i].add = msg.sender;
        s1[i].sid = i;
    }

    function addMarks(uint id, uint[5] memory marks) public {
        require(s1[id].sid != 0, "Student not found");

        s1[id].marks = marks;
```

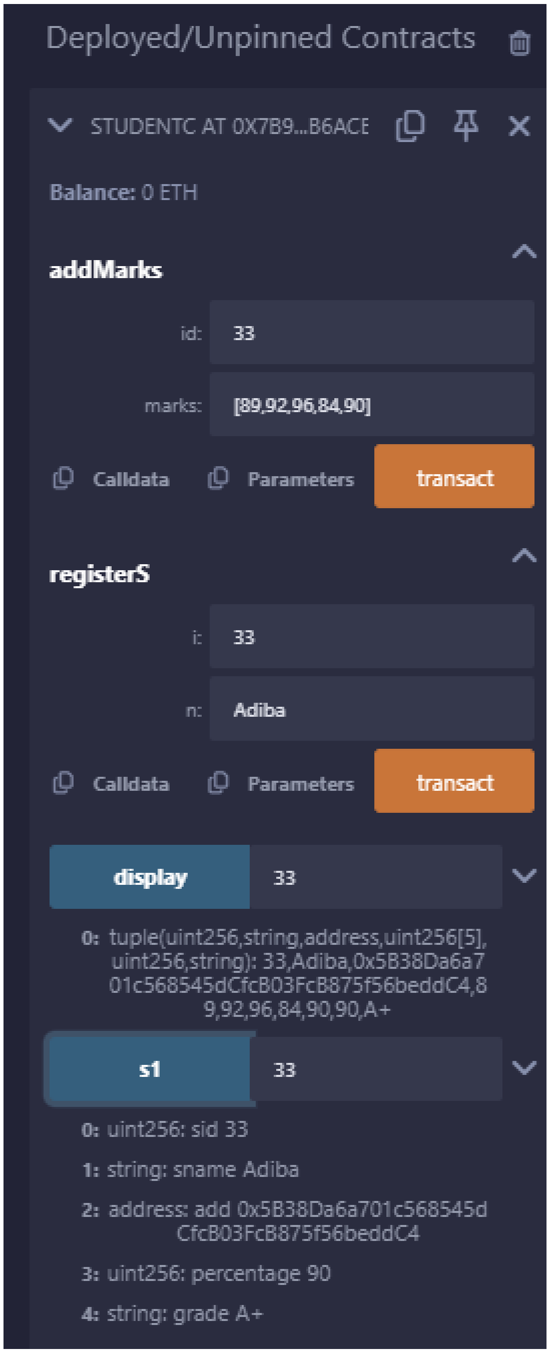


```
        uint totalMarks = 0;
        for (uint j = 0; j < marks.length; j++) {
            totalMarks += marks[j];
        }
        s1[id].percentage = totalMarks / marks.length;

        if (s1[id].percentage >= 90) {
            s1[id].grade = "A+";
        } else if (s1[id].percentage >= 75) {
            s1[id].grade = "A";

        } else if (s1[id].percentage >= 60) {
            s1[id].grade = "B";
        } else if (s1[id].percentage >= 50) {
            s1[id].grade = "C";
        } else {
            s1[id].grade = "Fail";
        }
    }

    function display(uint id) external view returns(std memory) {
        return s1[id];
    }
}
```

Output:



```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: studentC.s1(uint256) data: 0x448...00021

from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to                  studentC.s1(uint256) 0x7b96aF9Bd211cBf6BA5b0dd53aa61Dc5806b6AcE

execution cost      13460 gas (Cost only applies when called by a contract)

input               0x448...00021

output              0x000000000000000000000000000000000000000000000000000000000000002100000000000000000000000000000000000000000000000000000000000000a00000000000000000000000005b38da6a701c568545dcfcb03fcb
                    875f56beddc400000000000000000000000000000000000000000000000000000000000005a00000000000000000000000000000000000000000000000000000000000000e0000000000000000000000000000000000000000000
                    0000000000000000000005416469626100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002412b0000000000000000000000000000

                    00000000000000000000000000000000

decoded input       {
                          "uint256 ": "33"
                    }

decoded output      {
                          "0": "uint256: sid 33",
                          "1": "string: sname Adiba",
                          "2": "address: add 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                          "3": "uint256: percentage 90",
                          "4": "string: grade A+"
                    }

logs                []

raw logs            []
```



Balance: 0 ETH

addMarks

id: 33

marks: [89,92,96,84,90]

registerS

i: 33

n: Adiba

display 33

```
0: tuple(uint256,string,address,uint256[5],
   uint256,string): 33,Adiba,0x5B38Da6a7
   01c568545dCfcB03FcB875f56beddC4,8
   9,92,96,84,90,90,A+
```

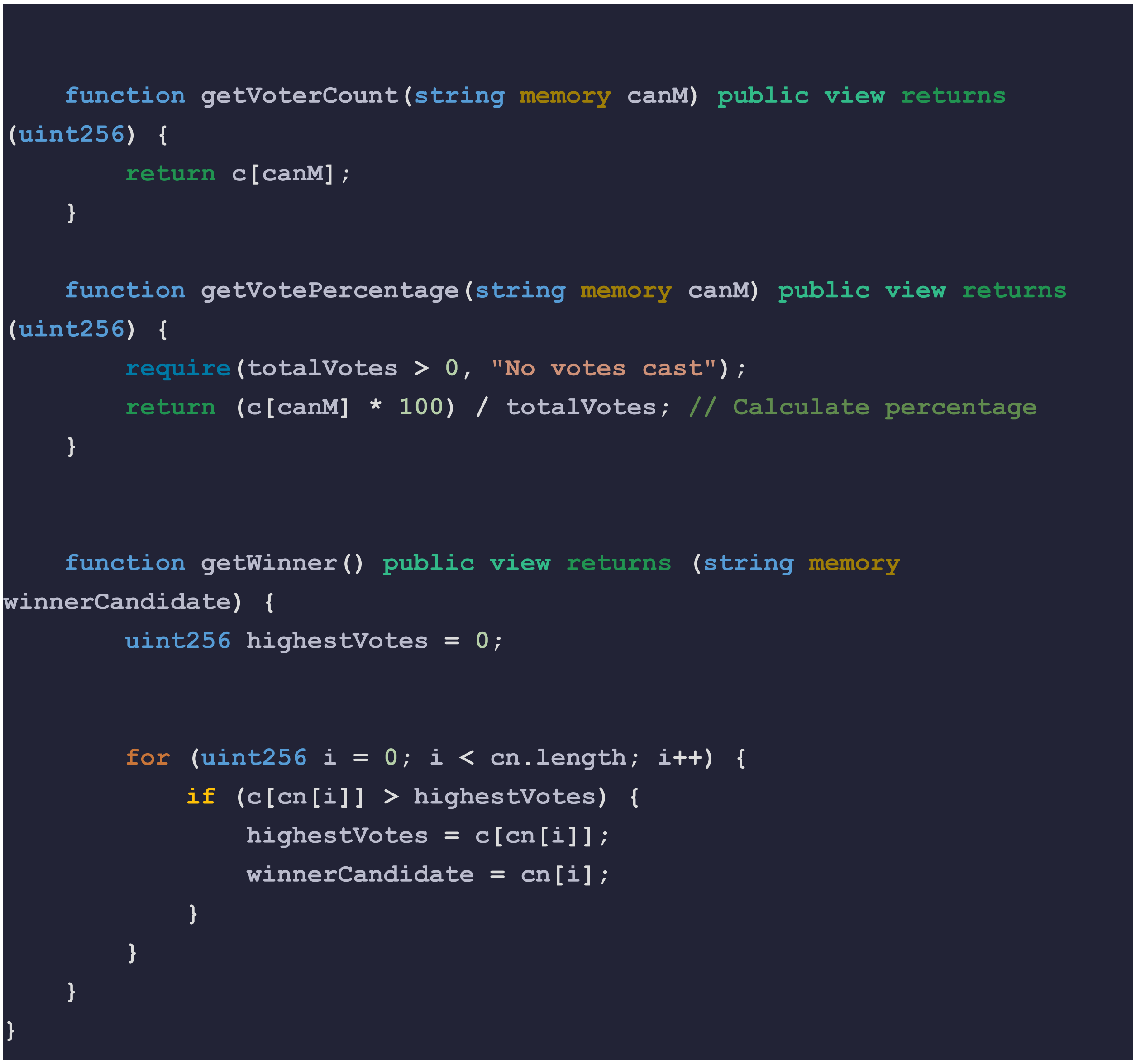
s1 33

```
0: uint256: sid 33
1: string: sname Adiba
2: address: add 0x5B38Da6a701c568545d
   CfcB03FcB875f56beddC4
3: uint256: percentage 90
4: string: grade A+
```

Practical 9

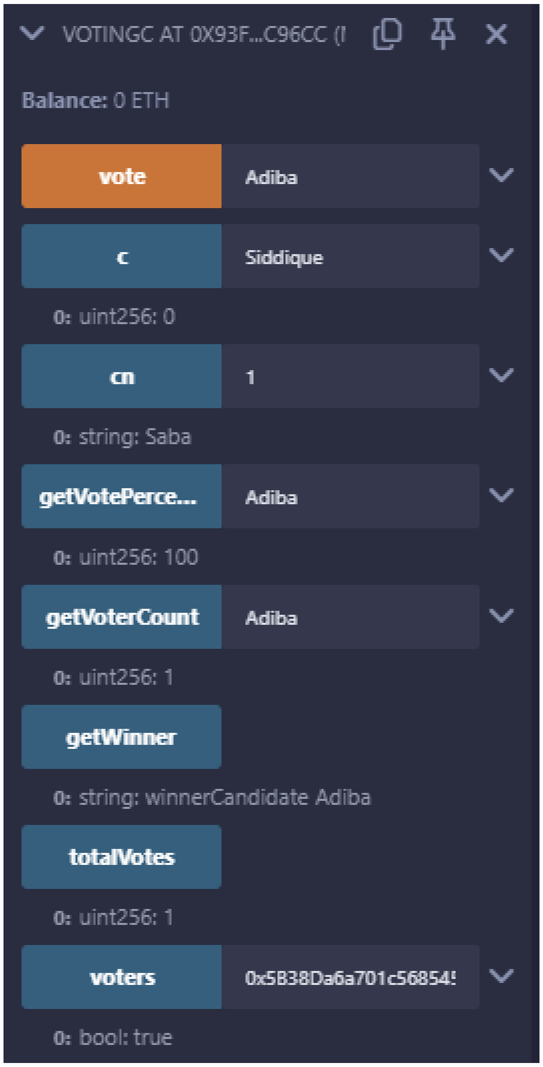
Aim: Create a Smart Contract for Voting Application.

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


contract VotingC {
    mapping(address => bool) public voters;
    mapping(string => uint256) public c;
    string[] public cn;
    uint256 public totalVotes;


    constructor(string[] memory candN) {
        cn = candN;
    }


    function vote(string memory caNm) public {
        require(!voters[msg.sender], "Already voted");
        bool exists = false;


        for (uint256 i = 0; i < cn.length; i++) {
            if (keccak256(bytes(caNm)) == keccak256(bytes(cn[i]))) {
                exists = true;
                break;
            }
        }
        require(exists, "Candidate does not exist");


        c[caNm]++;
        voters[msg.sender] = true;
        totalVotes++; // Increment total votes
    }
```



```
    function getVoterCount(string memory canM) public view returns
(uint256) {
        return c[canM];
    }

    function getVotePercentage(string memory canM) public view returns
(uint256) {
        require(totalVotes > 0, "No votes cast");
        return (c[canM] * 100) / totalVotes; // Calculate percentage
    }


    function getWinner() public view returns (string memory
winnerCandidate) {
        uint256 highestVotes = 0;


        for (uint256 i = 0; i < cn.length; i++) {
            if (c[cn[i]] > highestVotes) {
                highestVotes = c[cn[i]];
                winnerCandidate = cn[i];
            }
        }
    }
}
```

Output:



```
VOTINGC AT 0X93F...C96CC (I

Balance: 0 ETH

vote            Adiba
c               Siddique
  0: uint256: 0
cn              1
  0: string: Saba
getVotePerce... Adiba
  0: uint256: 100
getVoterCount   Adiba
  0: uint256: 1
getWinner
  0: string: winnerCandidate Adiba
totalVotes
  0: uint256: 1
voters          0x5B38Da6a701c56854!
  0: bool: true
```

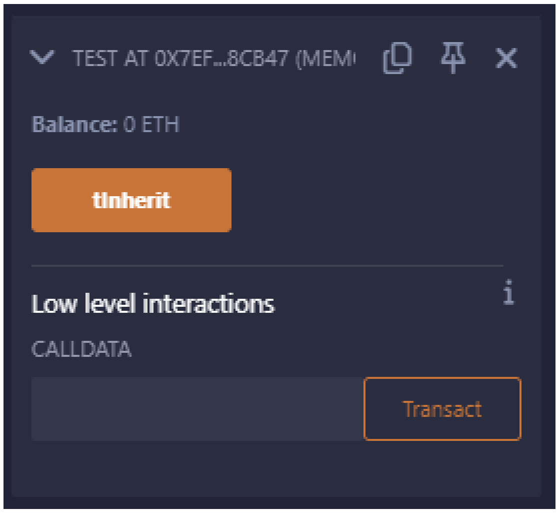
Practical 10

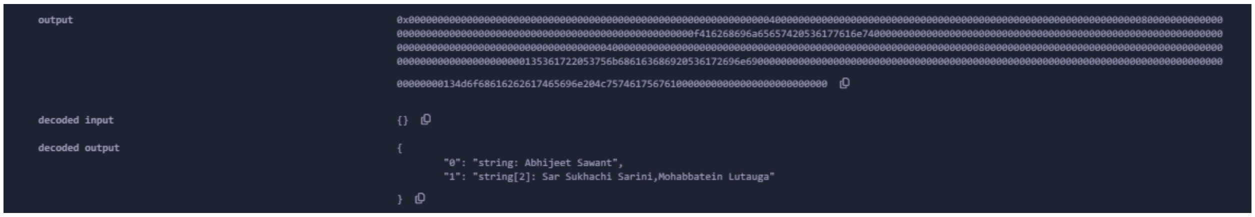
A] Aim: Write a smart contract for single Inheritance

Code:

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

pragma solidity ^0.8.0;

contract singer{
    string n;
    string[2] so;
    function setN(string memory a, string[2] memory arr) public {
        n = a;
        so = arr;
    }
}

contract song is singer{
    function getVal() public view returns (string memory, string[2]
memory){
        return (n, so);
    }
}

contract test{
    song s = new song();
    function tInherit() public returns(string memory, string[2] memory){
        s.setN("Abhijeet Sawant", ["Sar Sukhachi Sarini", "Mohabbatein
Lutauga"]);
        return s.getVal();
    }
}
```

Output:





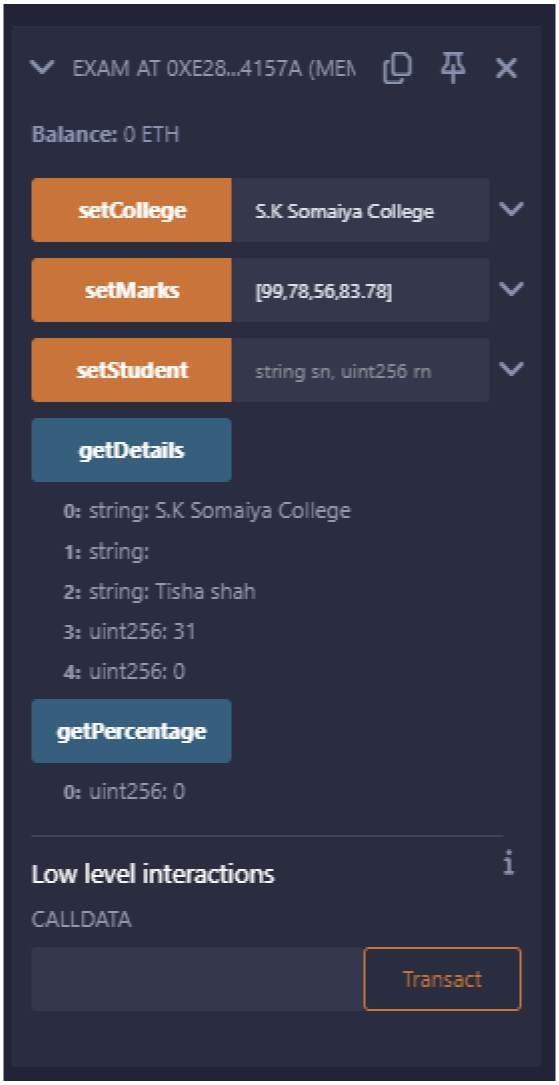
```
output          0x000000000000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000800000000000000
                000000000000000000000000000000000000000000000000000f416268696a65657420536177616e74000000000000000000000000000000000000000000000000000000000000
                0000000000000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000800000000000000000000000000000000000000000
                000000000000000000000013536172205375 6b686163686920536172696e6900000000000000000000000000000000000000000000000000000000000000000000000000000000

                00000000134d6f68616262617465696e204c75746175676100000000000000000000000000

decoded input   {}

decoded output  {
                        "0": "string: Abhijeet Sawant",
                        "1": "string[2]: Sar Sukhachi Sarini,Mohabbatein Lutauga"
                }
```

B] Aim: Write a smart contract for multi-level Inheritance

Code:

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

pragma solidity ^0.8.0;

contract college {
    string internal cname;
    string internal pname;

    function setCollege(string memory cn, string memory pn) public {
        cname = cn;
        pname = pn;
    }
}

contract student is college {
    string internal sname;
    uint internal rollno;

    function setStudent(string memory sn, uint rn) public {
        sname = sn;
        rollno = rn;
    }
}

contract exam is student {
    uint8[5] marks;

    function setMarks(uint8[5] memory m) public {
        marks = m;
    }

    function getPercentage() public view returns(uint) {

    }

    function getDetails() public view returns(string memory, string
memory, string memory, uint, uint) {
        uint total = 0;
        for(uint i = 0; i < 5; i++) {
            total += marks[i];
        }
        uint per = total/5;
        return (cname, pname, sname, rollno, per);
    }
}
```

Output:



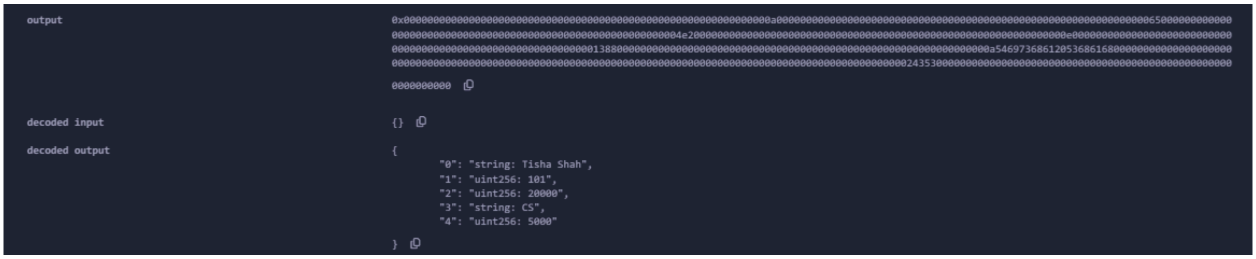
```
Balance: 0 ETH
setCollege    S.K Somaiya College
setMarks      [99,78,56,83,78]
setStudent    string sn, uint256 rn
getDetails
0: string: S.K Somaiya College
1: string:
2: string: Tisha shah
3: uint256: 31
4: uint256: 0
getPercentage
0: uint256: 0
```

C] Aim: Write a smart contract for multiple level Inheritance

Code:

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

pragma solidity ^0.8.0;

contract employee{
    string n;
    uint mid;
    uint sal;
    function setE(string memory a, uint b, uint c) public {
        n = a;
        mid = b;
        sal = c;
    }
}

contract department{
    string dep;
    function setD(string memory a) public {
        dep = a;
    }
}

contract salary is employee, department{
    uint HRA;
    function calHRA() public returns(uint){
        if (sal >= 15000){
            HRA = 5000;
        }
        else if(sal >= 25000){
            HRA = 10000;
        }
        else{
            HRA = 2000;
        }
        return HRA;
    }
    function getVal() public view returns (string memory, uint, uint,
string memory, uint){
        return (n, mid, sal, dep, HRA);
    }
}

contract test{
    salary s = new salary();
    function tInherit() public returns(string memory, uint, uint, string
memory, uint){
        s.setE("Tisha Shah", 101, 20000);
        s.setD("CS");
        s.calHRA();
        return s.getVal();
    }
}
```

Output:



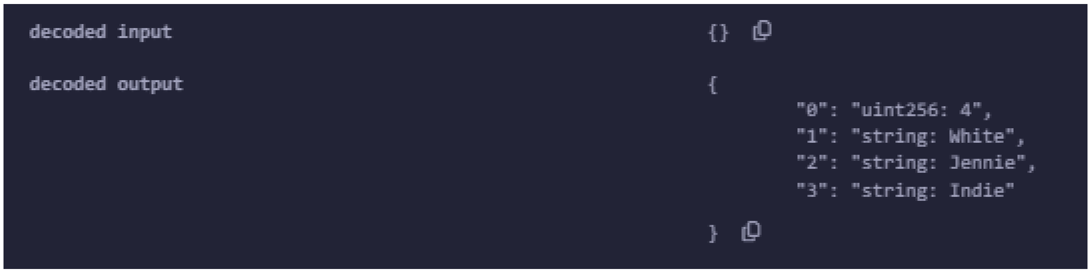
```
output          0x00000000000000000000000000000000000000000000000000000000000000a000000000000000000000000000000000000000000000000000000000000000650000000000000
                000000000000000000000000000000000000000000000000004e2000000000000000000000000000000000000000000000000000000000000000e0000000000000000000000000000
                000000000000000000000000000000000013880000000000000000000000000000000000000000000000000000000000000000a5469736861205368616800000000000000000000
                000000000000000000000000000000000000000000000000000000000000000000000000000000000000000243530000000000000000000000000000000000000000000000000
                0000000000

decoded input   {}

decoded output  {
                        "0": "string: Tisha Shah",
                        "1": "uint256: 101",
                        "2": "uint256: 20000",
                        "3": "string: CS",
                        "4": "uint256: 5000"
                }
```

D] Aim: Write a smart contract for hierarchical Inheritance

Code:

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

pragma solidity ^0.8.0;

contract animal{
    uint legs;
    string color;
    function setA(uint a, string memory b) public {
        legs = a;
        color = b;
    }

}

contract dog is animal{
    string name;
    string species;
    function setVal(string memory a, string memory b) public {
        name = a;
        species = b;
    }

    function getVal() public view returns (uint, string memory, string
memory, string memory){
        return (legs, color, name, species);
    }
}

contract cat is animal{
    string name;
    string species;
    function setVal(string memory a, string memory b) public {
        name = a;
        species = b;
    }
    function getVal() public view returns (uint, string memory, string memory, string memory){
        return (legs, color, name, species);
    }
}

contract test{
    dog d = new dog();
    cat c = new cat();
    function dInherit() public returns(uint, string memory, string memory, string memory){
        d.setA(4, "Black");
        d.setVal("Simba", "Labrador");
        return d.getVal();
    }

    function cInherit() public returns(uint, string memory, string memory, string memory){
        c.setA(4, "White");
        c.setVal("Jennie", "Indie");

        return c.getVal();
    }
}
```

Output:



```
decoded input                                    {}

decoded output                                   {
                                                         "0": "uint256: 4",
                                                         "1": "string: White",
                                                         "2": "string: Jennie",
                                                         "3": "string: Indie"
                                                 }
```

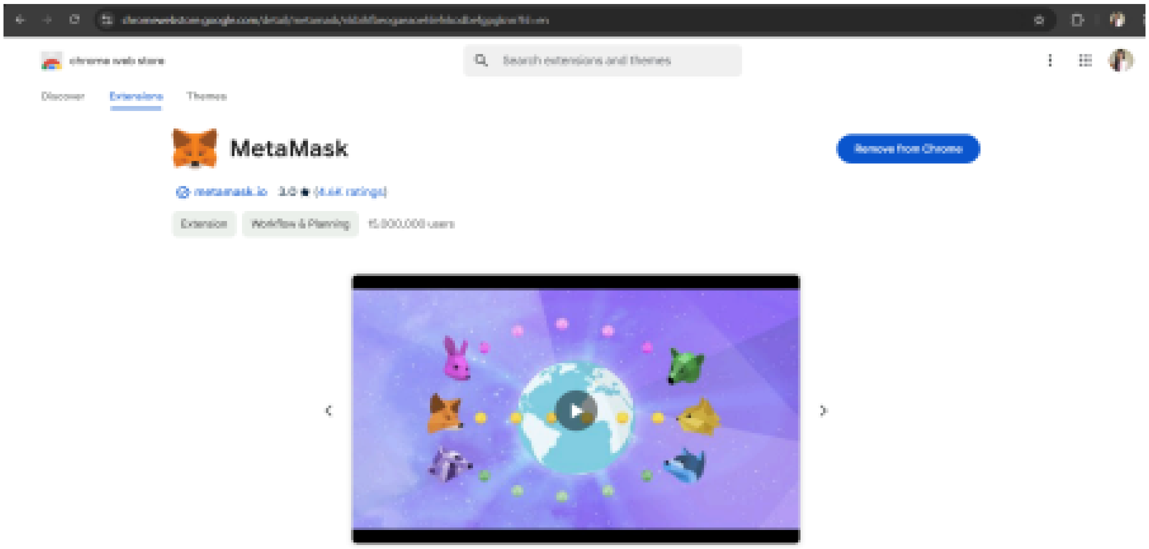
Practical 11

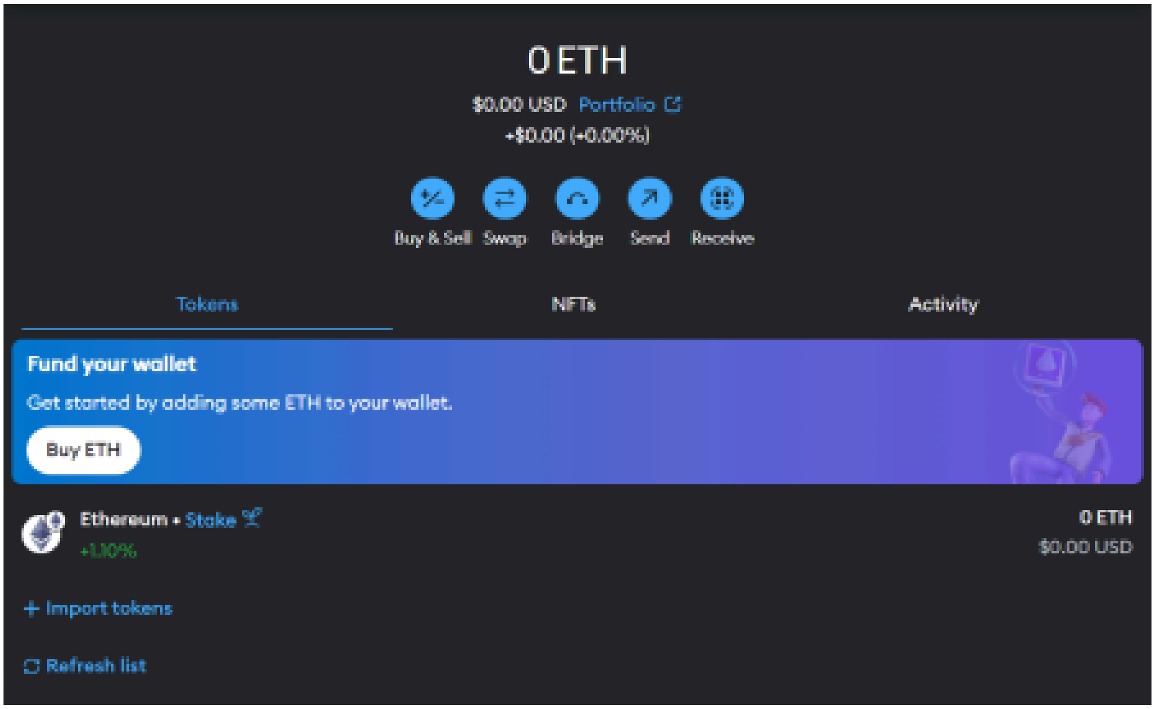
Aim: Creating a simple DApp for addition of two numbers.

