|  |
| --- |
| **PRACTICAL REPORT**  **ON**  **BLOCKCHAIN**    **SUBMITTED BY**  **AKASH K MISHRA**    **ROLL NO: 23**    **SUBMITTED TO**  **Mr. Sameer Kamble**  **MSc. (INFORMATION TECHNOLOGY) SEM -IV**  **2021 – 2022**    **CONDUCTED AT**  **CHIKITSAK SAMUHA’S**  **S. S. & L.S. PATKAR COLLEGE OF ARTS & SCIENCE AND**  **V. P. VARDE COLLEGE OF COMMERCE & ECONOMICS**  **Affiliated to University of Mumbai**  **GOREGAON (W). MUMBAI -400062** |
| |  |  |  | | --- | --- | --- | | **Practical No.** | **Practical Aim**  **INDEX** | **Dates** | | **1A.** | A Simple Client Class That Generates The Private And Public Keys By Using The Built In Python RSA Algorithm And Test | 10-01-2022 | | **1B.** | A Transaction Class To Send And Receive Money And Test It | 10-01-2022 | | **1C.** | Create Multiple Transactions And Display Them | 17-01-2022 | | **1D.** | Create A Blockchain, A Genesis Block And Execute It | 17-01-2022 | | **1E.** | Create A Mining Function And Test It. | 31-01-2022 | | **1F.** | Add Blocks To The Miner And Dump The Blockchain | 31-01-2022 | | **2** | Implement And Demonstrate The Use Of Solidity Programming | 02-03-2022 | | **2A.** | Your First Solidity Smart Contract(Counter Program) |  | | **2B.** | To Create And Explore Types Of Variables With Varying Data Types In Solidity Programming(Variables) |  | | **2C.** | Operators In Solidity |  | | **3** | Implement And Demonstrate the Use of Solidity Programming | 02-03-2022 | | **3A** | Loops In Solidity |  | | **3B.** | Decision Making In Solidity |  | | **3C.** | String Application Strings |  | | **4** | Implement And Demonstrate the Use of Solidity Programming | 07-03-2022 | | **4A.** | Solidity Arrays |  | | **4B.** | Enums In Solidity |  | | **4C.** | Structure In Solidity |  | | **5** | Implement And Demonstrate the Use of Solidity Programming | 07-03-2022 | | **5A.** | Solidity Mappings |  | | **5B.** | Special Variables In Solidity |  | | **5C.** | To Create A Complete Solidity Smart Contract |  | | **6** | Implement And Demonstrate the Use of Solidity Programming | 08-03-2022 | | **6A.** | Introduction To Functions In Solidity |  | | **6B.** | Function Modifiers In Solidity |  | | **6C.** | View And Pure Functions In Solidity |  | | **7** | Implement And Demonstrate the Use of Solidity Programming | 08-03-2022 | | **7A.** | Fallback Functions In Solidity |  | | **7B.** | Function Overloading In Solidity |  | | **7C.** | Mathematical & Cryptographic Functions In Solidity |  | | **8** | Implement And Demonstrate the Use of Solidity Programming | 09-03-2022 | | **8A.** | Withdrawal Patterns In Solidity |  | | **8B.** | Restricted Access In Solidity |  | | **8C.** | Contracts In Solidity |  | | **9** | Implement And Demonstrate the Use of Solidity Programming | 21-03-2022 | | **9A.** | Inheritance In Solidity |  | | **9B.** | Working Of Libraries In Solidity |  | | **9C.** | Working Of Assembly In Solidity |  | | **9D.** | Error Handling In In Solidity |  | |

# Practical no : 1

**1A. A simple client class that generates the private and public keys by using the builtin Python RSA algorithm and test it Code:**

import Crypto from Crypto.PublicKey import RSA from Crypto.Signature import PKCS1\_v1\_5

import binascii

class Client:

def \_\_init\_\_(self):

random = Crypto.Random.new().read self.\_private\_key = RSA.generate(1024,random) self.\_public\_key = self.\_private\_key.publickey() self.\_signer = PKCS1\_v1\_5.new(self.\_private\_key)

@property def identity(self):

return binascii.hexlify(self.\_public\_key.exportKey(format='DER')).decode('ascii')