Step 1: Setting up MetaMask and Ganache.

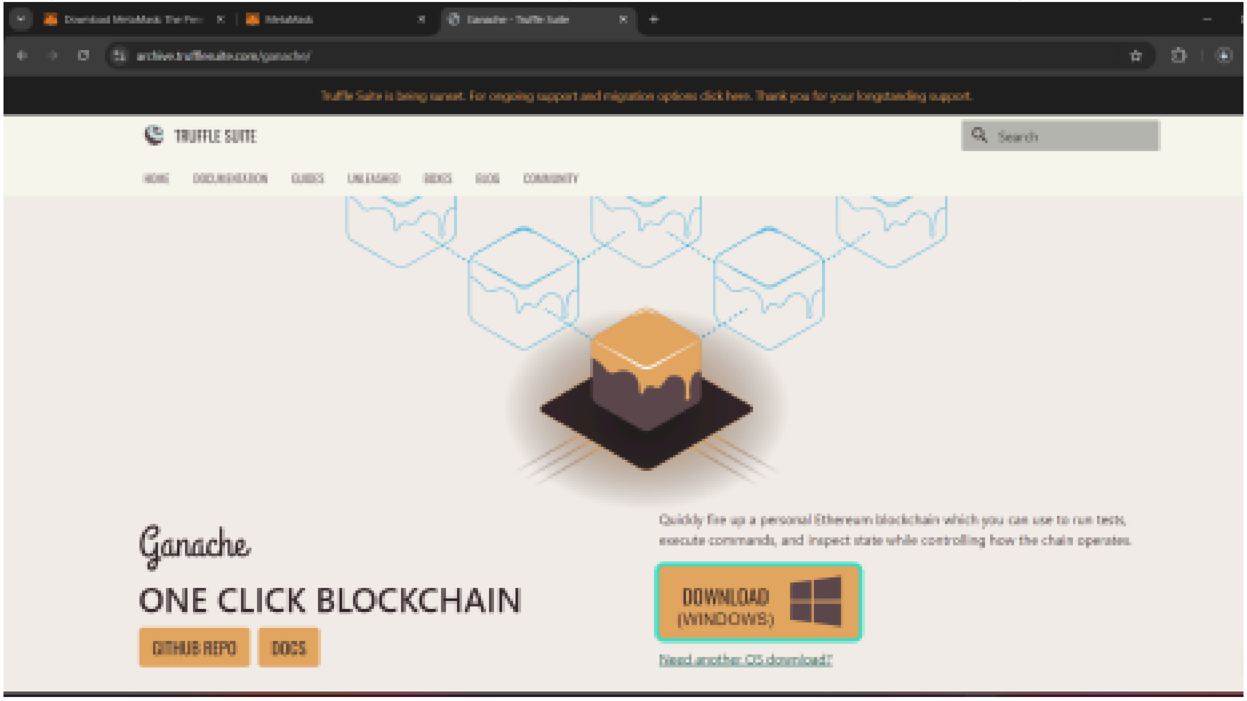
Install MetaMask Extension from here and create an account.

[https://chromewebstore.google.com/detail/metamask/nkbihfbeogaeaoehlefnkod befgpgknn?hl=en](https://chromewebstore.google.com/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en)



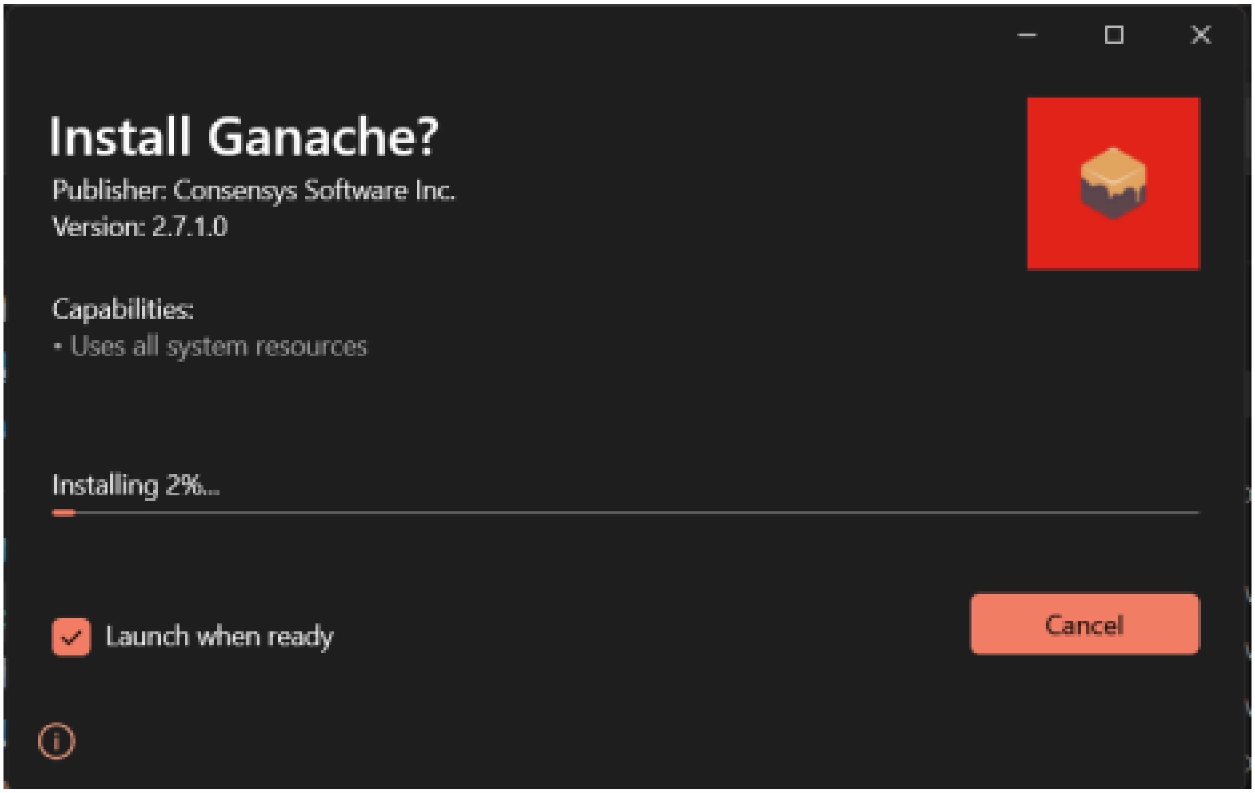


Step 2: Install Ganache from here. https://archive.trufflesuite.com/ganache/



Truffle Suite is being sunset. For ongoing support and migration options click here. Thank you for your longstanding support.

Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.



Install Ganache?

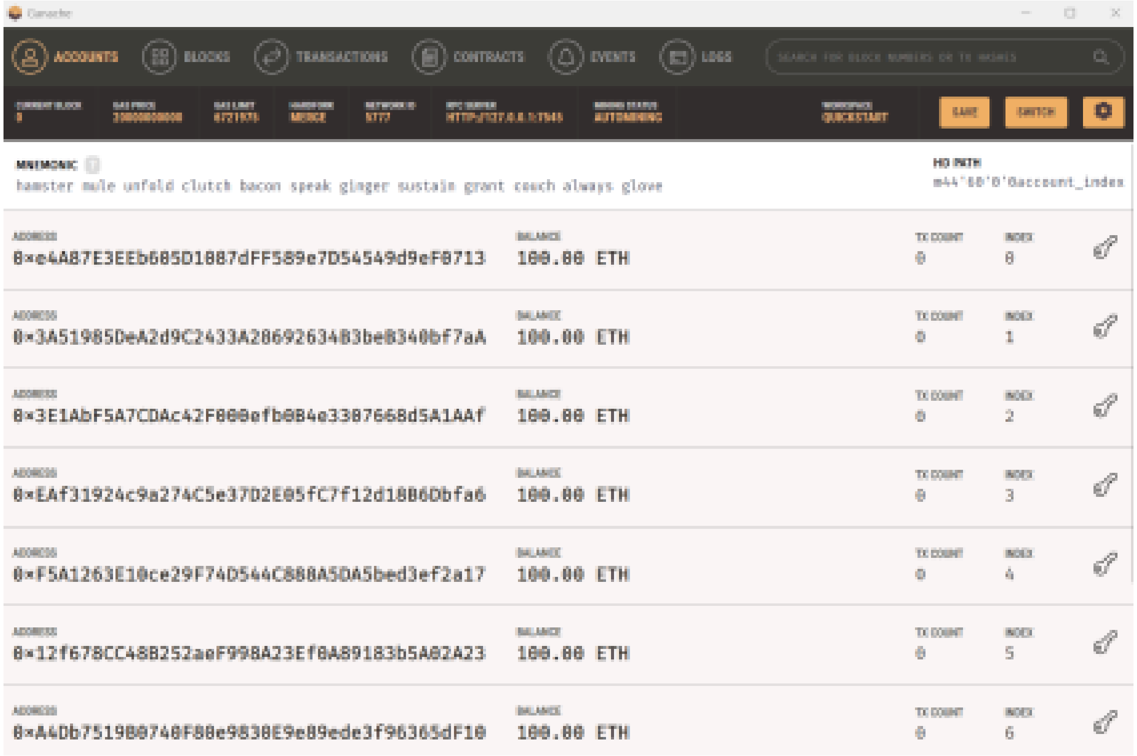
Publisher: Consensys Software Inc.
Version: 2.7.1.0

Capabilities:
- Uses all system resources

Installing 2%...

After install open it and click on **QUICKSTART**

Step 3: Now Copy the key of any id.



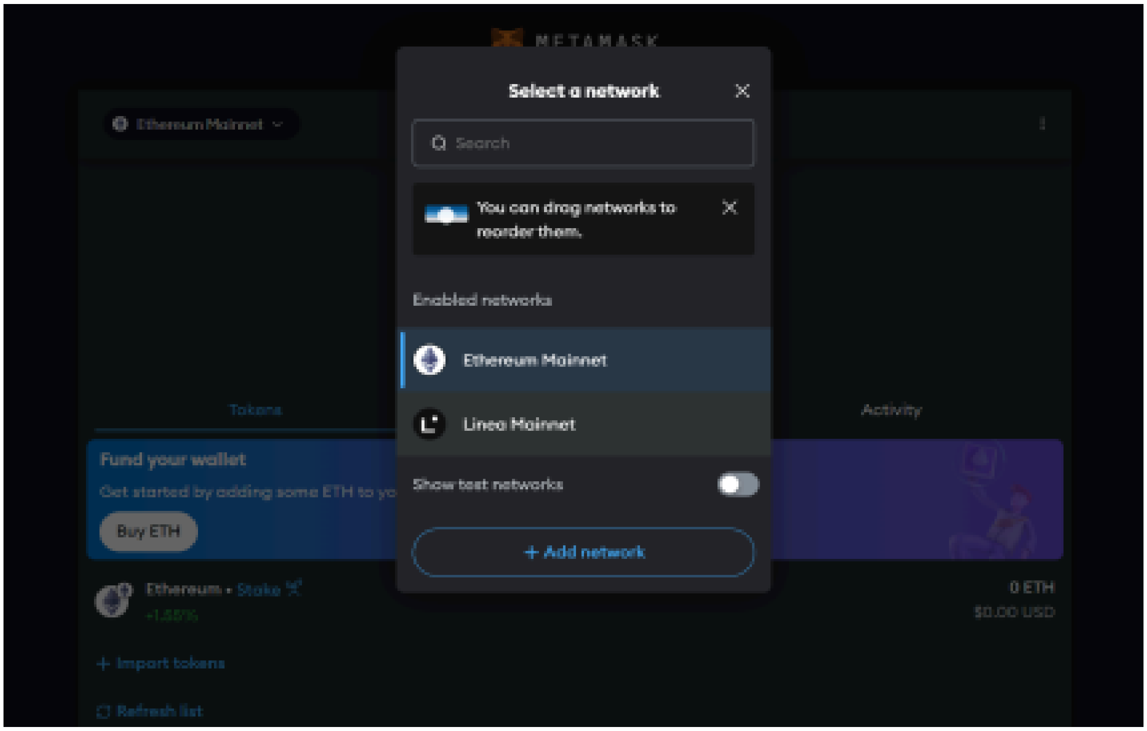
CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
0	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

MNEMONIC
hamster mule unfold clutch bacon speak ginger sustain grant couch always glove

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×e4A87E3EEb685D1887dFF589e7D54549d9eF0713	100.00 ETH	0	0
0×3A51985DeA2d9C2433A28692634B3be8340bf7aA	100.00 ETH	0	1
0×3E1AbF5A7CDAc42F000efb0B4e3387668d5A1AAf	100.00 ETH	0	2
0×EAf31924c9a274C5e37D2E05fC7f12d18B6Dbfa6	100.00 ETH	0	3
0×F5A1263E10ce29F74D544C888A5DA5bed3ef2a17	100.00 ETH	0	4
0×12f678CC48B252aeF998A23Ef0A89183b5A82A23	100.00 ETH	0	5
0×A4Db7519B0740F80e9838E9e89ede3f96365dF10	100.00 ETH	0	6

Step 4: Now add the network in metamask.



Fund your wallet
Get started by adding some ETH to yo

Select a network

Search

You can drag networks to reorder them.

Enabled networks

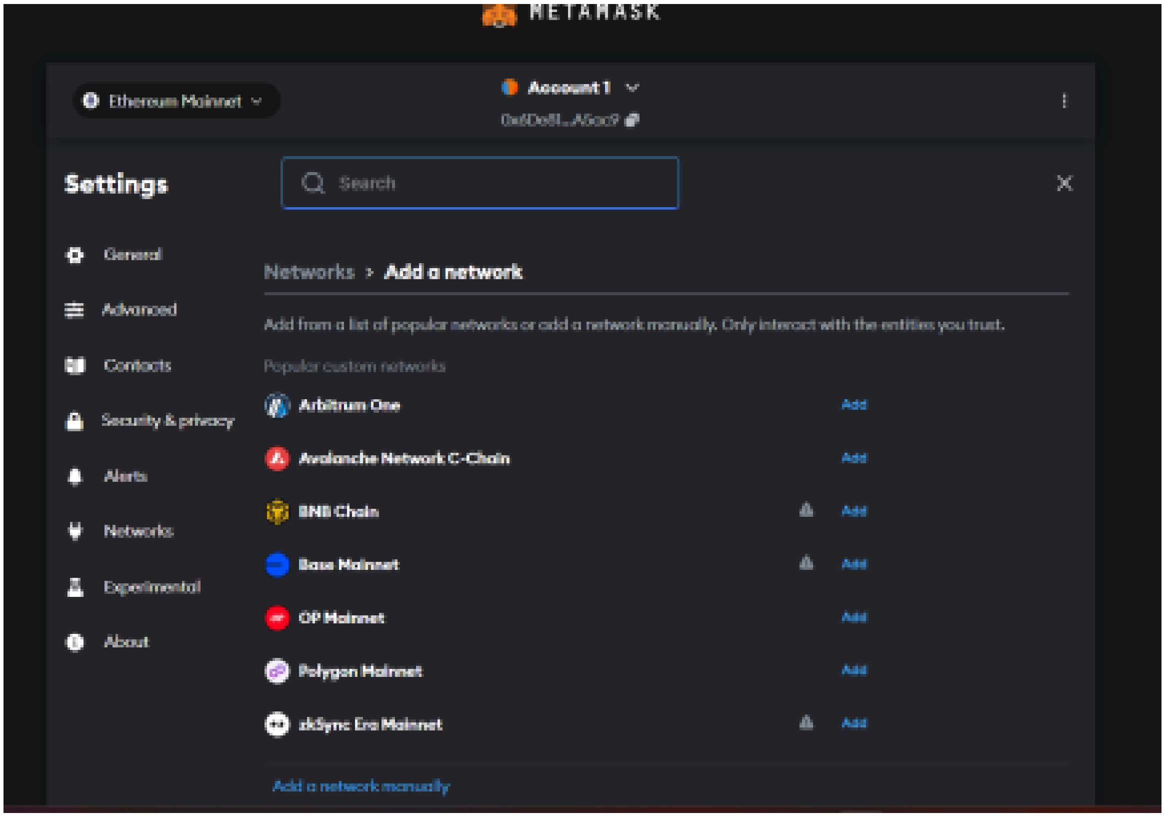
Ethereum Mainnet

Linea Mainnet

Show test networks

+ Add network

Click on add network.



METAMASK

Ethereum Mainnet

Account 1
0x6De81...A5ac9

Settings

Search

- General
- Advanced
- Contacts
- Security & privacy
- Alerts
- Networks
- Experimental
- About

Networks > Add a network

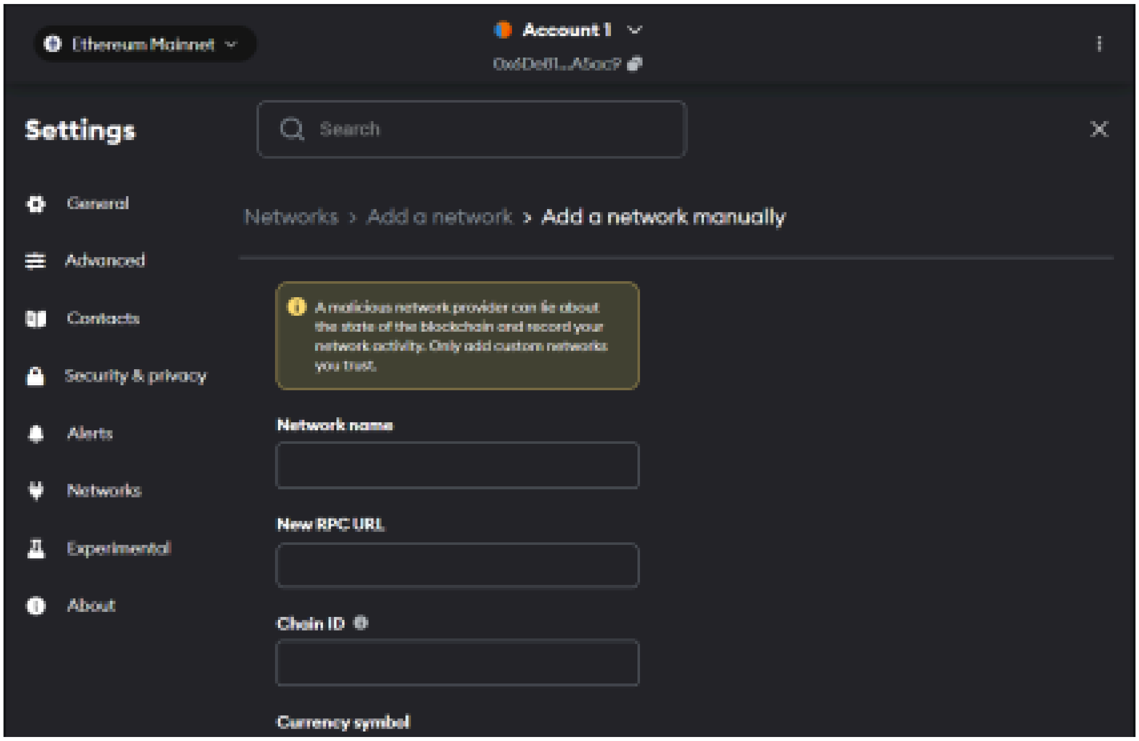
Add from a list of popular networks or add a network manually. Only interact with the entities you trust.

Popular custom networks

Arbitrum One	Add
Avalanche Network C-Chain	Add
BNB Chain	Add
Base Mainnet	Add
OP Mainnet	Add
Polygon Mainnet	Add
zkSync Era Mainnet	Add

Add a network manually

And add a network manually.



Ethereum Mainnet

Account 1
0x6De81...A5ac9

Settings

Search

- General
- Advanced
- Contacts
- Security & privacy
- Alerts
- Networks
- Experimental
- About

Networks > Add a network > Add a network manually

A malicious network provider can lie about the state of the blockchain and record your network activity. Only add custom networks you trust.

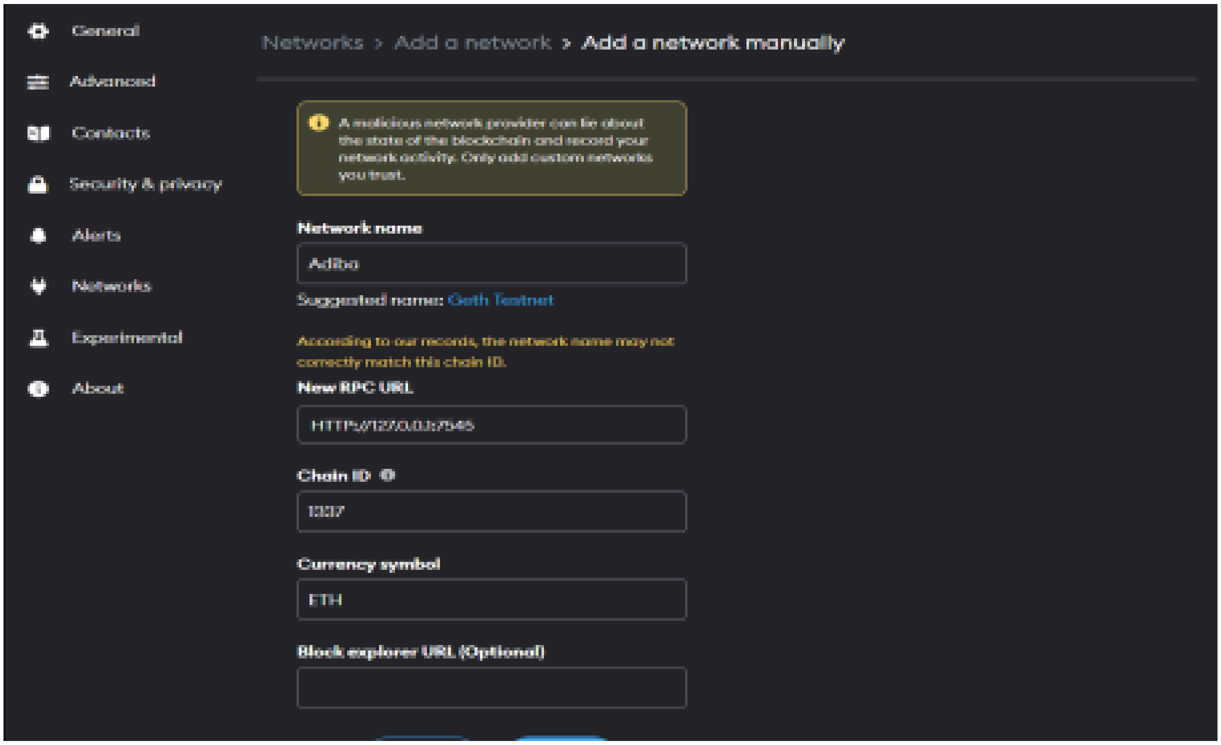
Network name

New RPC URL

Chain ID

Currency symbol

Step 5: Copy The RPC Url from the ganache
Chain ID as 1337
And currency symbol as ETH



A malicious network provider can lie about the state of the blockchain and record your network activity. Only add custom networks you trust.

Network name

Adiba

Suggested name: Geth Testnet

According to our records, the network name may not correctly match this chain ID.

New RPC URL

HTTP://127.0.0.1:7545

Chain ID

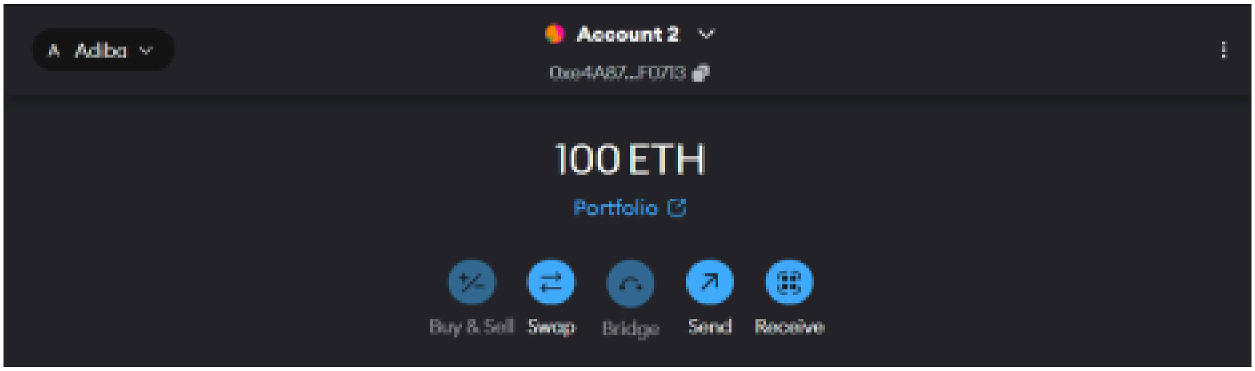
1337

Currency symbol

ETH

Block explorer URL (Optional)

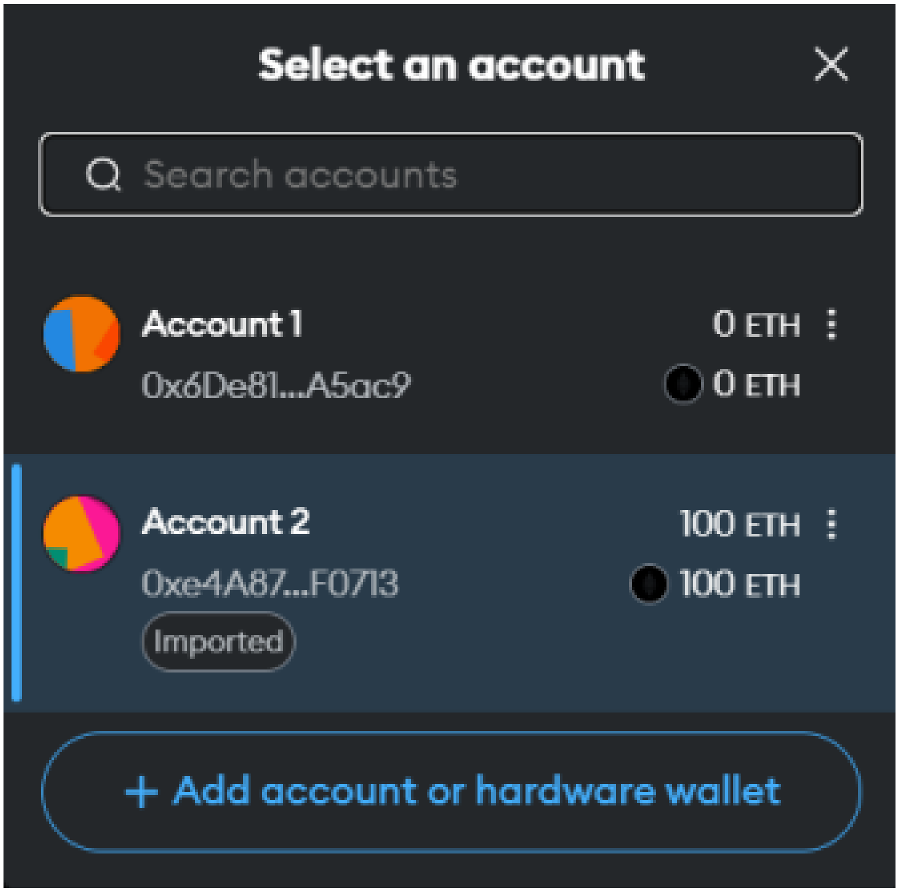
Step 6: To add ether click on account



0xe4A87...F0713

100 ETH

Step 7: Now click on add account or hardware wallet



Select an account

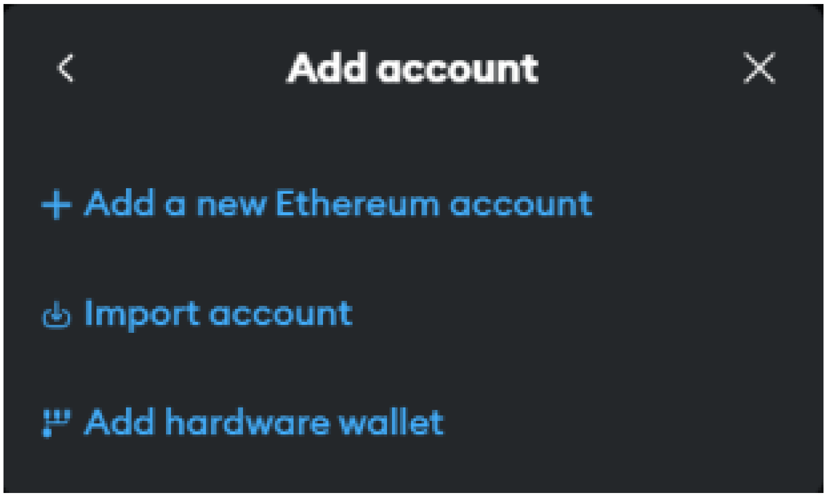
Search accounts

Account 1 — 0 ETH
0x6De81...A5ac9 — 0 ETH

Account 2 — 100 ETH
0xe4A87...F0713 — 100 ETH
Imported

+ Add account or hardware wallet

Step 8: Click on import account



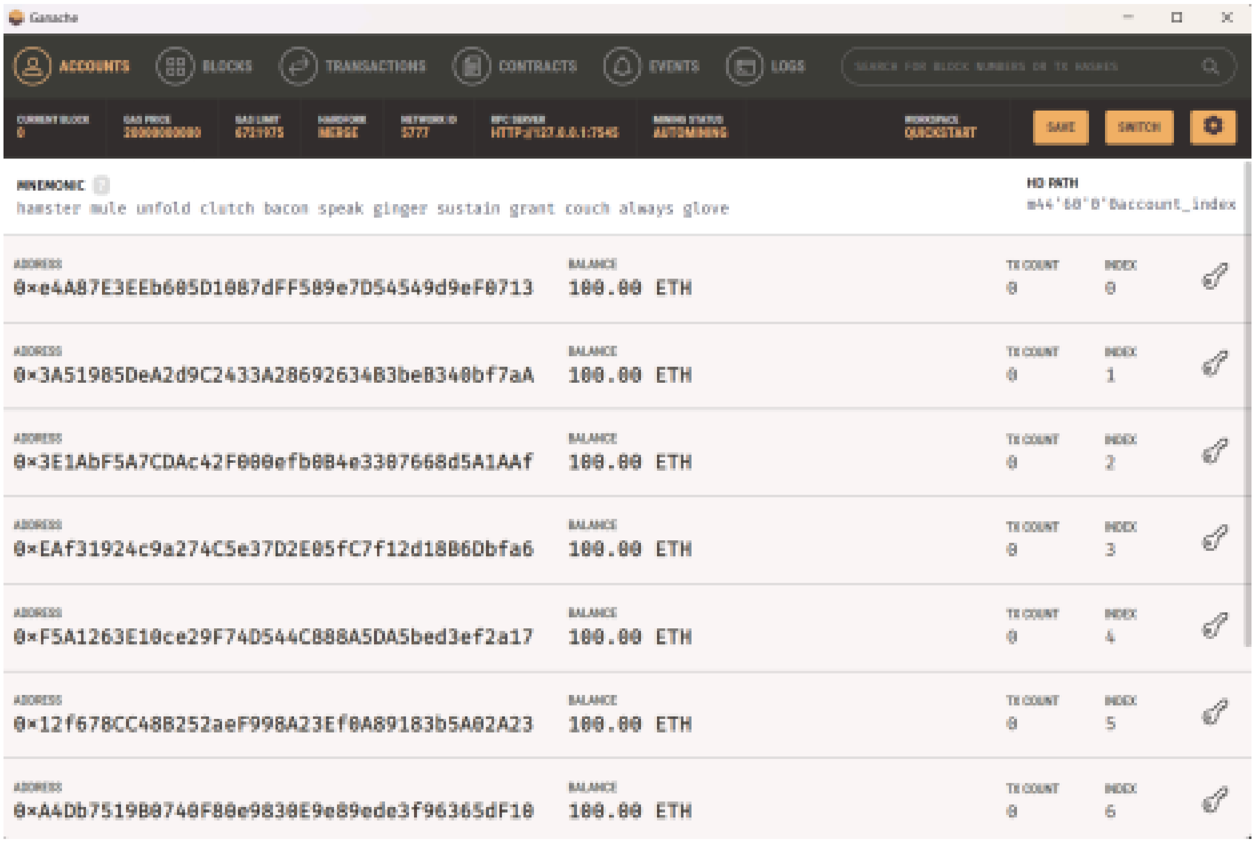
Add account

+ Add a new Ethereum account

Import account

Add hardware wallet

Step 9: Now copy the private key from ganache

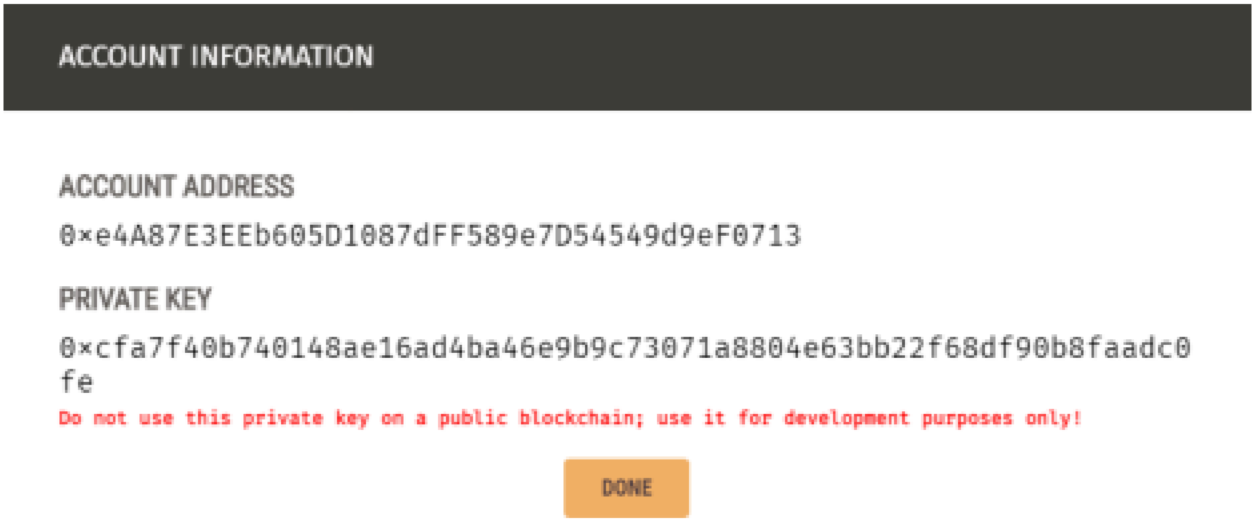


CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
0	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

MNEMONIC
hamster mule unfold clutch bacon speak ginger sustain grant couch always glove

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×e4A87E3EEb605D1087dFF589e7D54549d9eF0713	100.00 ETH	0	0
0×3A51985DeA2d9C2433A28692634B3beB340bf7aA	100.00 ETH	0	1
0×3E1AbF5A7CDAc42F000efb0B4e3307668d5A1AAf	100.00 ETH	0	2
0×EAf31924c9a274C5e37D2E05fC7f12d18B6Dbfa6	100.00 ETH	0	3
0×F5A1263E10ce29F74D544C888A5DA5bed3ef2a17	100.00 ETH	0	4
0×12f678CC48B252aeF998A23Ef0A89183b5A02A23	100.00 ETH	0	5
0×A4Db7519B0740F80e9830E9e89ede3f96365dF10	100.00 ETH	0	6



ACCOUNT INFORMATION

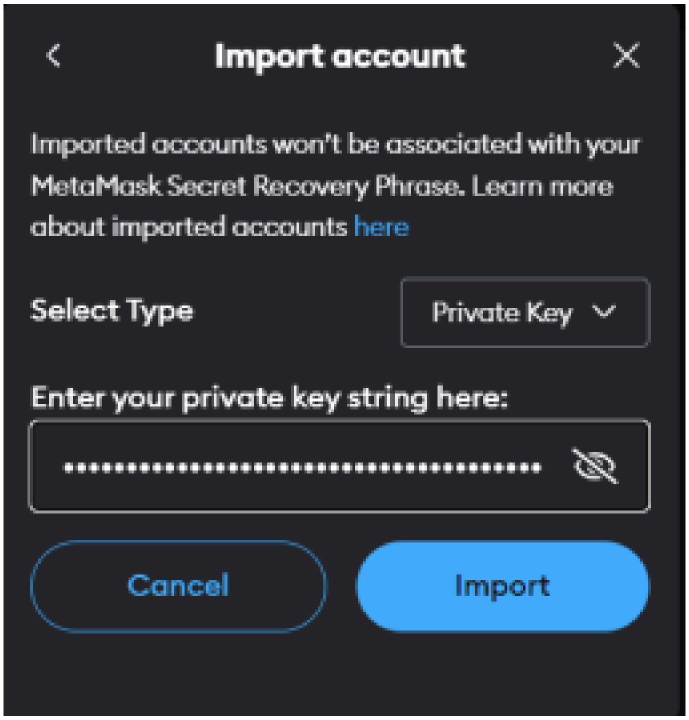
ACCOUNT ADDRESS

0×e4A87E3EEb605D1087dFF589e7D54549d9eF0713

PRIVATE KEY

0×cfa7f40b740148ae16ad4ba46e9b9c73071a8804e63bb22f68df90b8faadc0fe

Do not use this private key on a public blockchain; use it for development purposes only!



Import account

Imported accounts won't be associated with your MetaMask Secret Recovery Phrase. Learn more about imported accounts here

Select Type: Private Key

Enter your private key string here:

••

Step 10: Paste it over here and click on import

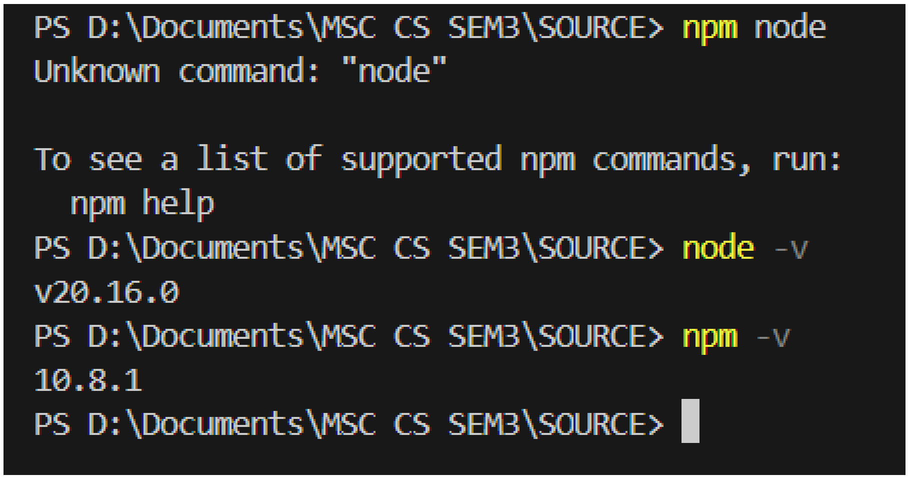
Now type this command in terminal of vs code

node -v

Npm -v

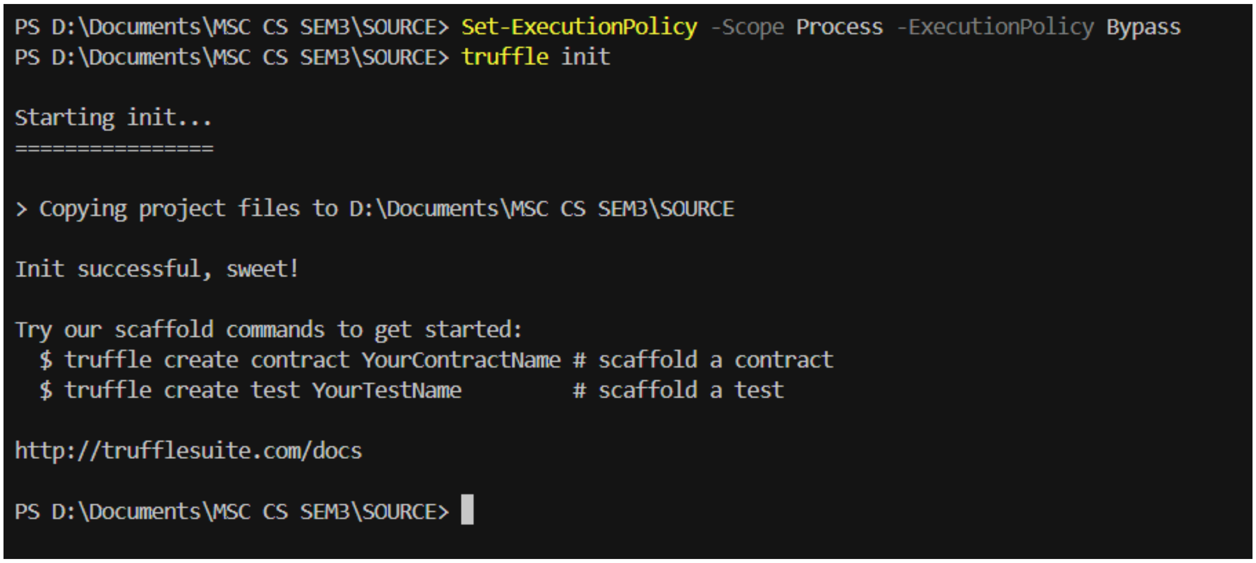
truffle version

Step 11: sudo npm install -g truffle



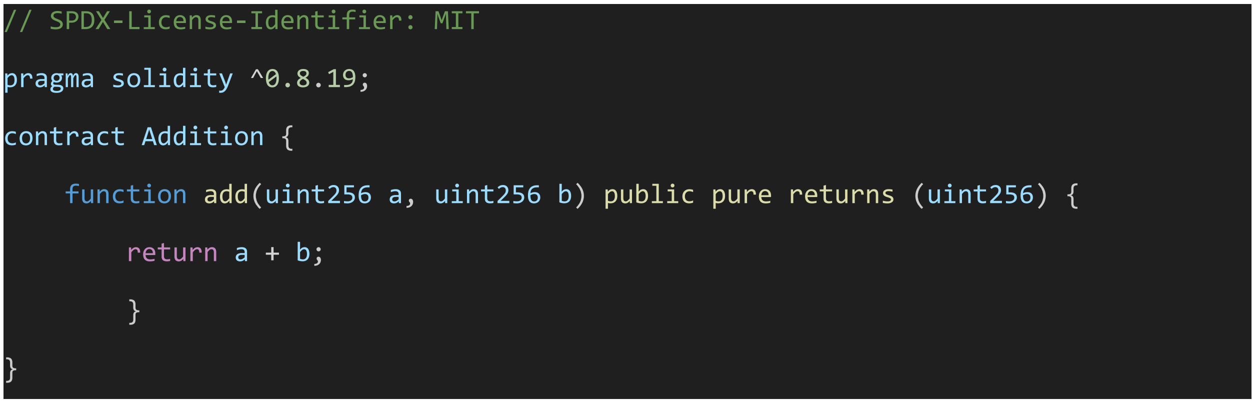
```
PS D:\Documents\MSC CS SEM3\SOURCE> npm node
Unknown command: "node"

To see a list of supported npm commands, run:
  npm help
PS D:\Documents\MSC CS SEM3\SOURCE> node -v
v20.16.0
PS D:\Documents\MSC CS SEM3\SOURCE> npm -v
10.8.1
PS D:\Documents\MSC CS SEM3\SOURCE> 
```

Step 12: Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass



```
PS D:\Documents\MSC CS SEM3\SOURCE> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
PS D:\Documents\MSC CS SEM3\SOURCE> truffle init

Starting init...
================

> Copying project files to D:\Documents\MSC CS SEM3\SOURCE

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs

PS D:\Documents\MSC CS SEM3\SOURCE> 
```

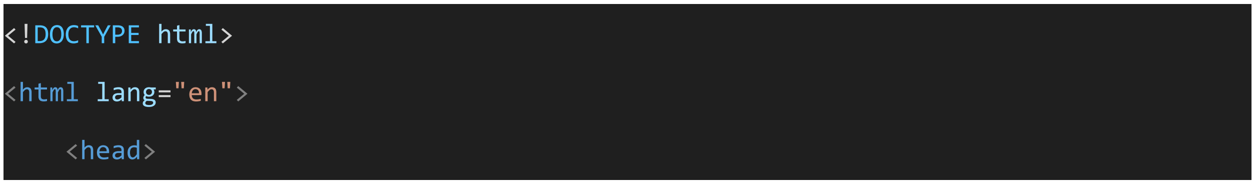
Step 13: Create a new contract in the contracts folder. And write a smart contract for adding two numbers.



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;
contract Addition {
    function add(uint256 a, uint256 b) public pure returns (uint256) {
        return a + b;
        }
}
```

Step 14: Create a new folder frontend and make index.html and app.js files inside.

index.html



```
<!DOCTYPE html>
<html lang="en">
    <head>
```

```
        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>DApp-1</title>

    </head>


    <body>

        <h1>Blockchain Addition DApp</h1>

        <input type="number" id="num1" placeholder="Enter first number">

        <input type="number" id="num2" placeholder="Enter second number">

        <button onclick="addNumber()">Add Numbers</button>

        <h3>Result: <span id="result"></span></h3>


        <script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
<script src="app.js"></script>

    </body>

</html>
```

app.js

```
const contractAddress = ""; // Replace with your deployed contract address

const contractABI = []; // Use ABI from compiled contract


let web3;

let contract;


window.addEventListener("load", async () => {

    if (window.ethereum) {

        web3 = new Web3(window.ethereum);

        await window.ethereum.enable();

    } else {
        console.log("MetaMask not detected. Please install MetaMask.");
    }

    contract = new web3.eth.Contract(contractABI, contractAddress); });
async function addNumber() {
    const num1 = document.getElementById("num1").value;
    const num2 = document.getElementById("num2").value;
    const accounts = await web3.eth.getAccounts();
    console.log(num1);
    console.log(num2);
    contract.methods
    .add(num1, num2)
    .call({ from: accounts[0] })
    .then((result) => {
        console.log(result);
        document.getElementById("result").innerText = `${result}`;
    });
}
```

Step 15: Create 1_deploy.js in the migrations folder.

```
const Addition = artifacts.require("Addition");
module.exports = async function (deployer) {
    await deployer.deploy(Addition);
    const instance = await Addition.deployed();
    console.log("Addition deployed at:", instance.address);
};
```

Step 16: Create test.js in the test folder to verify the contracts before deploying it.

```
const Addition = artifacts.require("Addition");

contract("Addition", () => {

    it("should add two numbers correctly", async () => {

        const addition = await Addition.deployed();

        console.log("Contract Address: ", addition.address);

        const result = await addition.add(5, 3);

        assert.equal(result.toNumber(), 8, "Addition of 5 and 3 should be 8");

    });

});
```

Step 17: In the source directory create a new file bs-config.json and set the base directory as frontend.

```
{
    "server": {
        "baseDir": ["./frontend" ]
    }
}
```

Step 18:

Make sure about the following things

a. In the truffle-config.js uncomment your network details. And ensure the port and network_id match with the RPC Server which can be found in Ganache GUI

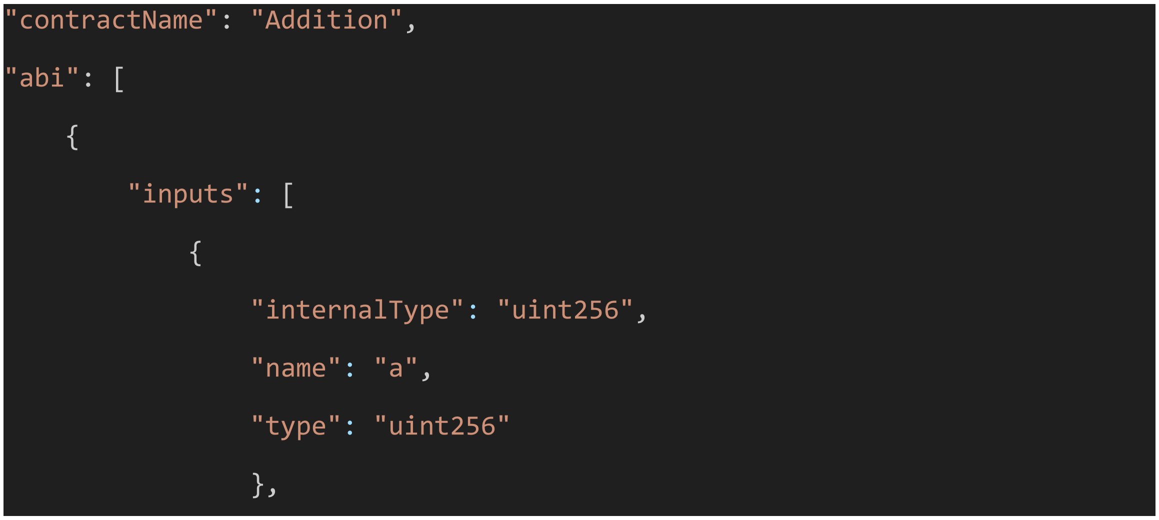
b. Ensure that the solidity compiler version is set to 0.8.19 in the same file.

c. Ensure necessary dependencies are mentioned in the package.json.

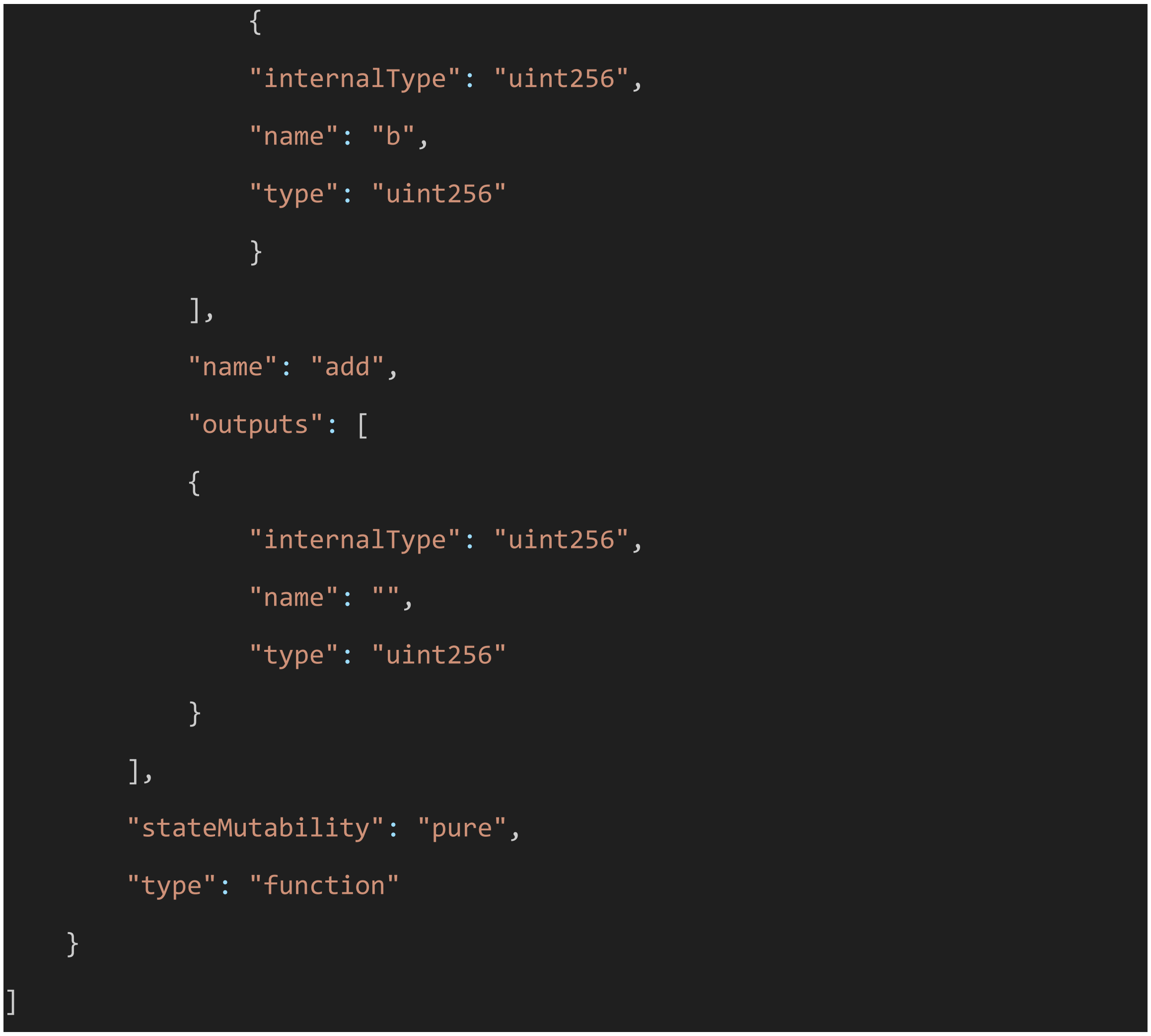
```
{

    "dependencies": {

        "lite-server": "^2.6.1"

    }

}
```

Running the DApp.
1. In a new terminal set directory to source and run truffle compile command.

2. Go to build → contracts → Addition.json. Look for abi and Copy the complete array. Paste it in the contractABI constant inside app.js.