Ramesh = Client() print(Ramesh.identity) **Output:**



**1B. A transaction class to send and receive money and test it.**

**Code:**

import datetime import collections

from Crypto.Hash import SHA

# from client import \*

class Transaction: def \_\_init\_\_(self, sender, recipient, value):

self.sender = sender self.recipient = recipient self.value = value

self.time = datetime.datetime.now()

def to\_dict(self): if self.sender == "Genesis":

identity = "Genesis" else:

identity = self.sender.identity

return collections.OrderedDict({

'sender':self.sender,

'recipient':self.recipient,

'value':self.value,

'time':self.time})

def sign\_transaction(self):

private\_key = self.sender.\_private\_key signer = PKCS1\_v1\_5.new(private\_key) h = SHA.new(str(self.to\_dict()).encode('utf8')) return binascii.hexlify(signer.sign(h)).decode('ascii')

Dinesh = Client()

t = Transaction(Ramesh,Dinesh.identity, 5.0) signature = t.sign\_transaction() print(signature)

**Output:**



**1C. Create multiple transactions and display them Code:**

def display\_transaction(transaction): dict = transaction.to\_dict() print("sender: " + str(dict['sender'])) print('------')

print("Recipient: " + str(dict['recipient'])) print('------')

print("Value: " + str(dict['value'])) print('------')

print("Time: " + str(dict['time']))

print('------')

transaction = []

Seema = Client()

Vijay = Client()

t1 = Transaction(Dinesh, Ramesh.identity, 15.0) t1.sign\_transaction()

transaction.append(t1)

t2 = Transaction(Dinesh, Seema.identity, 6.0) t2.sign\_transaction() transaction.append(t2)

t3 = Transaction(Ramesh, Vijay.identity, 2.0) t3.sign\_transaction()

transaction.append(t3)

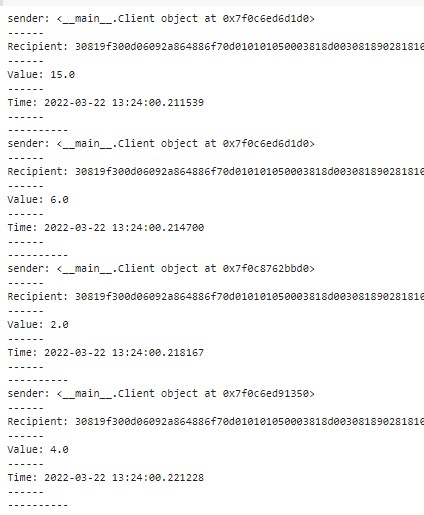
t4 = Transaction(Seema, Ramesh.identity, 4.0)

t4.sign\_transaction()

transaction.append(t4)

for transaction in transaction: display\_transaction(transaction) print("----------")

**Output:**



**1D. Create a blockchain, a genesis block and execute it.**

**Code:**

import datetime

import hashlib

class Block:

def \_\_init\_\_(self, previous\_block\_hash, date, timestamp): self.previous\_block\_hash = previous\_block\_hash

self.date = date self.timestamp = timestamp self.hash = self.get\_hash()

@staticmethod def create\_genesis\_block():

return Block("0", "0", datetime.datetime.now())

def get\_hash(self):

header = (str(self.previous\_block\_hash) + str(self.date) + str(self.timestamp)) inner\_hash = hashlib.sha256(header.encode()).hexdigest().encode() comp\_hash = hashlib.sha256(inner\_hash).hexdigest() return comp\_hash

number\_of\_blocks = 14

Blockchain = [Block.create\_genesis\_block()] print('Akash Block is Created') print("Hash: %s" % Blockchain[0].hash)

for i in range(1, number\_of\_blocks):

Blockchain.append(Block(Blockchain[i - 1].hash, "Block number %d" %i, datetime.datetime.now())

)

print("Akash %d block created" %i)

print("Hash: %s" % Blockchain[-1].hash)

**Output:**



**1E. Create a mining function and test it.**

**Code:**

from hashlib import sha256

MAX\_NONCE = 10000000000

def SHA256(text):

return sha256(text.encode('ascii')).hexdigest()

def mine(block\_number, transactions, previous\_hash, prefix\_zeros):

prefix\_str = '0'\*prefix\_zeros for nonce in range(MAX\_NONCE):

text = str(block\_number) + transactions + previous\_hash+ str(nonce) new\_hash = SHA256(text) if new\_hash.startswith(prefix\_str):

print(f"Yahhh!..... Successfully mined bitcoins with nonce value: {nonce}") return new\_hash

raise BaseException(f"Couldn't find correct has after trying {MAX\_NONCE} times")

if \_\_name\_\_ == '\_\_main\_\_':

transactions = ''' Dhaval->Bhavin->20,

Mando->Cara->45

'''

difficulty = 4 import time start = time.time() print("Start Mining")