```
"contractName": "Addition",
"abi": [
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "a",
                "type": "uint256"
            },
```



```
                {
                "internalType": "uint256",
                "name": "b",
                "type": "uint256"
                }
            ],
            "name": "add",
            "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "pure",
        "type": "function"
    }
]
```

3. Next run truffle migrate. Make note of the Contract Address displayed in the terminal.

4. Copy the contract address and paste it in the contractAddress constant in the app.js file.

5. Run a truffle test to ensure our contract is correct.

6. Run npm start if everything is correct.

7. Sign in to MetaMask and grant the required access.

8. Give the input and click on Add Numbers. The result should be displayed.

Now, Modify the DApp to integrate subtraction, multiplication & division operations.

1. Make changes in the smart contract (Operations.sol → I have renamed the file)

2. Modify index.html to accommodate other buttons and onClick functions.

3. Similarly modify app.js.

Name: Adiba Mohammed Raza Siddique
Seat No: 31031523033

S K Somaiya College
Somaiya Vidyavihar University

4. Modify 1_deploy.js (If you haven’t renamed leave it as is)

5. Update test.js to include different test cases.

6. Run the DApp following the same steps in addition.

Name: Adiba Mohammed Raza Siddique
Seat No: 31031523033

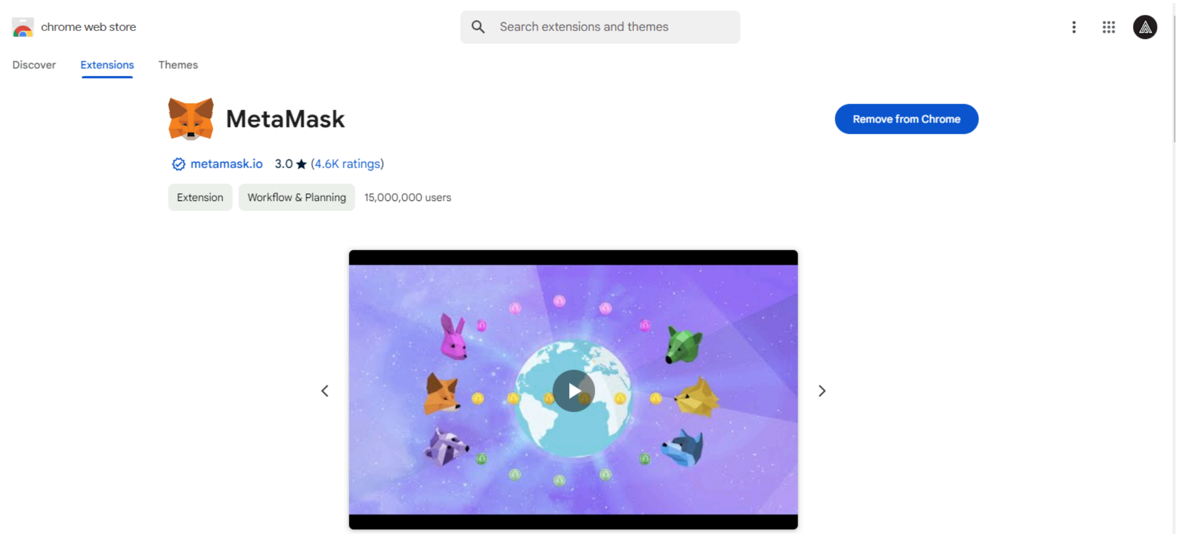
S K Somaiya College
Somaiya Vidyavihar University

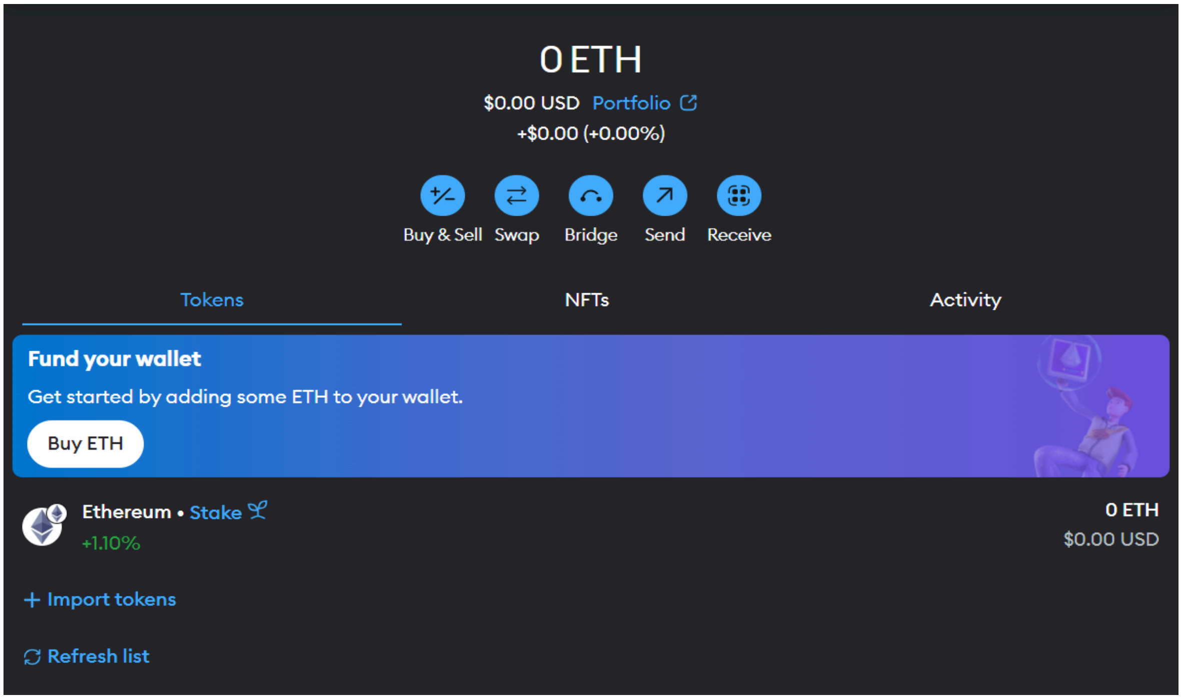
Practical 12

A] Aim: Creating a simple DApp for Factorial of numbers.

Step 1: Setting up MetaMask and Ganache.

Install MetaMask Extension from here and create an account. https://chromewebstore.google.com/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en

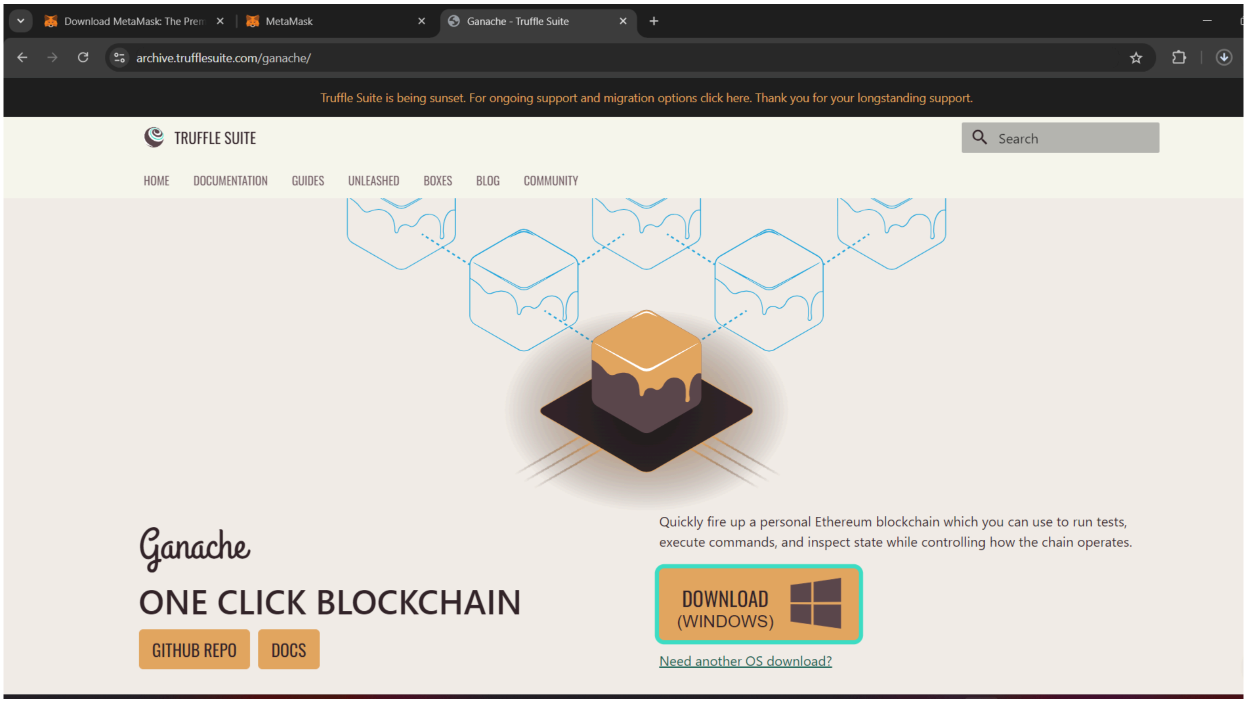




Fund your wallet

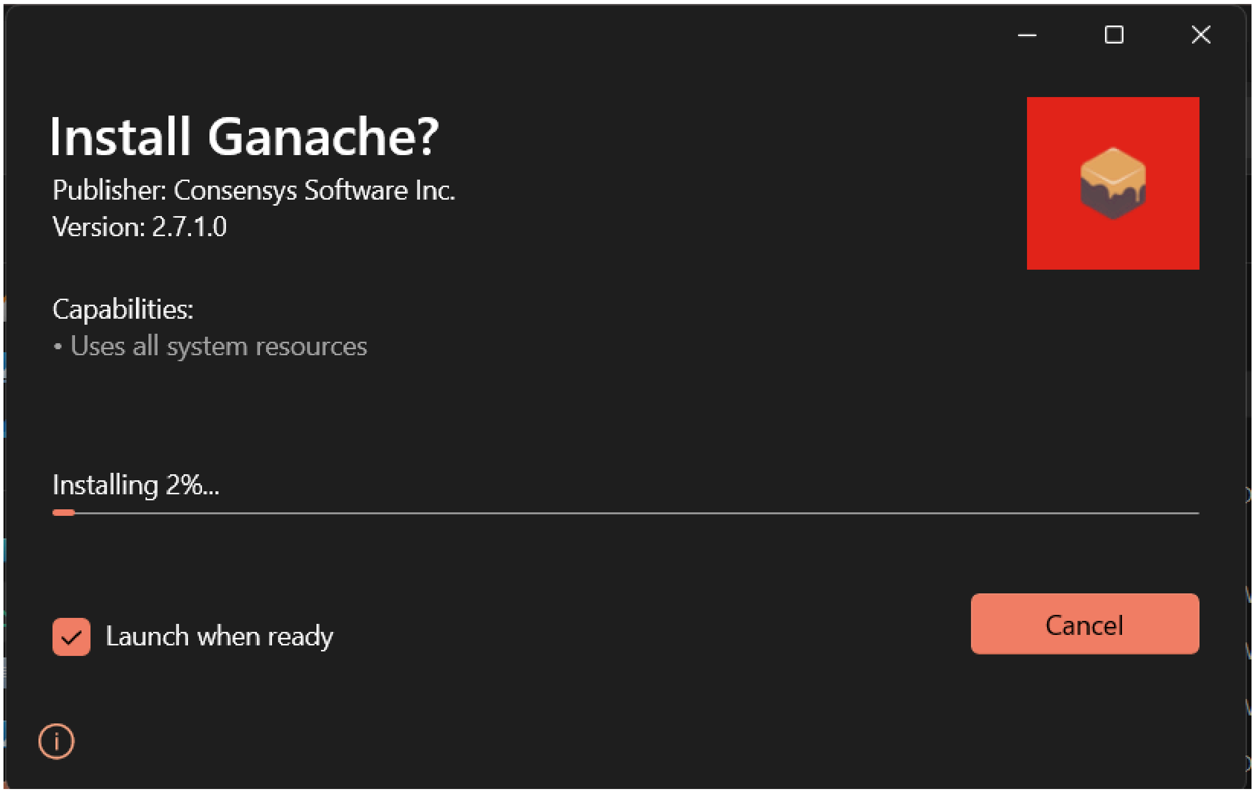
Get started by adding some ETH to your wallet.

Step 2: Install Ganache from here. https://archive.trufflesuite.com/ganache/



Truffle Suite is being sunset. For ongoing support and migration options click here. Thank you for your longstanding support.

Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.



Install Ganache?

Publisher: Consensys Software Inc.
Version: 2.7.1.0

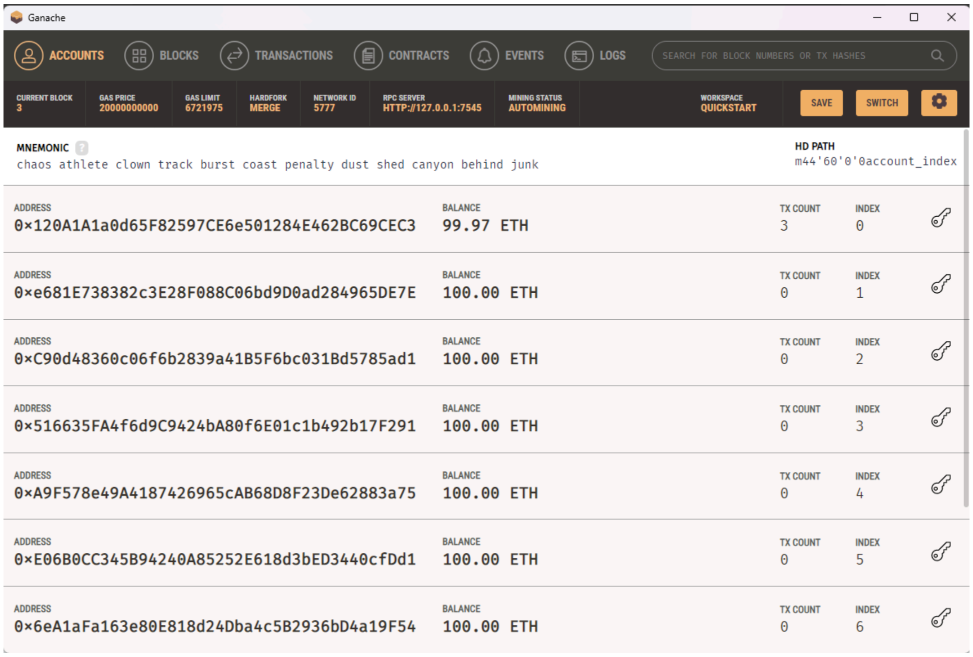
Capabilities:
- Uses all system resources

Installing 2%...

☑ Launch when ready

After install open it and click on **QUICKSTART**

Step 3: Now Copy the key of any id.



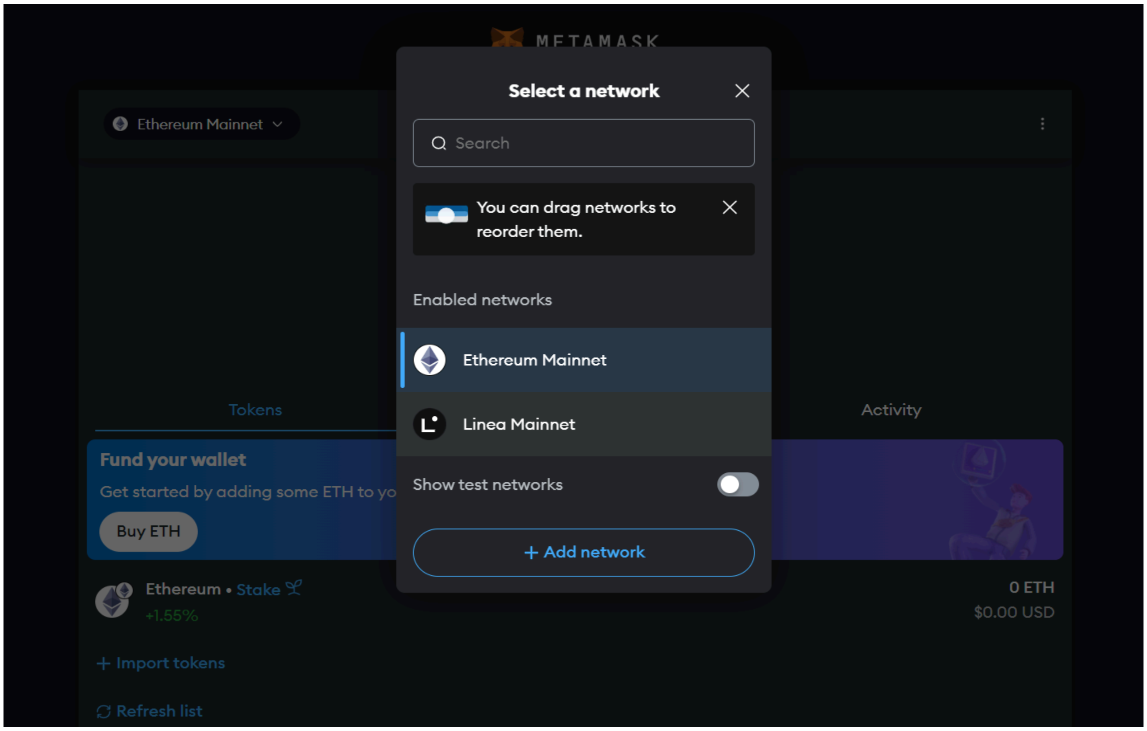
CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
3	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

MNEMONIC
chaos athlete clown track burst coast penalty dust shed canyon behind junk

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×120A1A1a0d65F82597CE6e501284E462BC69CEC3	99.97 ETH	3	0
0×e681E738382c3E28F088C06bd9D0ad284965DE7E	100.00 ETH	0	1
0×C90d48360c06f6b2839a41B5F6bc031Bd5785ad1	100.00 ETH	0	2
0×516635FA4f6d9C9424bA80f6E01c1b492b17F291	100.00 ETH	0	3
0×A9F578e49A4187426965cAB68D8F23De62883a75	100.00 ETH	0	4
0×E06B0CC345B94240A85252E618d3bED3440cfDd1	100.00 ETH	0	5
0×6eA1aFa163e80E818d24Dba4c5B2936bD4a19F54	100.00 ETH	0	6

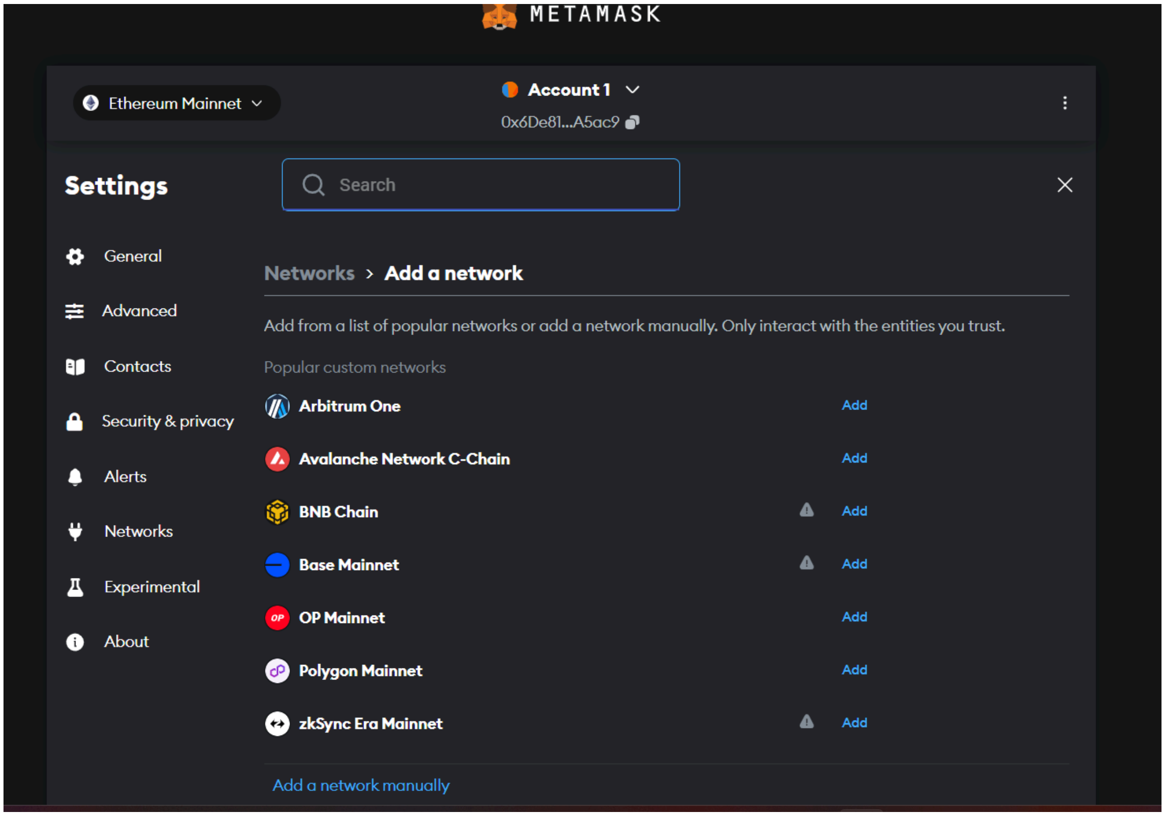
Step 4: Now add the network in metamask.



Select a network

You can drag networks to reorder them.

Click on add network.



Settings

Networks > Add a network

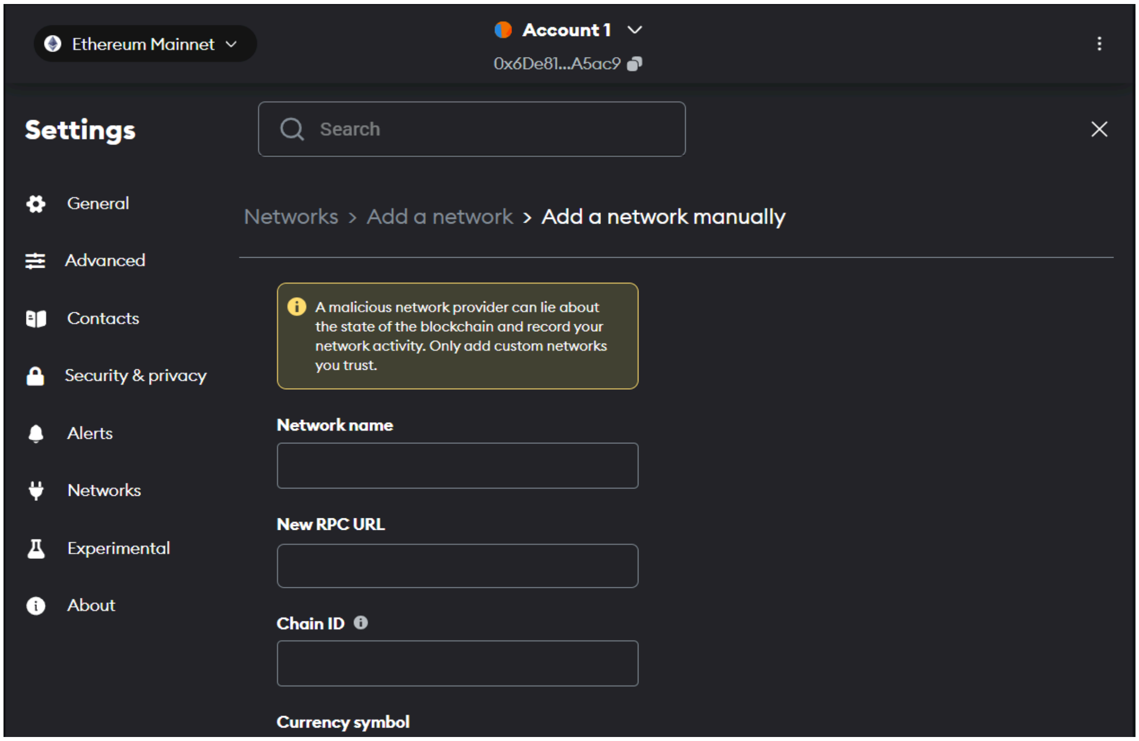
Add from a list of popular networks or add a network manually. Only interact with the entities you trust.

Popular custom networks

Arbitrum One	Add
Avalanche Network C-Chain	Add
BNB Chain	Add
Base Mainnet	Add
OP Mainnet	Add
Polygon Mainnet	Add
zkSync Era Mainnet	Add

Add a network manually

And add a network manually.



Settings

Networks > Add a network > Add a network manually

A malicious network provider can lie about the state of the blockchain and record your network activity. Only add custom networks you trust.

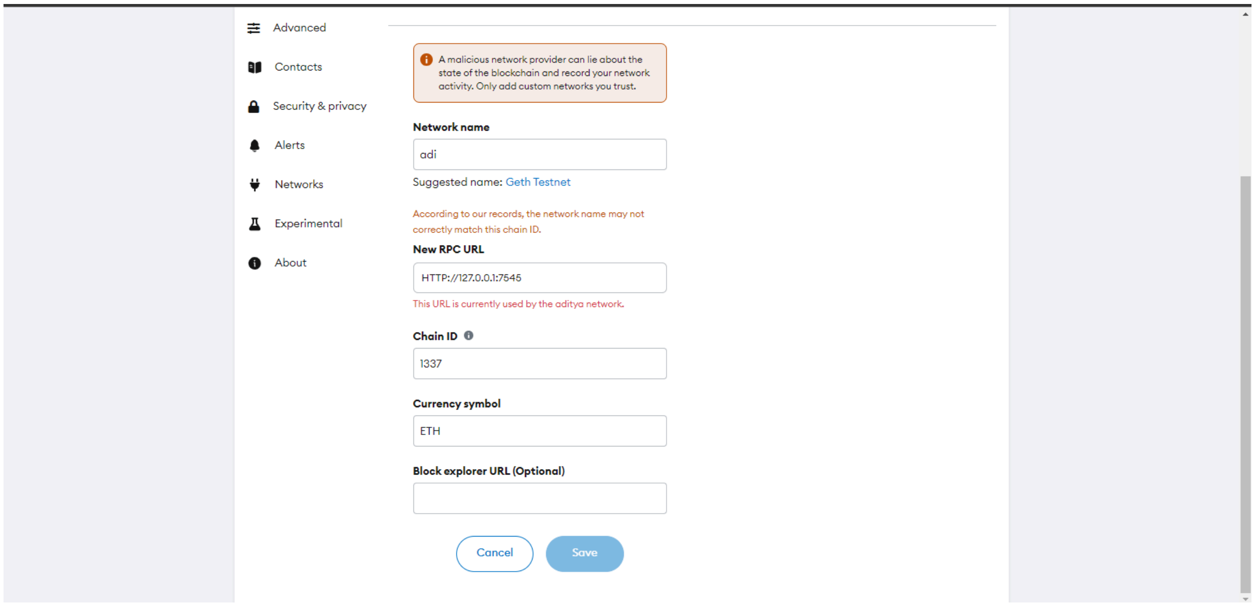
Network name

New RPC URL

Chain ID

Currency symbol

Step 5: Copy The RPC Url from the ganache
Chain ID as 1337
And currency symbol as ETH



A malicious network provider can lie about the state of the blockchain and record your network activity. Only add custom networks you trust.

Network name: adi

Suggested name: Geth Testnet

According to our records, the network name may not correctly match this chain ID.

New RPC URL: HTTP://127.0.0.1:7545

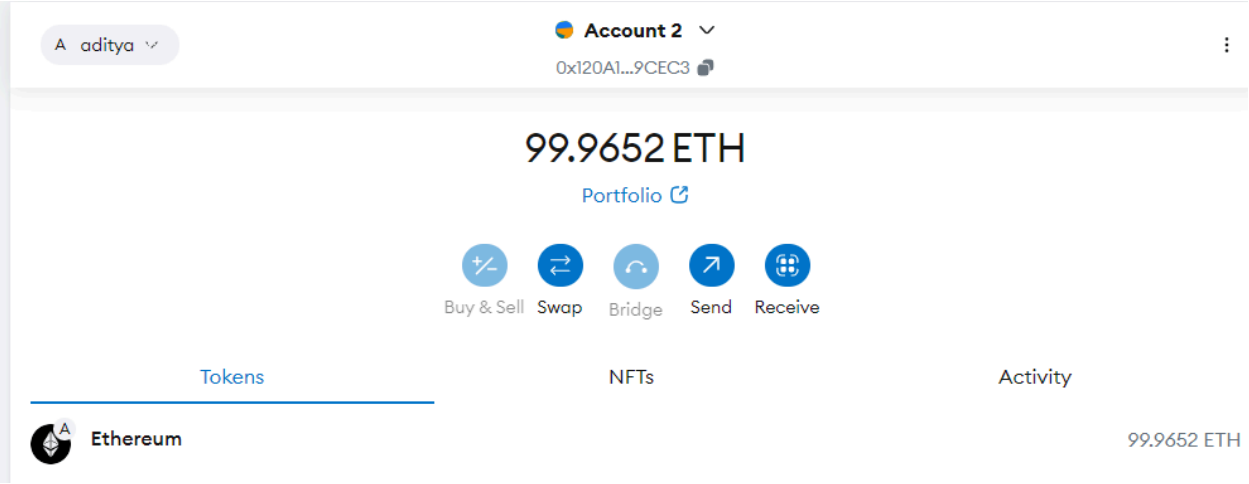
This URL is currently used by the aditya network.

Chain ID: 1337

Currency symbol: ETH

Block explorer URL (Optional):

Step 6: To add ether click on account

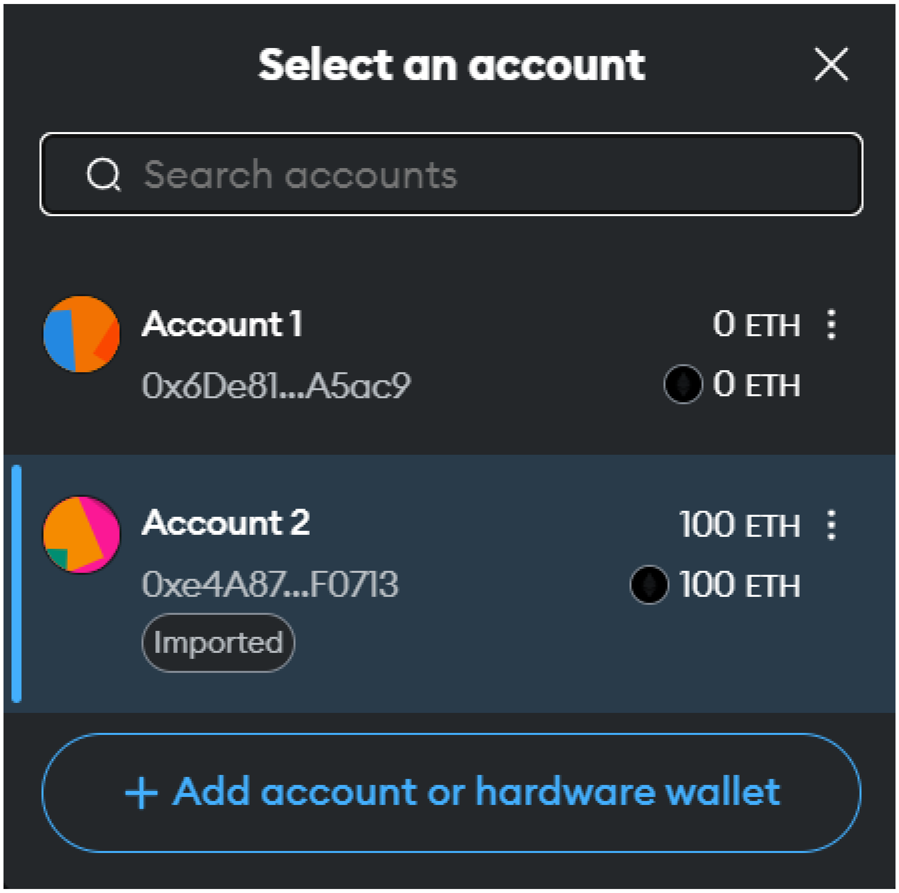


Account 2
0x120A1...9CEC3

99.9652 ETH

Ethereum 99.9652 ETH

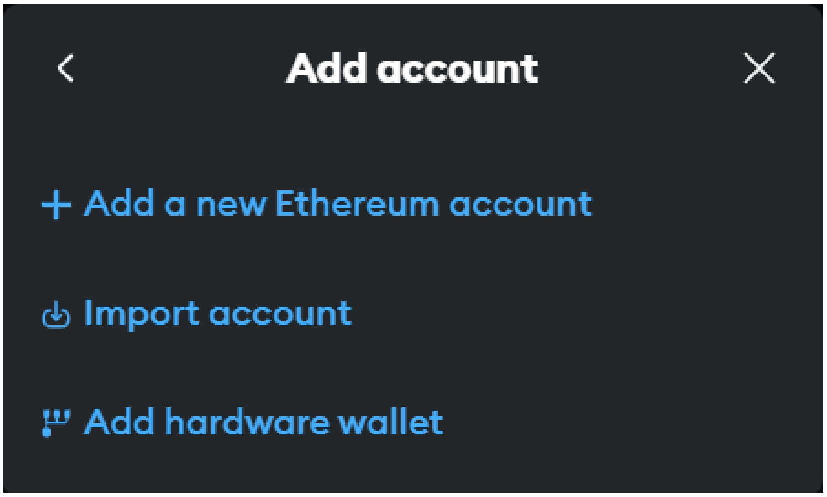
Step 7: Now click on add account or hardware wallet



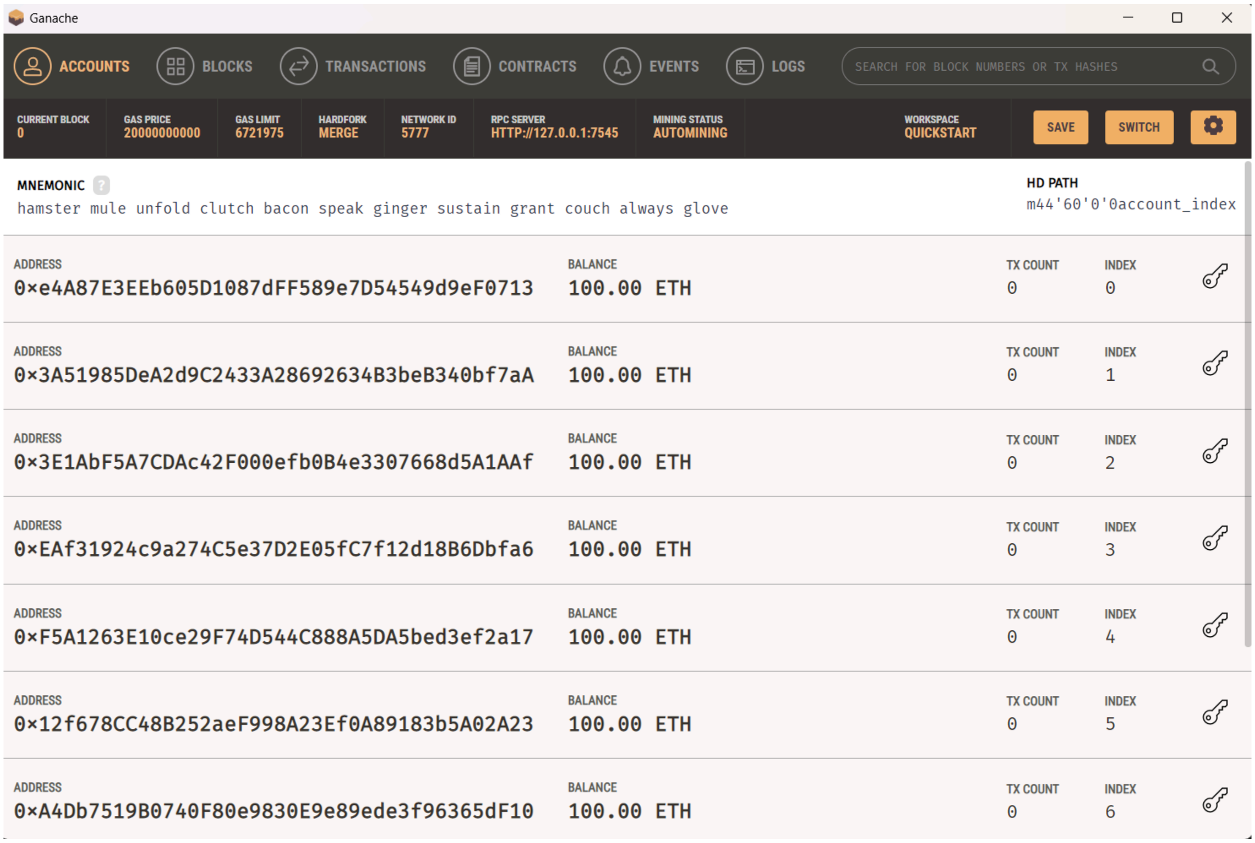
Account 1 0 ETH
0x6De81...A5ac9 0 ETH

Account 2 100 ETH
0xe4A87...F0713 100 ETH
Imported

Step 8: Click on import account



Step 9: Now copy the private key from ganache

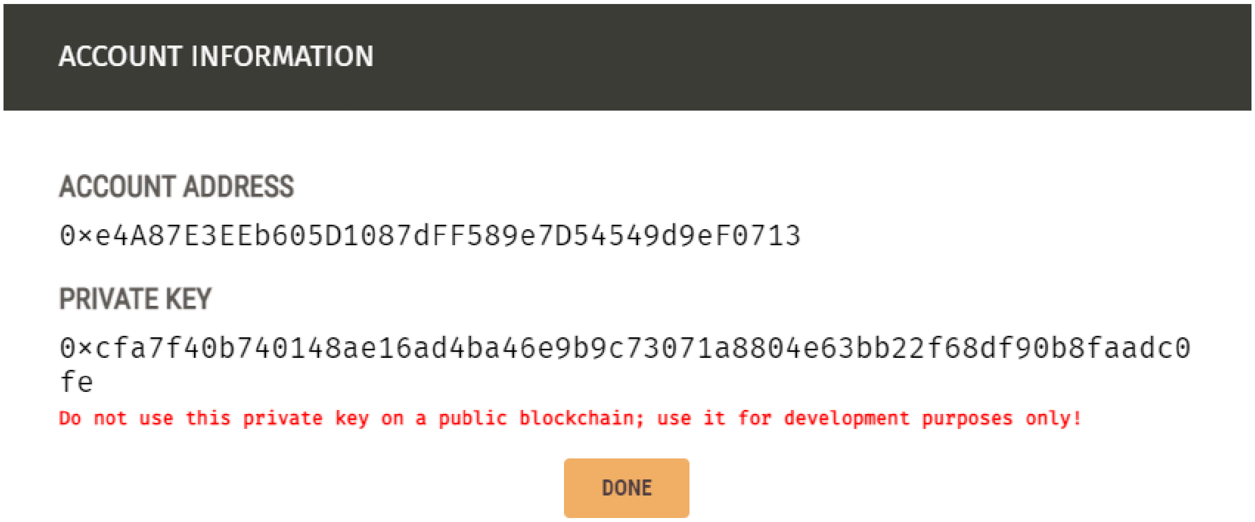


CURRENT BLOCK 0 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE QUICKSTART

MNEMONIC
hamster mule unfold clutch bacon speak ginger sustain grant couch always glove

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×e4A87E3EEb605D1087dFF589e7D54549d9eF0713	100.00 ETH	0	0
0×3A51985DeA2d9C2433A28692634B3beB340bf7aA	100.00 ETH	0	1
0×3E1AbF5A7CDAc42F000efb0B4e3307668d5A1AAf	100.00 ETH	0	2
0×EAf31924c9a274C5e37D2E05fC7f12d18B6Dbfa6	100.00 ETH	0	3
0×F5A1263E10ce29F74D544C888A5DA5bed3ef2a17	100.00 ETH	0	4
0×12f678CC48B252aeF998A23Ef0A89183b5A02A23	100.00 ETH	0	5
0×A4Db7519B0740F80e9830E9e89ede3f96365dF10	100.00 ETH	0	6

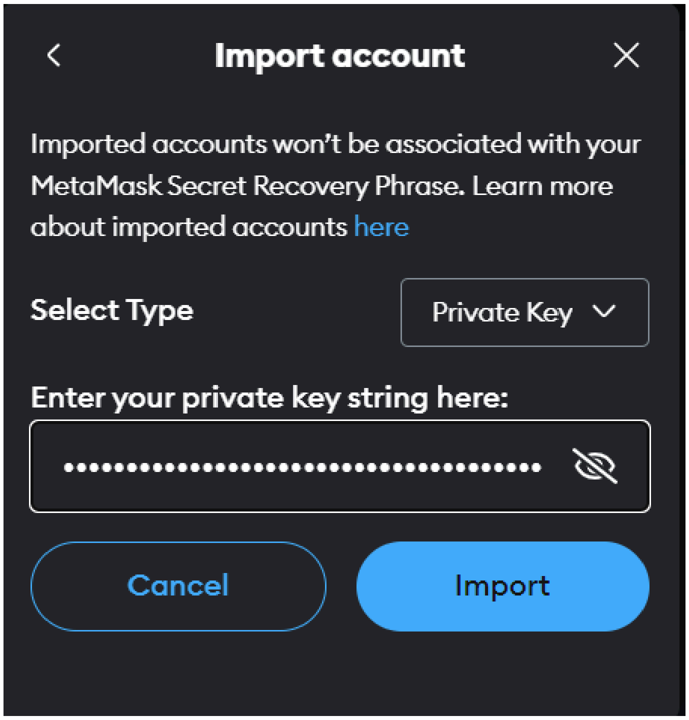


ACCOUNT INFORMATION

ACCOUNT ADDRESS
0×e4A87E3EEb605D1087dFF589e7D54549d9eF0713

PRIVATE KEY
0×cfa7f40b740148ae16ad4ba46e9b9c73071a8804e63bb22f68df90b8faadc0fe

Do not use this private key on a public blockchain; use it for development purposes only!



Import account

Imported accounts won't be associated with your MetaMask Secret Recovery Phrase. Learn more about imported accounts here

Select Type: Private Key

Enter your private key string here:

••••••••••••••••••••••••••••••••••••

Step 10: Paste it over here and click on import

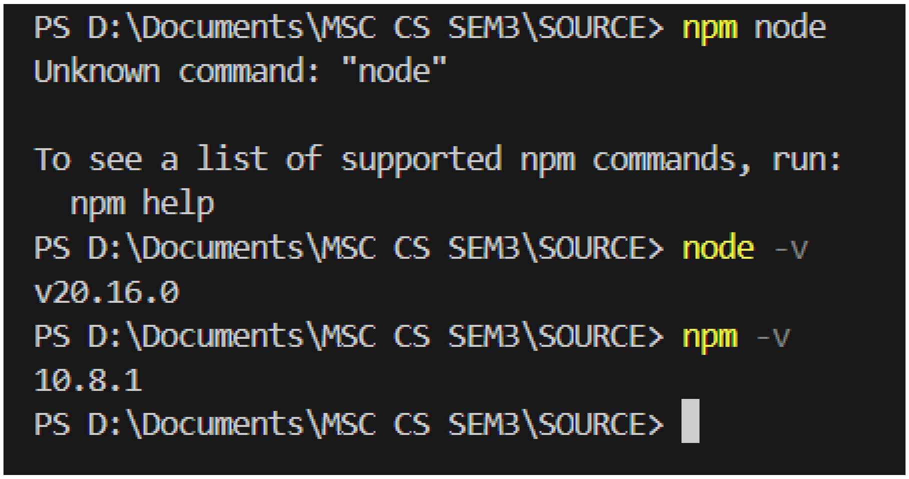
Now type this command in terminal of vs code

node -v

Npm -v

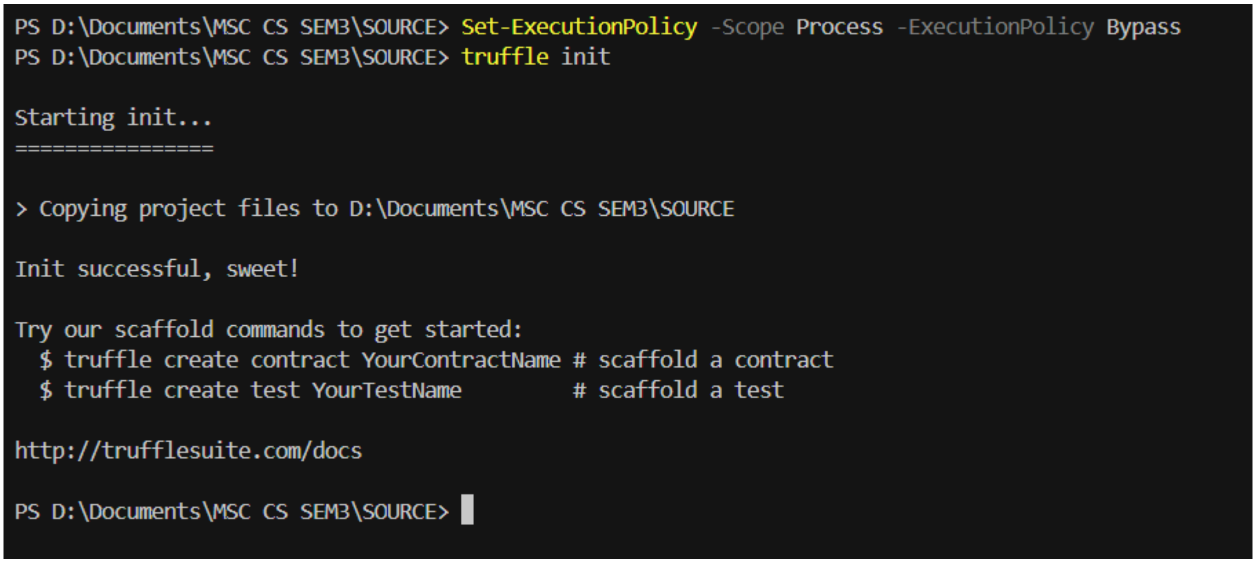
truffle version

Step 11: sudo npm install -g truffle



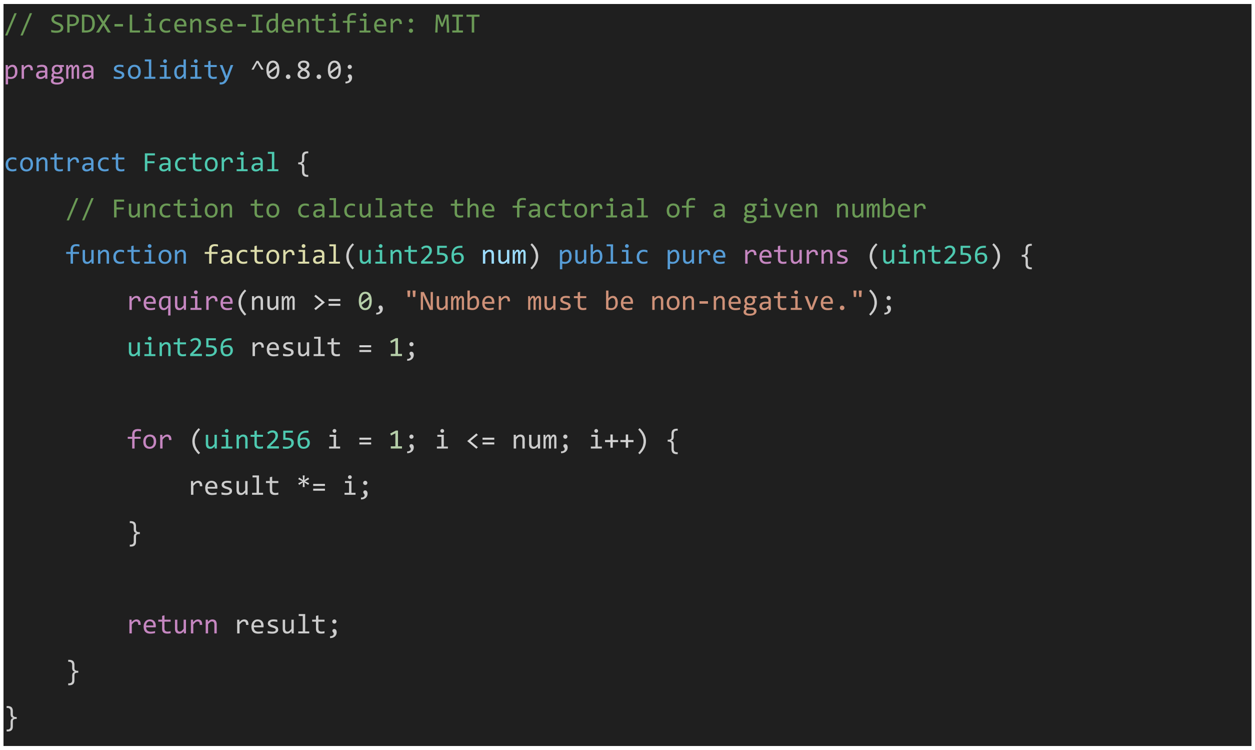
```
PS D:\Documents\MSC CS SEM3\SOURCE> npm node
Unknown command: "node"

To see a list of supported npm commands, run:
  npm help
PS D:\Documents\MSC CS SEM3\SOURCE> node -v
v20.16.0
PS D:\Documents\MSC CS SEM3\SOURCE> npm -v
10.8.1
PS D:\Documents\MSC CS SEM3\SOURCE>
```

Step 12: Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass



```
PS D:\Documents\MSC CS SEM3\SOURCE> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
PS D:\Documents\MSC CS SEM3\SOURCE> truffle init

Starting init...
================

> Copying project files to D:\Documents\MSC CS SEM3\SOURCE

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs

PS D:\Documents\MSC CS SEM3\SOURCE>
```

Step 13: Create a new contract in the contracts folder. And write a smart contract for adding two numbers.



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

contract Factorial {
    // Function to calculate the factorial of a given number
    function factorial(uint256 num) public pure returns (uint256) {
        require(num >= 0, "Number must be non-negative.");
        uint256 result = 1;

        for (uint256 i = 1; i <= num; i++) {
            result *= i;
        }

        return result;
    }
}
```

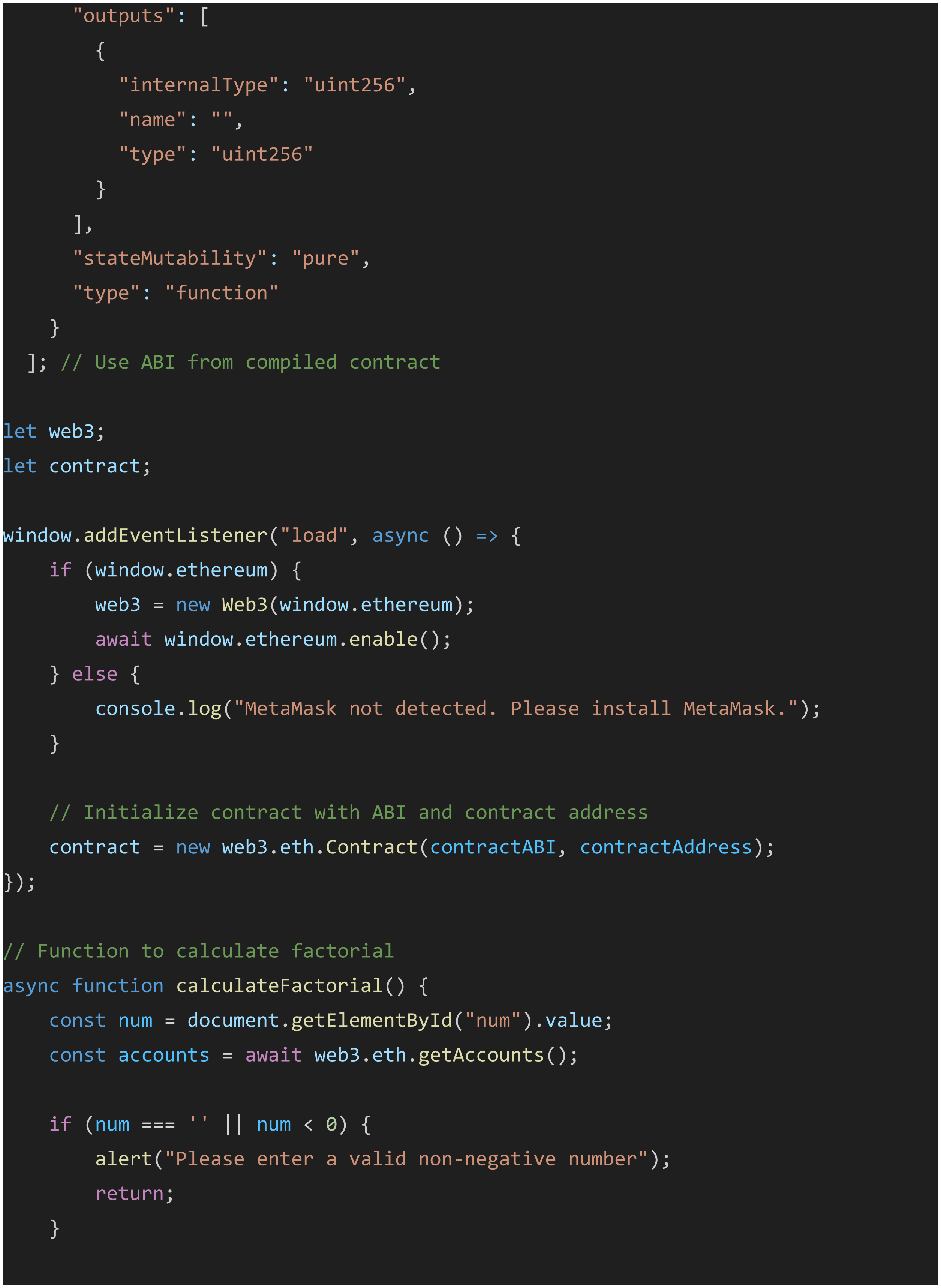
Step 14: Create a new folder frontend and make index.html and app.js files inside.

index.html

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>DApp-1 Factorial</title>
    </head>

    <body>
        <h1>Blockchain Factorial DApp</h1>
        <input type="number" id="num" placeholder="Enter a number">
        <button onclick="calculateFactorial()">Calculate Factorial</button>
        <h3>Result: <span id="result"></span></h3>

        <script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
        <script src="app.js"></script>
    </body>
</html>
```

app.js

```javascript
const contractAddress = "0x82910e8f1Af0aAd0A7c4bf50e9C59612c2e82b41"; // Replace with your deployed contract address
const contractABI =  [
    {
      "inputs": [
        {
          "internalType": "uint256",
          "name": "num",
          "type": "uint256"
        }
      ],
      "name": "factorial",
```



```
      "outputs": [
        {
          "internalType": "uint256",
          "name": "",
          "type": "uint256"
        }
      ],
      "stateMutability": "pure",
      "type": "function"
    }
  ]; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
    if (window.ethereum) {
        web3 = new Web3(window.ethereum);
        await window.ethereum.enable();
    } else {
        console.log("MetaMask not detected. Please install MetaMask.");
    }

    // Initialize contract with ABI and contract address
    contract = new web3.eth.Contract(contractABI, contractAddress);
});

// Function to calculate factorial
async function calculateFactorial() {
    const num = document.getElementById("num").value;
    const accounts = await web3.eth.getAccounts();

    if (num === '' || num < 0) {
        alert("Please enter a valid non-negative number");
        return;
    }
    console.log(num);

    // Call the factorial method from the contract
    contract.methods
    .factorial(num)
    .call({ from: accounts[0] })
    .then((result) => {
        console.log(result);
        document.getElementById("result").innerText = `${result}`;
    })
    .catch((error) => {
        console.error("Error:", error);
    });
}
```

Step 15: Create 1_deploy.js in the migrations folder.