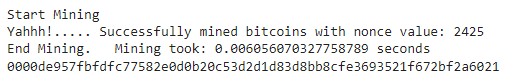
new\_hash = mine(5, transactions, '0000000xa036944e29568d0cff17edbe038f81208fecf9a66be9a2b

8321c6ec7', difficulty)

total\_time = str((time.time() - start))

print(f"End Mining. Mining took: {total\_time} seconds") print(new\_hash)

**Output:**



**1F. Add blocks to the miner and dump the blockchain.**

**Code:**

import datetime import hashlib

class Block: blockNo = 0 data = None next = None hash = None nonce = 0 previous\_hash = 0x0

timestamp = datetime.datetime.now()

def \_\_init\_\_(self, data):

self.data = data

def hash(self): h = hashlib.sha256() h.update( str(self.nonce).encode('utf-8') + str(self.data).encode('utf-8') + str(self.previous\_hash).encode('utf-8') + str(self.timestamp).encode('utf-8') +

str(self.blockNo).encode('utf-8')

)

return h.hexdigest()

def \_\_str\_\_(self):

return "Block Hash: " + str(self.hash()) + "\nBlock No: " + str(self.blockNo) + "\nBlock Data: " + str(self.data) + "\nHashes: " + str(self.nonce) + "\n-------------------"

class Blockchain:

diff = 20

maxNonce = 2\*\*32

target = 2 \*\*(256-diff)

block = Block("Genesis") dummy = head = block

def add(self,block):

block.previous\_hash = self.block.hash() block.blockNo = self.block.blockNo + 1

self.block.next = block self.block = self.block.next def mine (self, block): for n in range(self.maxNonce): if int(block.hash(), 16) <= self.target:

self.add(block) print(block) break else:

block.nonce += 1

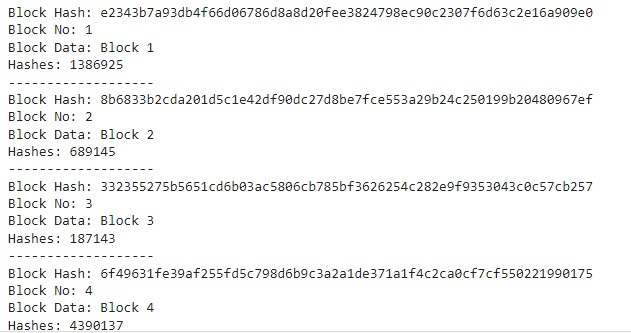
blockchain = Blockchain()

for n in range (10):

blockchain.mine(Block("Block " + str(n+1)))

while blockchain.head != None: print(blockchain.head) blockchain.head = blockchain.head.next

**Output:**

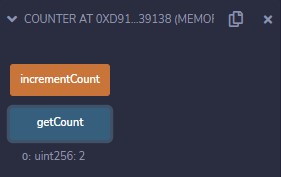


# Practical no : 2

**2a. Your First Solidity Smart Contract (Counter Program) Code:**

|  |
| --- |
| pragma solidity ^0.6.0; contract Counter { uint count; constructor() public{ count = 0;  } function getCount() public view returns(uint){ return count  } function incrementCount() public{ count = count +1  }  } |

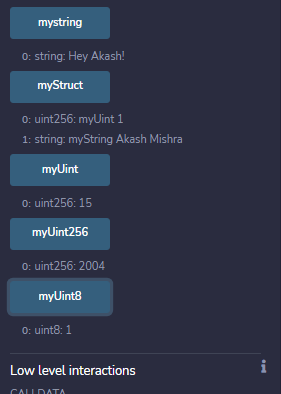
**Output:**

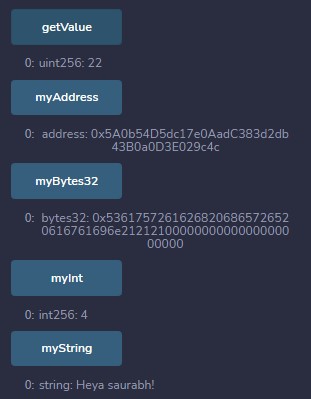


**2b. To create and explore types of variables with varying data types in solidity programming.**

**Code:**

|  |
| --- |
|  |

**Output:**



**2C. Operators in solidity Code:**

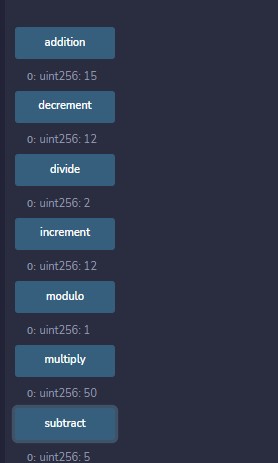
pragma solidity ^0.6.0

contract Arithmetic { function addition() public view returns (uint){ uint a = 10; uint b = 5; uint result = a+b; return result;