```
const Factorial = artifacts.require("Factorial");

module.exports = async function (deployer) {
    // Deploy the contract
    await deployer.deploy(Factorial);

    // Log the deployed contract address
    const instance = await Factorial.deployed();
    console.log("Factorial contract deployed at:", instance.address);
};
```

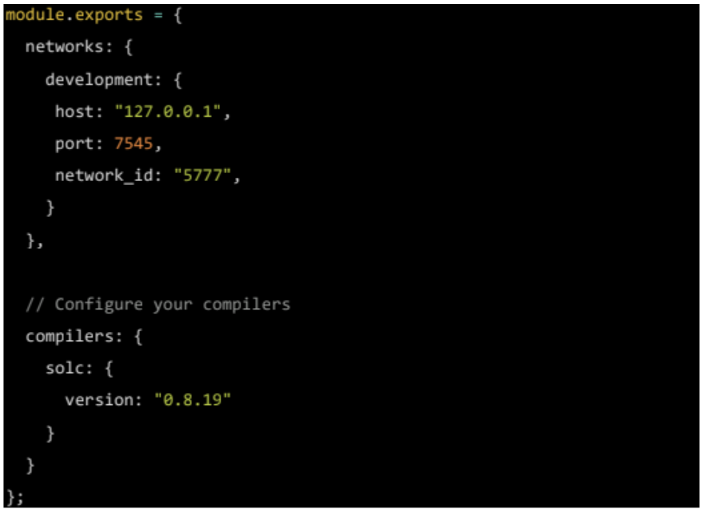
Step 16: Create test.js in the test folder to verify the contracts before deploying it.

```
const Factorial = artifacts.require("Factorial");

contract("Factorial", () => {
    it("should calculate the factorial of a number correctly", async () => {
        const factorial = await Factorial.deployed();
        console.log("Contract Address: ", factorial.address);
        // Test factorial of 5 (5! = 120)
        const result = await factorial.factorial(5);
        assert.equal(result.toNumber(), 120, "Factorial of 5 should be 120");

        // Test factorial of 3 (3! = 6)
        const result2 = await factorial.factorial(3);
        assert.equal(result2.toNumber(), 6, "Factorial of 3 should be 6");
    });
});
```

Step 17: In the source directory create a new file bs-config.json and set the base directory as frontend.

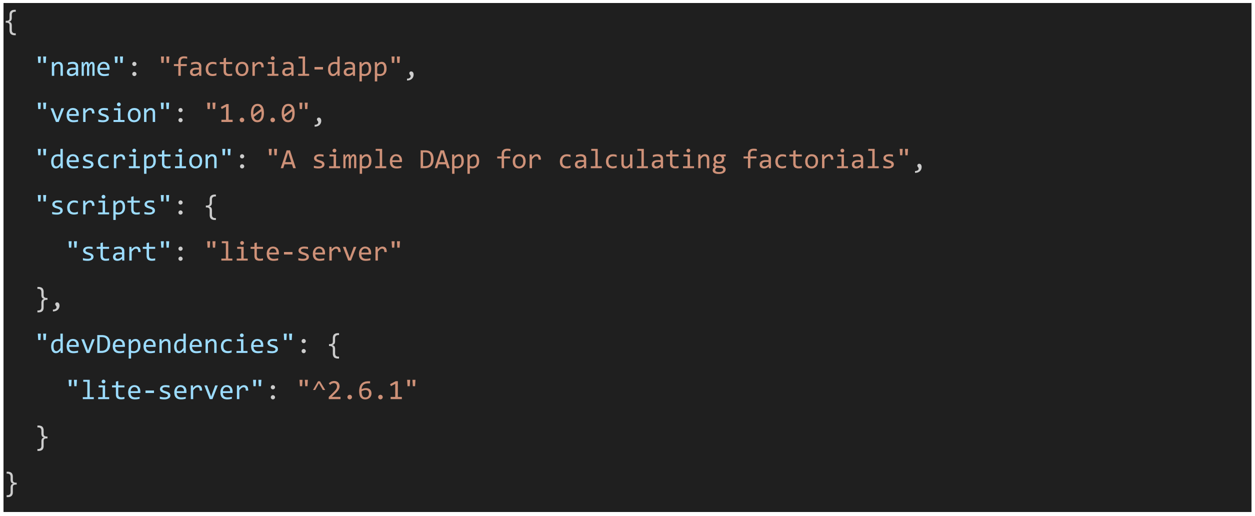
```
{
    "server": {
        "baseDir": ["./frontend" ]
    }
}
```

Step 18: Make sure about the following things
a. In the truffle-config.js uncomment your network details. And ensure the port and network_id match with the RPC Server which can be found in Ganache GUI
b. Ensure that the solidity compiler version is set to 0.8.19 in the same file.

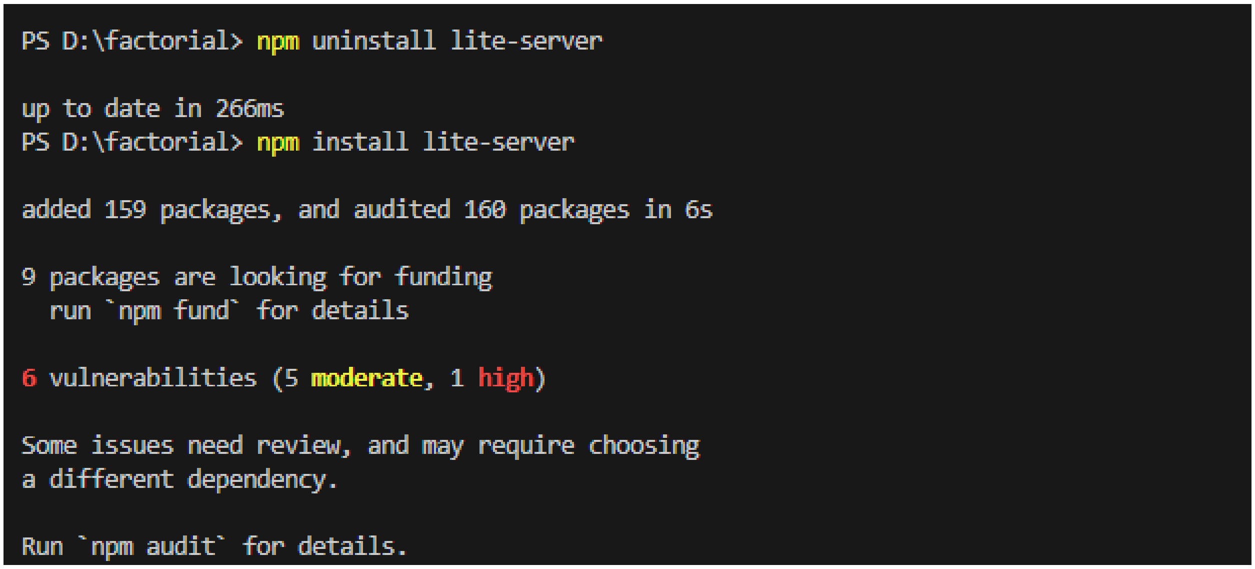


```
module.exports = {
  networks: {
    development: {
     host: "127.0.0.1",
     port: 7545,
     network_id: "5777",
    }
  },

  // Configure your compilers
  compilers: {
    solc: {
      version: "0.8.19"
    }
  }
};
```

c. Ensure necessary dependencies are mentioned in the package.json.



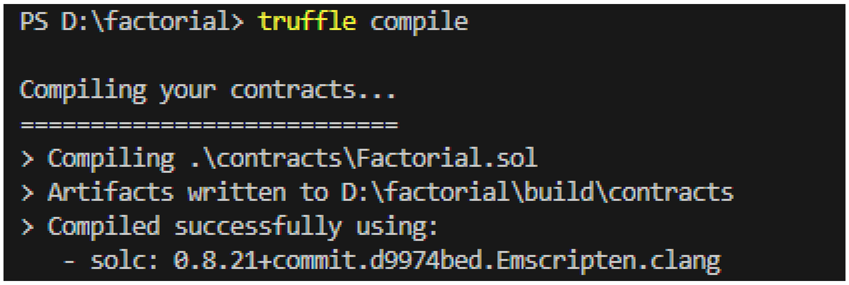
```
{
  "name": "factorial-dapp",
  "version": "1.0.0",
  "description": "A simple DApp for calculating factorials",
  "scripts": {
    "start": "lite-server"
  },
  "devDependencies": {
    "lite-server": "^2.6.1"
  }
}
```



```
PS D:\factorial> npm uninstall lite-server

up to date in 266ms
PS D:\factorial> npm install lite-server

added 159 packages, and audited 160 packages in 6s

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

6 vulnerabilities (5 moderate, 1 high)

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```

Running the DApp.

1. In a new terminal set directory to source and run truffle compile command.

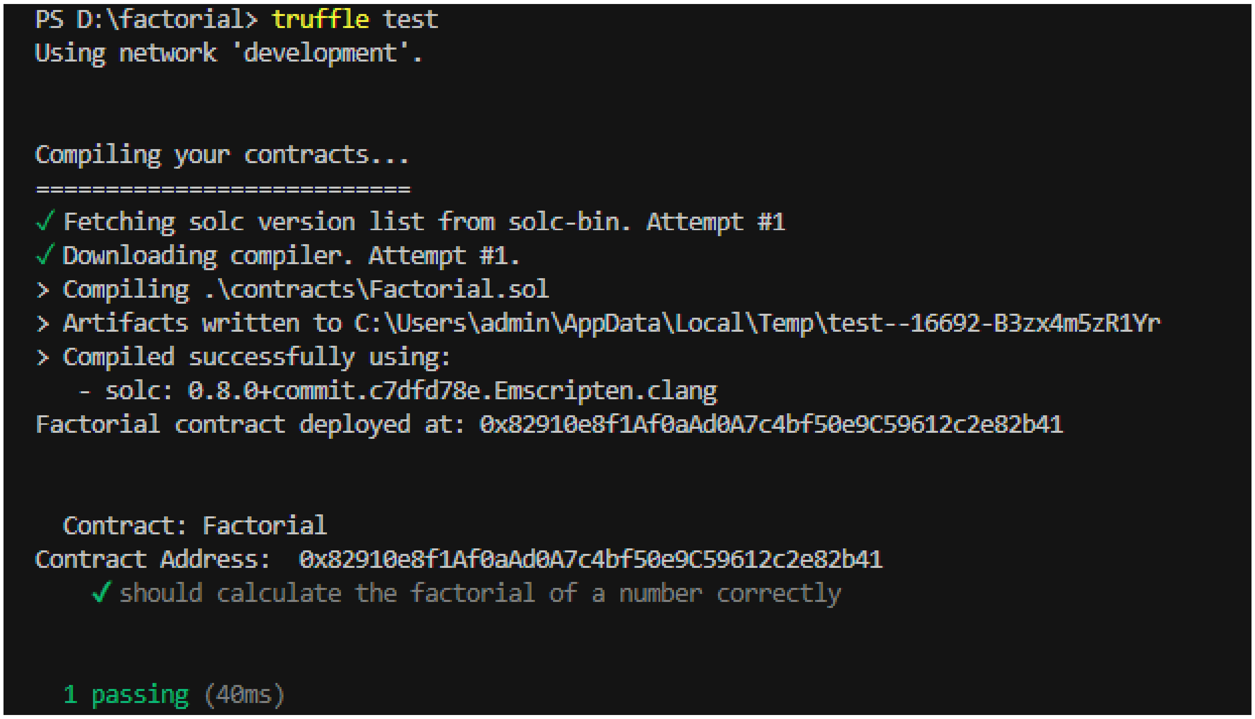


```
PS D:\factorial> truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\Factorial.sol
> Artifacts written to D:\factorial\build\contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
```

2. Go to build → contracts → Factorial.json. Look for abi and Copy the complete array. Paste it in the contractABI constant inside app.js.

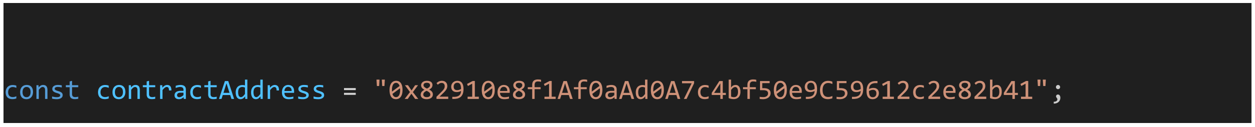
```
{
  "contractName": "Factorial",
  "abi": [
    {
      "inputs": [
        {
          "internalType": "uint256",
          "name": "num",
          "type": "uint256"
        }
      ],
      "name": "factorial",
      "outputs": [
        {
          "internalType": "uint256",
          "name": "",
          "type": "uint256"
        }
      ],
      "stateMutability": "pure",
      "type": "function"
    }
  ]
```

3. Next run truffle migrate. Make note of the Contract Address displayed in the terminal.



```
PS D:\factorial> truffle test
Using network 'development'.


Compiling your contracts...
===========================
✓ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
> Compiling .\contracts\Factorial.sol
> Artifacts written to C:\Users\admin\AppData\Local\Temp\test--16692-B3zx4m5zR1Yr
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang
Factorial contract deployed at: 0x82910e8f1Af0aAd0A7c4bf50e9C59612c2e82b41


  Contract: Factorial
Contract Address:  0x82910e8f1Af0aAd0A7c4bf50e9C59612c2e82b41
    ✓ should calculate the factorial of a number correctly


  1 passing (40ms)
```

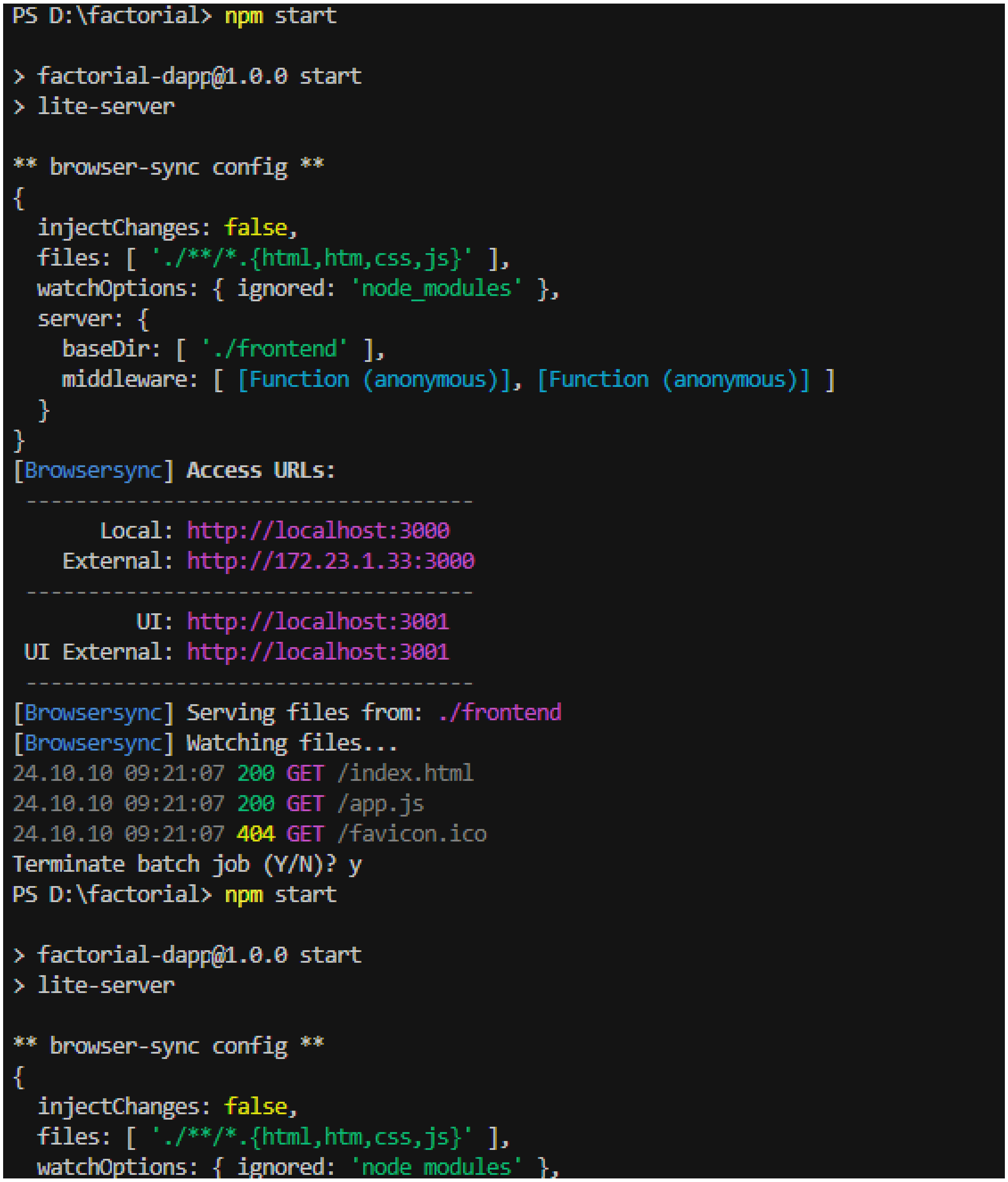
4. Copy the contract address and paste it in the contractAddress constant in the app.js file.



```
const contractAddress = "0x82910e8f1Af0aAd0A7c4bf50e9C59612c2e82b41";
```

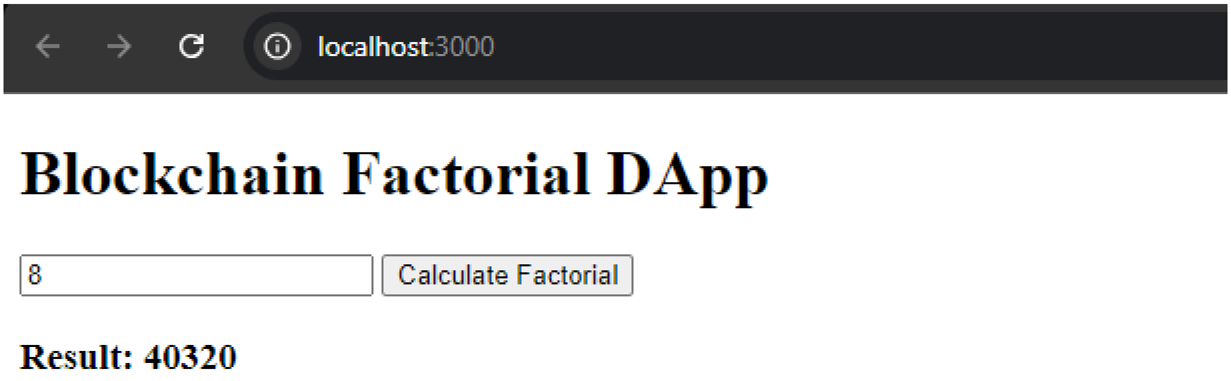
5. Run a truffle test to ensure our contract is correct.

6. Run npm start if everything is correct.



```
PS D:\factorial> npm start

> factorial-dapp@1.0.0 start
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './frontend' ],
    middleware: [ [Function (anonymous)], [Function (anonymous)] ]
  }
}
[Browsersync] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://172.23.1.33:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 --------------------------------------
[Browsersync] Serving files from: ./frontend
[Browsersync] Watching files...
24.10.10 09:21:07 200 GET /index.html
24.10.10 09:21:07 200 GET /app.js
24.10.10 09:21:07 404 GET /favicon.ico
Terminate batch job (Y/N)? y
PS D:\factorial> npm start

> factorial-dapp@1.0.0 start
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
```

7. Sign in to MetaMask and grant the required access.



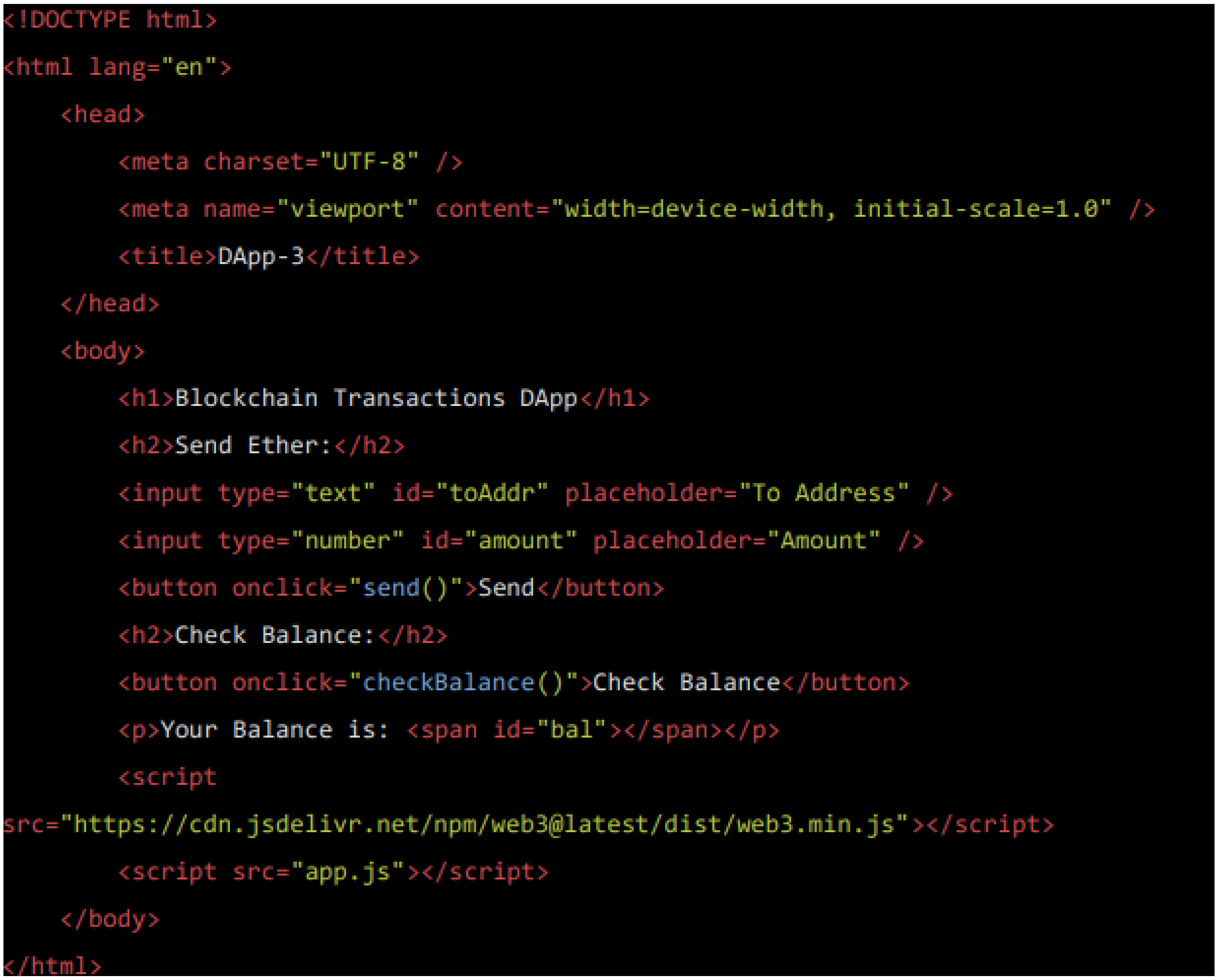
Blockchain Factorial DApp

8

Result: 40320

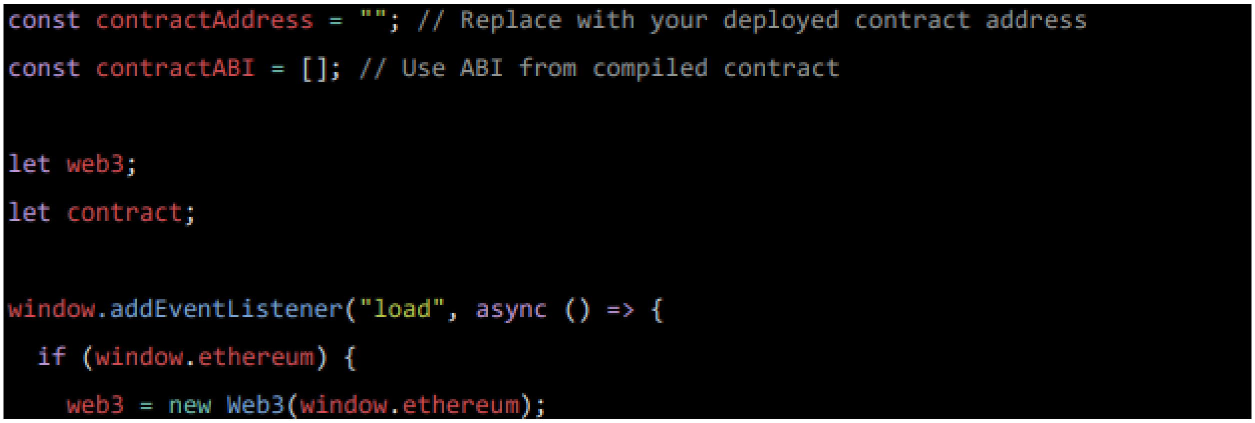
B] Aim: Create a DApp to implement transactions between two accounts.

1. Index.html

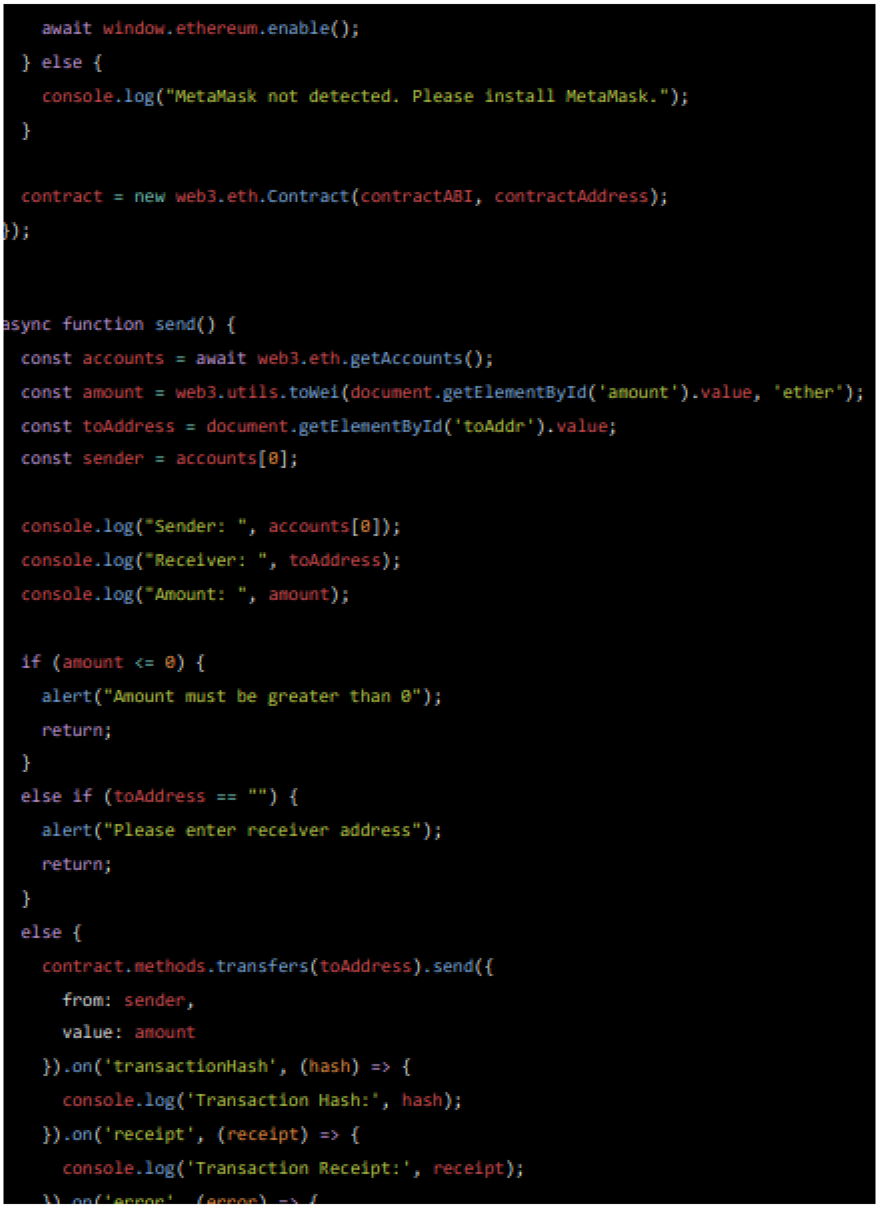


```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>DApp-3</title>
    </head>
    <body>
        <h1>Blockchain Transactions DApp</h1>
        <h2>Send Ether:</h2>
        <input type="text" id="toAddr" placeholder="To Address" />
        <input type="number" id="amount" placeholder="Amount" />
        <button onclick="send()">Send</button>
        <h2>Check Balance:</h2>
        <button onclick="checkBalance()">Check Balance</button>
        <p>Your Balance is: <span id="bal"></span></p>
        <script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
        <script src="app.js"></script>
    </body>
</html>
```

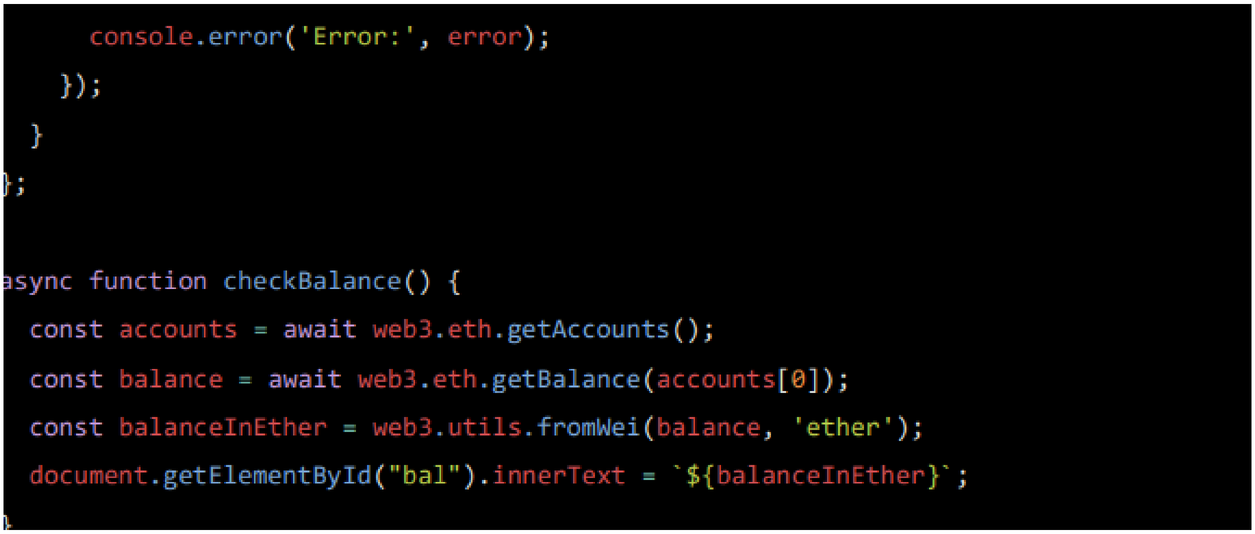
2. app.js



```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
  if (window.ethereum) {
    web3 = new Web3(window.ethereum);
```

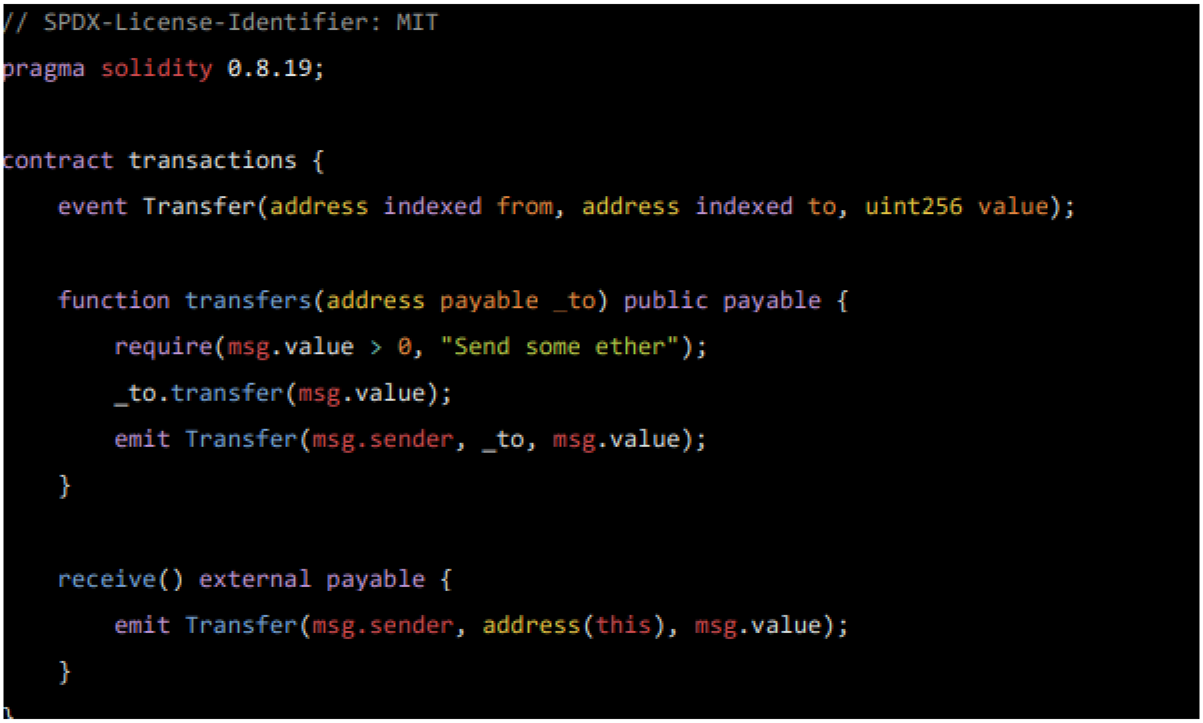


```
    await window.ethereum.enable();
  } else {
    console.log("MetaMask not detected. Please install MetaMask.");
  }

  contract = new web3.eth.Contract(contractABI, contractAddress);
});


async function send() {
  const accounts = await web3.eth.getAccounts();
  const amount = web3.utils.toWei(document.getElementById('amount').value, 'ether');
  const toAddress = document.getElementById('toAddr').value;
  const sender = accounts[0];

  console.log("Sender: ", accounts[0]);
  console.log("Receiver: ", toAddress);
  console.log("Amount: ", amount);

  if (amount <= 0) {
    alert("Amount must be greater than 0");
    return;
  }
  else if (toAddress == "") {
    alert("Please enter receiver address");
    return;
  }
  else {
    contract.methods.transfers(toAddress).send({
      from: sender,
      value: amount
    }).on('transactionHash', (hash) => {
      console.log('Transaction Hash:', hash);
    }).on('receipt', (receipt) => {
      console.log('Transaction Receipt:', receipt);
    }).on('error', (error) => {
```



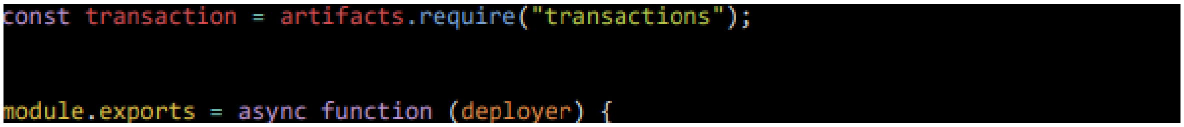
```
      console.error('Error:', error);
    });
  }
};

async function checkBalance() {
  const accounts = await web3.eth.getAccounts();
  const balance = await web3.eth.getBalance(accounts[0]);
  const balanceInEther = web3.utils.fromWei(balance, 'ether');
  document.getElementById("bal").innerText = `${balanceInEther}`;
```

Transactions.sol

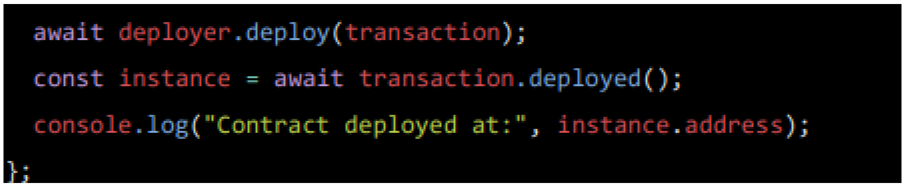


```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.19;

contract transactions {
    event Transfer(address indexed from, address indexed to, uint256 value);

    function transfers(address payable _to) public payable {
        require(msg.value > 0, "Send some ether");
        _to.transfer(msg.value);
        emit Transfer(msg.sender, _to, msg.value);
    }

    receive() external payable {
        emit Transfer(msg.sender, address(this), msg.value);
    }
```

deploy.js

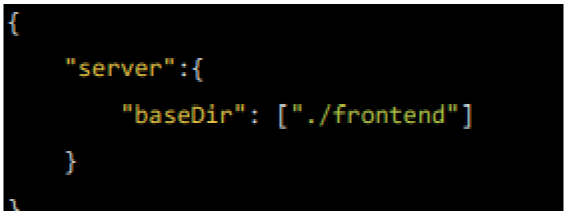


```
const transaction = artifacts.require("transactions");


module.exports = async function (deployer) {
```

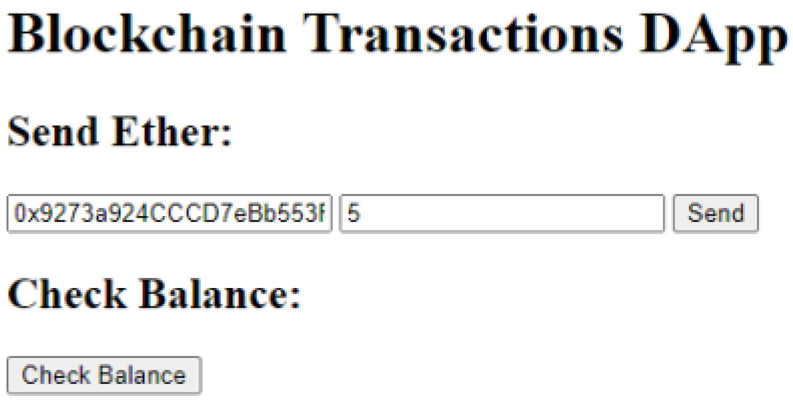


```
  await deployer.deploy(transaction);
  const instance = await transaction.deployed();
  console.log("Contract deployed at:", instance.address);
};
```

Bs-config.json



```
{
    "server":{
        "baseDir": ["./frontend"]
    }
}
```



Blockchain Transactions DApp

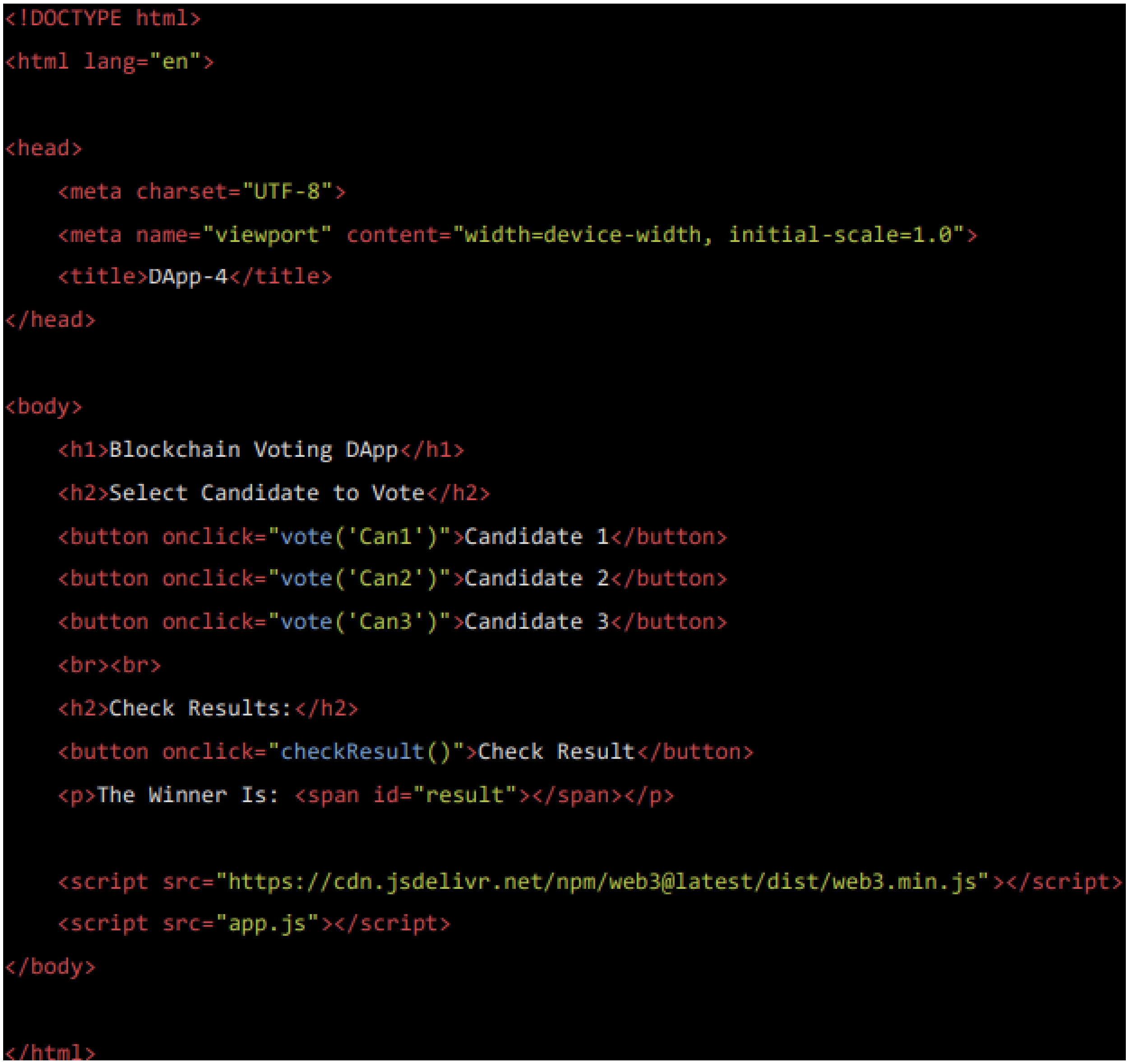
Send Ether:

0x9273a924CCCD7eBb553f | 5

Check Balance:

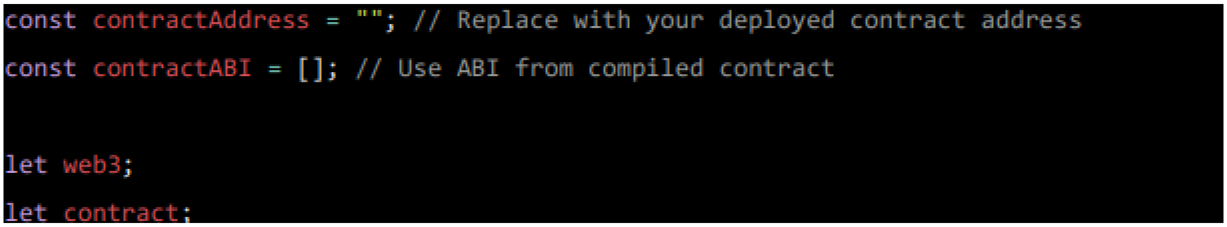
C] Aim: Create a DApp to implement elections.

1. Index.html

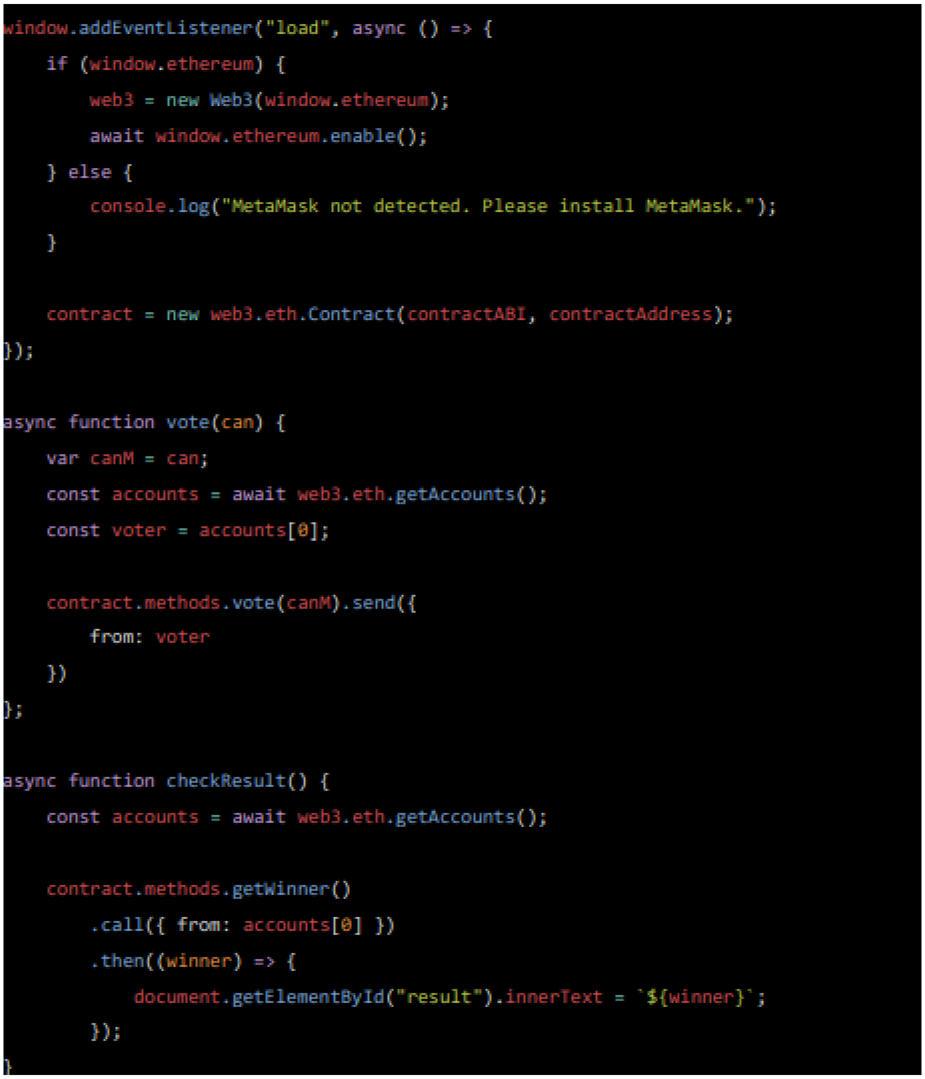


```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DApp-4</title>
</head>

<body>
    <h1>Blockchain Voting DApp</h1>
    <h2>Select Candidate to Vote</h2>
    <button onclick="vote('Can1')">Candidate 1</button>
    <button onclick="vote('Can2')">Candidate 2</button>
    <button onclick="vote('Can3')">Candidate 3</button>
    <br><br>
    <h2>Check Results:</h2>
    <button onclick="checkResult()">Check Result</button>
    <p>The Winner Is: <span id="result"></span></p>

    <script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
    <script src="app.js"></script>
</body>

</html>
```

2. app.js

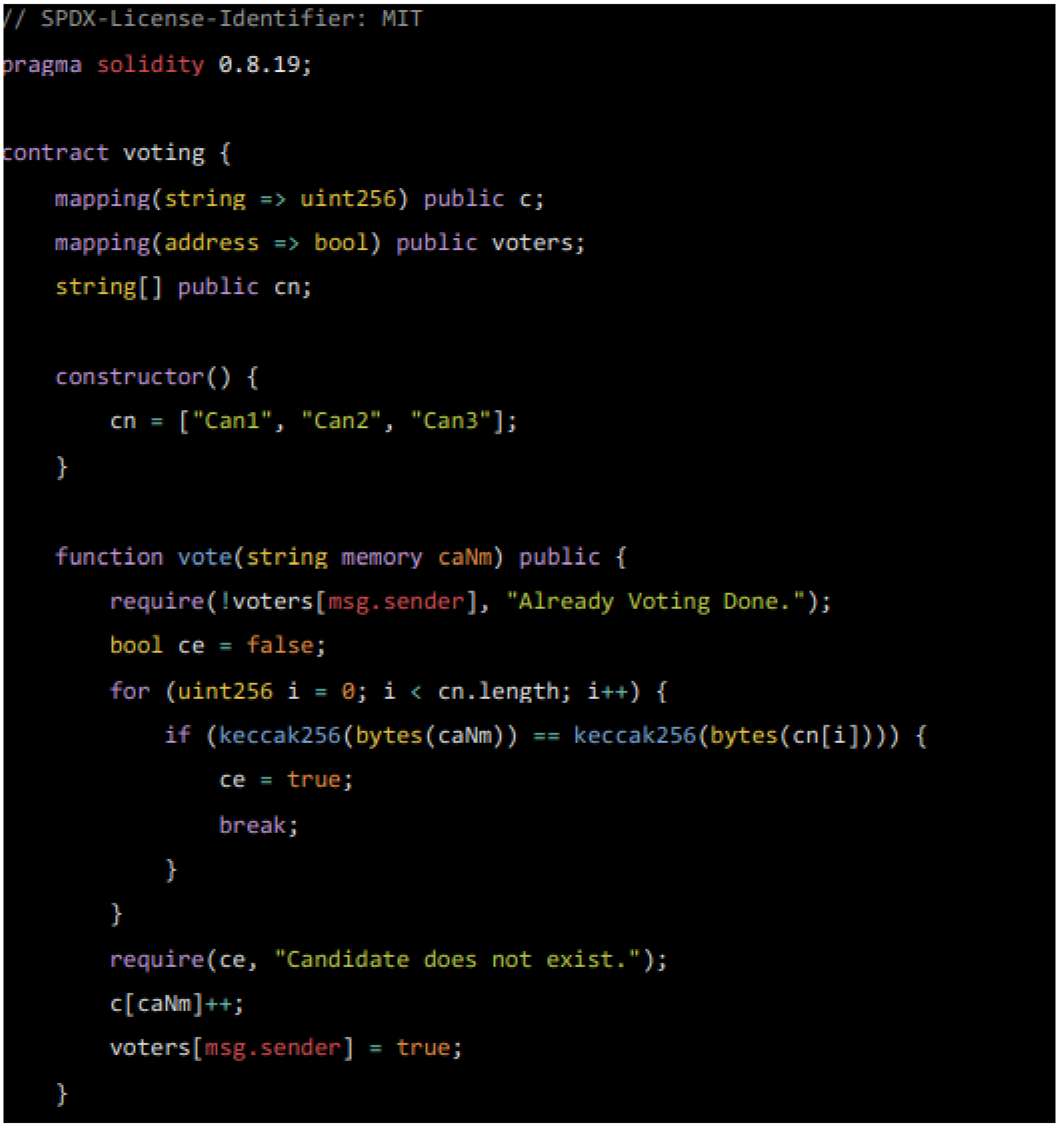


```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;
```

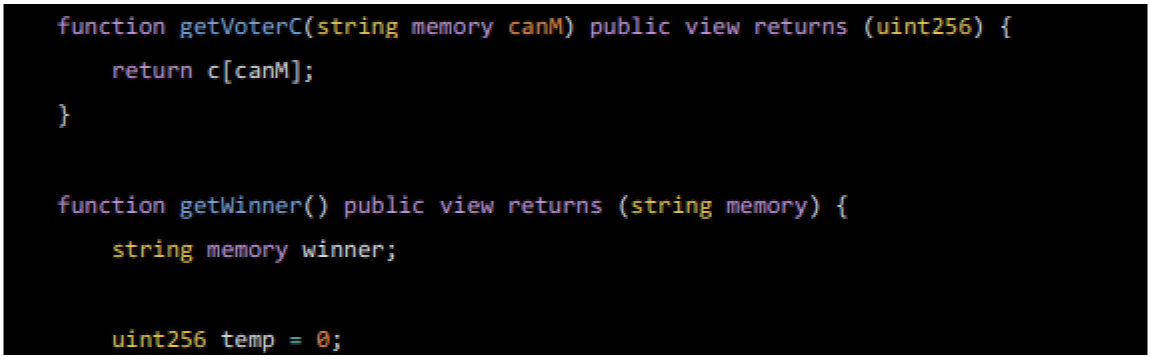


```
window.addEventListener("load", async () => {
    if (window.ethereum) {
        web3 = new Web3(window.ethereum);
        await window.ethereum.enable();
    } else {
        console.log("MetaMask not detected. Please install MetaMask.");
    }

    contract = new web3.eth.Contract(contractABI, contractAddress);
});

async function vote(can) {
    var canM = can;
    const accounts = await web3.eth.getAccounts();
    const voter = accounts[0];

    contract.methods.vote(canM).send({
        from: voter
    })
};

async function checkResult() {
    const accounts = await web3.eth.getAccounts();

    contract.methods.getWinner()
        .call({ from: accounts[0] })
        .then((winner) => {
            document.getElementById("result").innerText = `${winner}`;
        });
}
```

3. Voting.sol

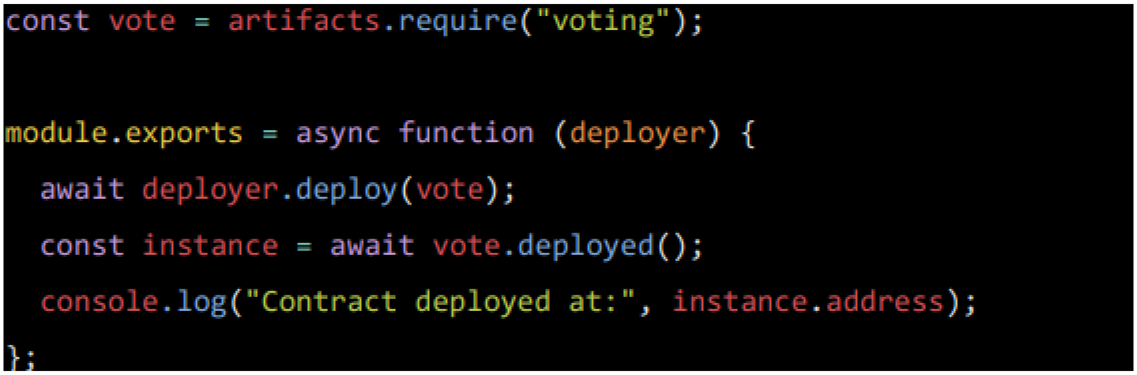


```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.19;

contract voting {
    mapping(string => uint256) public c;
    mapping(address => bool) public voters;
    string[] public cn;

    constructor() {
        cn = ["Can1", "Can2", "Can3"];
    }

    function vote(string memory caNm) public {
        require(!voters[msg.sender], "Already Voting Done.");
        bool ce = false;
        for (uint256 i = 0; i < cn.length; i++) {
            if (keccak256(bytes(caNm)) == keccak256(bytes(cn[i]))) {
                ce = true;
                break;
            }
        }
        require(ce, "Candidate does not exist.");
        c[caNm]++;
        voters[msg.sender] = true;
    }
```



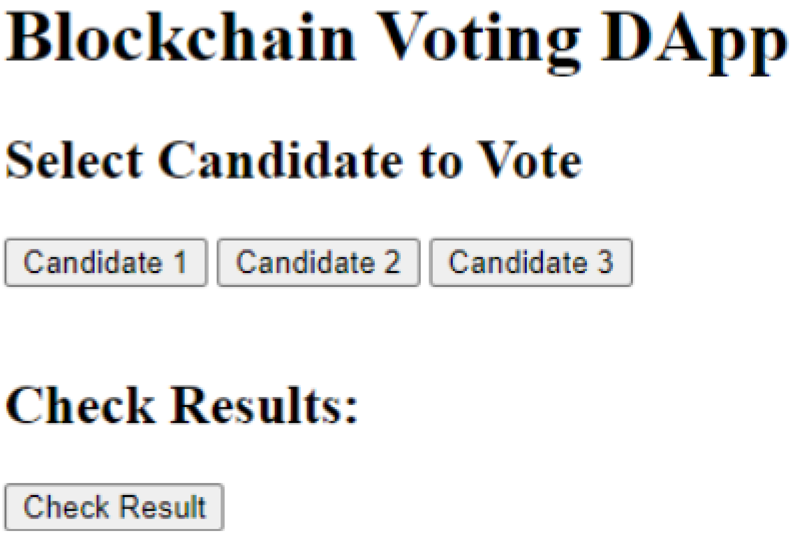
```
    function getVoterC(string memory canM) public view returns (uint256) {
        return c[canM];
    }

    function getWinner() public view returns (string memory) {
        string memory winner;

        uint256 temp = 0;
```

4. deploy.js



```
const vote = artifacts.require("voting");

module.exports = async function (deployer) {
  await deployer.deploy(vote);
  const instance = await vote.deployed();
  console.log("Contract deployed at:", instance.address);
};
```

Output:



Blockchain Voting DApp

Select Candidate to Vote

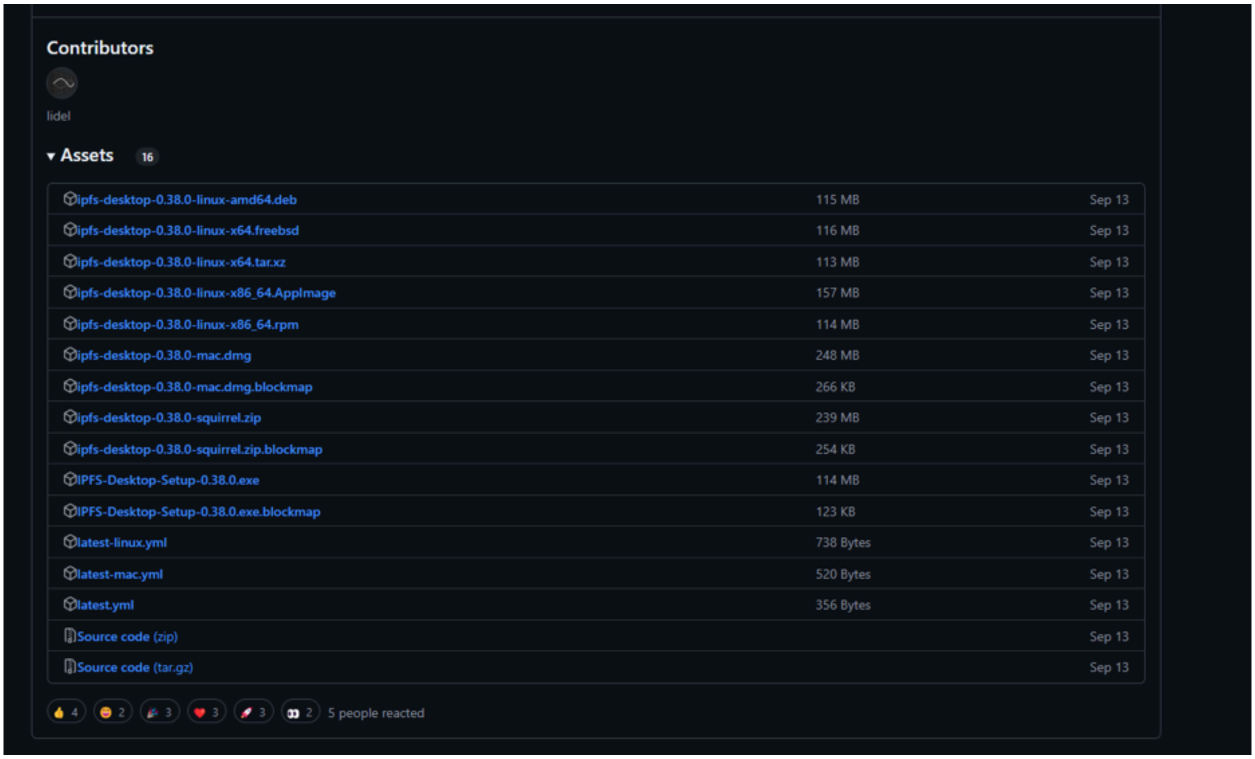
Check Results:

Name: Adiba Mohammed Raza Siddique
Seat No: 31031523033

Practical 13

Aim: Storing and Retrieving files using IPFS.

Step 1: Download and Install IPFS Desktop from here.



Contributors

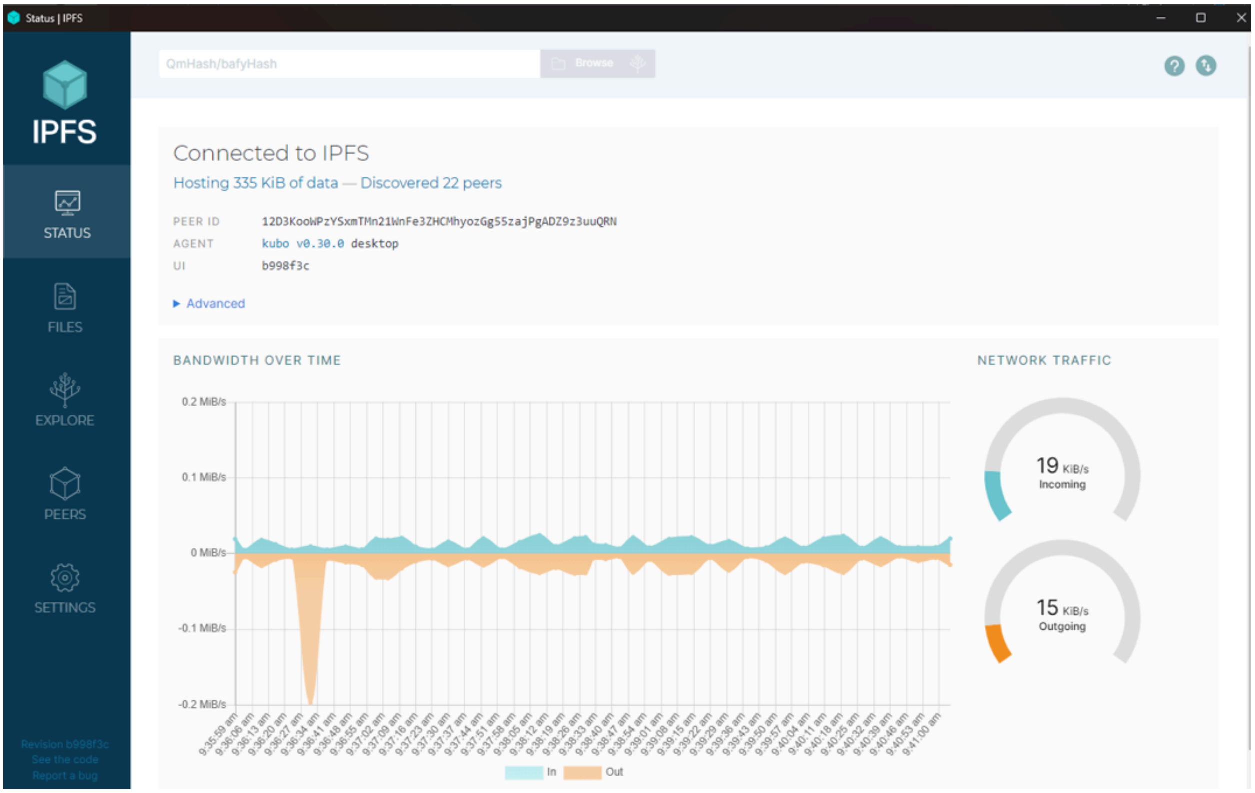
lidel

▾ Assets 16

ipfs-desktop-0.38.0-linux-amd64.deb	115 MB	Sep 13
ipfs-desktop-0.38.0-linux-x64.freebsd	116 MB	Sep 13
ipfs-desktop-0.38.0-linux-x64.tar.xz	113 MB	Sep 13
ipfs-desktop-0.38.0-linux-x86_64.AppImage	157 MB	Sep 13
ipfs-desktop-0.38.0-linux-x86_64.rpm	114 MB	Sep 13
ipfs-desktop-0.38.0-mac.dmg	248 MB	Sep 13
ipfs-desktop-0.38.0-mac.dmg.blockmap	266 KB	Sep 13
ipfs-desktop-0.38.0-squirrel.zip	239 MB	Sep 13
ipfs-desktop-0.38.0-squirrel.zip.blockmap	254 KB	Sep 13
IPFS-Desktop-Setup-0.38.0.exe	114 MB	Sep 13
IPFS-Desktop-Setup-0.38.0.exe.blockmap	123 KB	Sep 13
latest-linux.yml	738 Bytes	Sep 13
latest-mac.yml	520 Bytes	Sep 13
latest.yml	356 Bytes	Sep 13
Source code (zip)		Sep 13
Source code (tar.gz)		Sep 13

👍 4 😄 2 🎉 3 ❤️ 3 🚀 3 👀 2 5 people reacted

Step 2: Click on files and import a sample file.

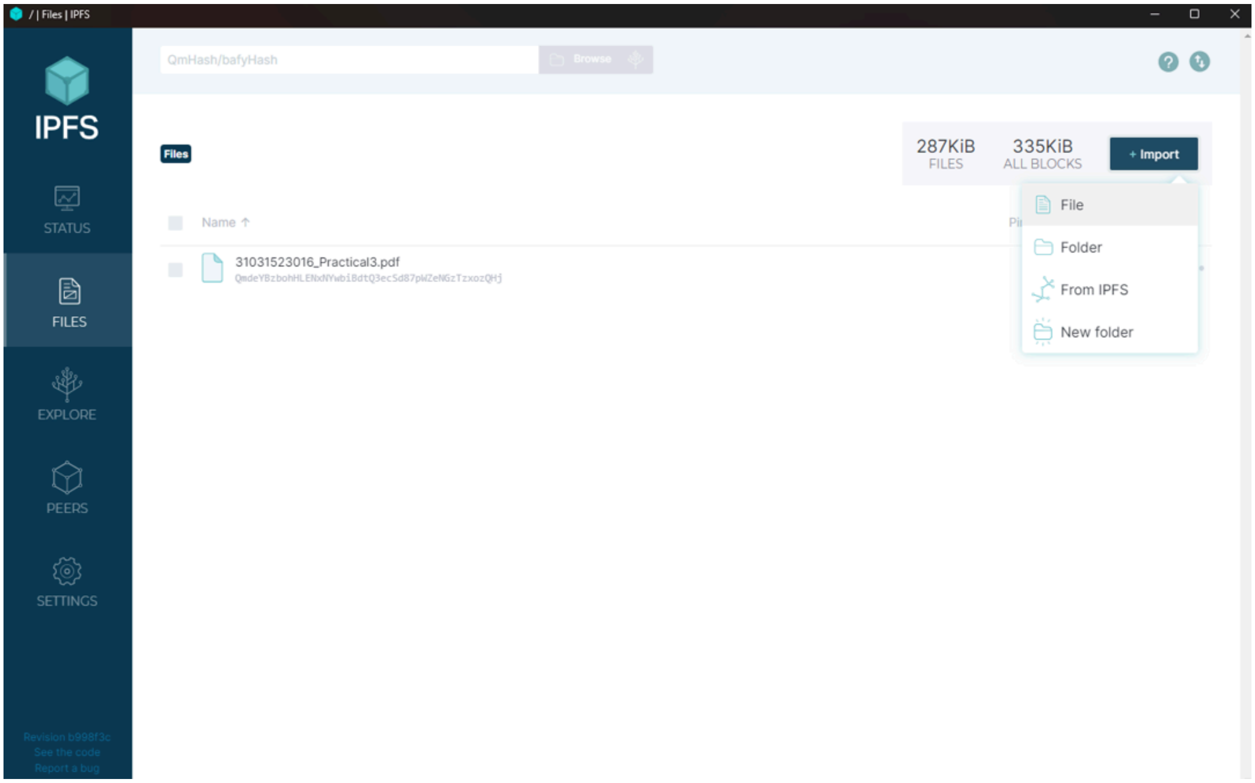


Connected to IPFS

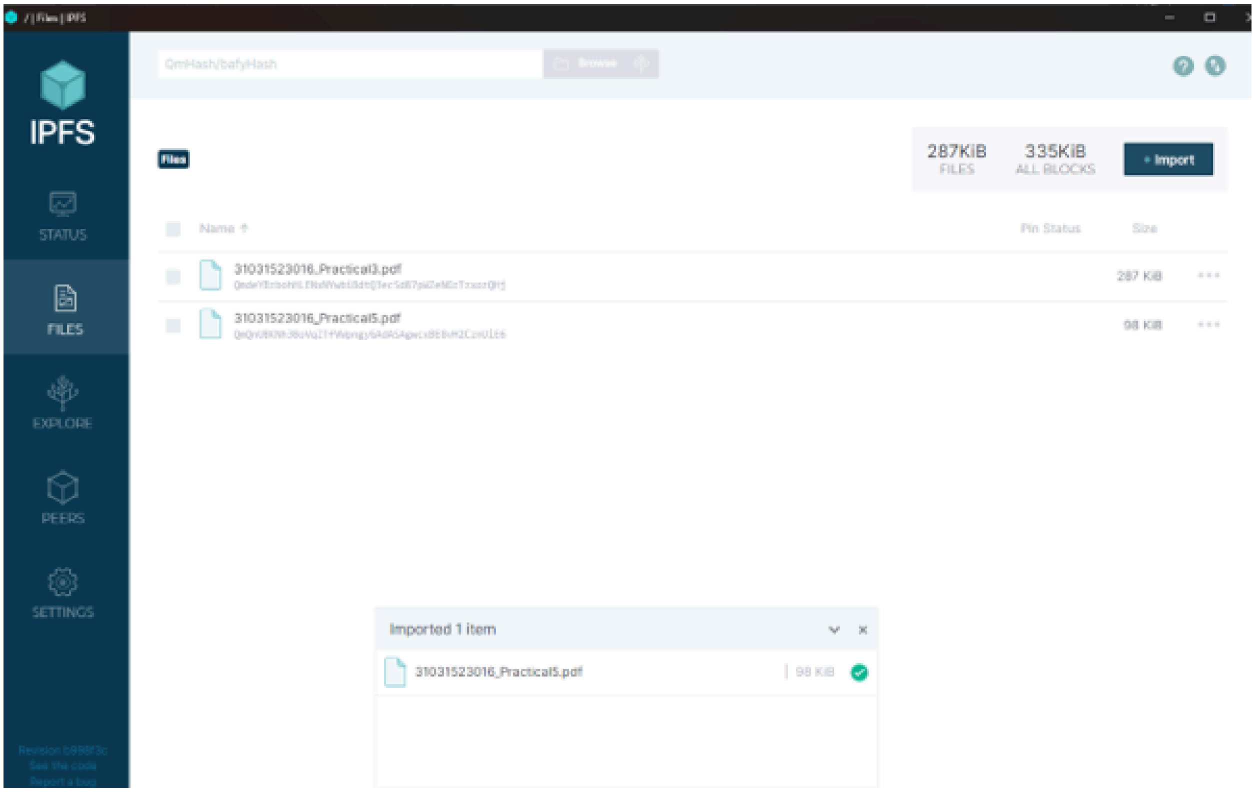
Hosting 335 KiB of data — Discovered 22 peers

```
PEER ID    12D3KooWPzYSxmTMn21WnFe3ZHCMhyozGg55zajPgADZ9z3uuQRN
AGENT      kubo v0.30.0 desktop
UI         b998f3c
```

▸ Advanced



31031523016_Practical3.pdf
QmdeYBzbohHLENxNYwbiBdtQ3ec5d87pWZeNGzTzxozQHj

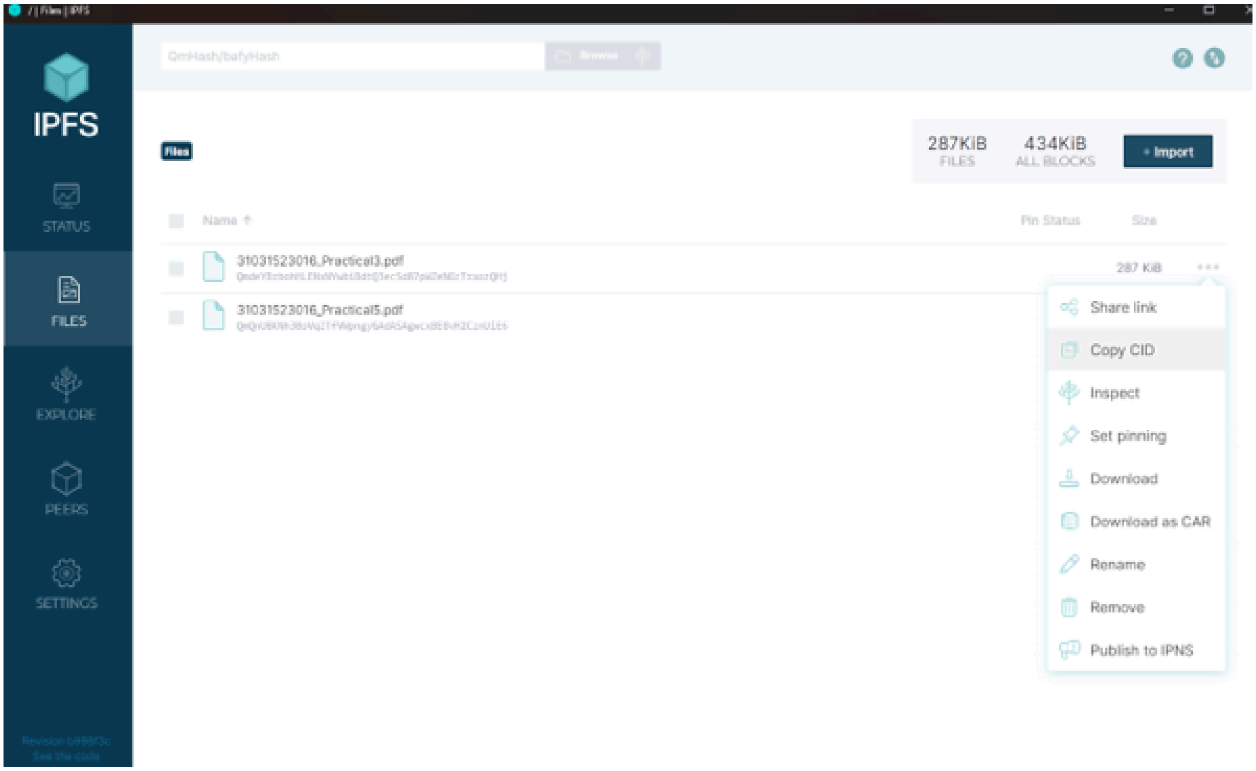


Name ↑	Pin Status	Size
31031523016_Practical3.pdf [illegible]		287 KiB
31031523016_Practical5.pdf [illegible]		98 KiB

287KiB FILES
335KiB ALL BLOCKS

Imported 1 item
31031523016_Practical5.pdf | 98 KiB ✓

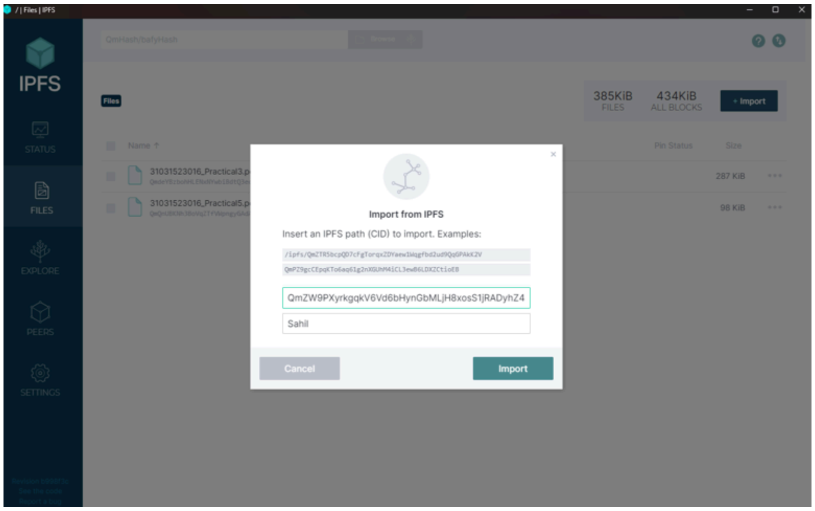
Step 3: Click on 3 dots and copy CID



Name ↑	Pin Status	Size
31031523016_Practical3.pdf [illegible]		287 KiB
31031523016_Practical5.pdf [illegible]		

287KiB FILES
434KiB ALL BLOCKS

Step 4: Share the CID to someone else to open the shared file.



Import from IPFS

Insert an IPFS path (CID) to import. Examples:

```
/ipfs/QmZTR5bcpQD7cFgTorqxZDYaew1Wqgfbd2ud9QqGPAkK2V
QmPZ9gcCEpqKTo6aq61g2nXGUhM4iCL3ewB6LDXZCtioEB
```

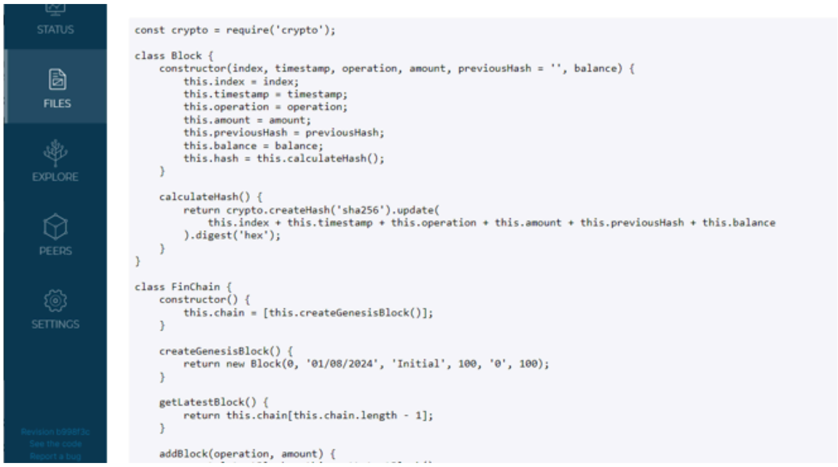
QmZW9PXyrkgqkV6Vd6bHynGbMLjH8xosS1jRADyhZ4

Sahil

Cancel | Import

Step 5: Click on import à Import from IPFS

Step 6: The imported file will be visible.



```
const crypto = require('crypto');

class Block {
    constructor(index, timestamp, operation, amount, previousHash = '', balance) {
        this.index = index;
        this.timestamp = timestamp;
        this.operation = operation;
        this.amount = amount;
        this.previousHash = previousHash;
        this.balance = balance;
        this.hash = this.calculateHash();
    }

    calculateHash() {
        return crypto.createHash('sha256').update(
            this.index + this.timestamp + this.operation + this.amount + this.previousHash + this.balance
        ).digest('hex');
    }
}

class FinChain {
    constructor() {
        this.chain = [this.createGenesisBlock()];
    }

    createGenesisBlock() {
        return new Block(0, '01/08/2024', 'Initial', 100, '0', 100);
    }

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

    addBlock(operation, amount) {
```