} function subtract() public view returns (uint){ uint a = 10; uint b = 5;

|  |
| --- |
| uint result = a-b; return result;  } function multiply() public view returns (uint){ uint a = 10; uint b = 5; uint result = a\*b; return result;  } function divide() public view returns (uint){ uint a = 10; uint b = 5; uint result = a/b; return result;  } function modulo() public view returns (uint){ uint a = 10; uint b = 3; uint result = a%b; return result;  } function increment() public view returns (uint){ uint a = 12; uint result = a++; return result;  } function decrement() public view returns (uint){ uint a = 12; uint result = a--; return result;  }  } |

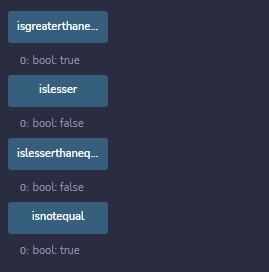
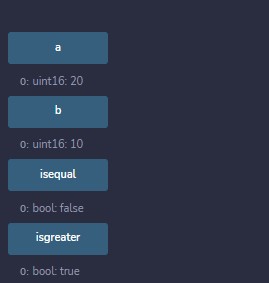
**Output:**



**Comparison Code:**

|  |
| --- |
| pragma solidity ^0.6.0;  contract Comparison{    //Declaring variables uint16 public a = 20; uint16 public b = 10;    //initializing a variable //with bool equal result bool public isequal = a==b;    //initializing a variable //with bool not equal result bool public isnotequal = a!=b;    //initializing a variable  //with bool greater than result bool public isgreater = a>b;    //initializing a variable //with bool less than result bool public islesser = a<b;    //initializing a variable //with bool greater than equal to result bool public isgreaterthanequal = a>=b;    //initializing a variable  //with bool less than equal to result bool public islesserthanequal = a<=b;  } |

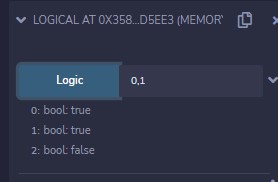
**Output:**



**Logical Operator Code:**

|  |
| --- |
| pragma solidity ^0.6.0;  contract Logical{  constructor() public{  } function Logic(bool a, bool b) public view returns(bool, bool, bool){  //Logical AND operator bool and = a&&b;    //Logical OR operator bool or - a||b    //Logical NOT operator bool not = !a;  return (and, or, not)  }  } |

**Output:**

 **Bitwise operators Code:**

|  |
| --- |
| pragma solidity ^0.6.0;    contract Bitwise{ uint16 public a = 20; uint16 public b = 10;    uint16 public and = a & b;  uint16 public or = a | b;  uint16 public Xor = a ^ b; |
| uint16 public leftshift = a<<b;  uint16 public rightshift = a>>b;  uint16 public not = ~a;  } |

**Output:**

**Assignment Code:**



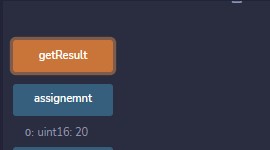
pragma solidity ^0.6.0;

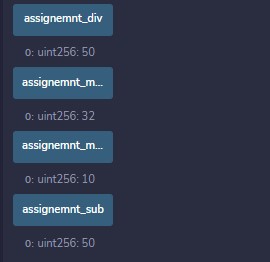
contract Assignment{ uint16 public assignemnt = 20; uint public assignemnt\_add = 50; uint public assignemnt\_sub = 50; uint public assignemnt\_mul = 10; uint public assignemnt\_div = 50; uint public assignemnt\_mod = 32;

function getResult() public{ assignemnt\_add += 10; assignemnt\_sub -= 20; assignemnt\_mul \*= 10; assignemnt\_div /= 10; assignemnt\_mod %= 20; return;

} }

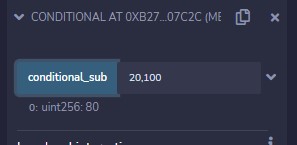
**Output:**



 **Conditional Operator Code:**

|  |
| --- |
| pragma solidity ^0.6.0;  contract Conditional{  function conditional\_sub(uint a, uint b) public view returns(uint){ uint result = (a>b? a-b : b-a); return result;  }  } |

**Output:**



**Practical no 3**

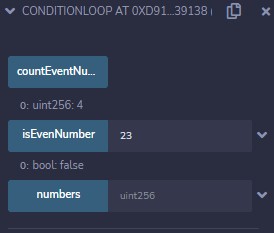
**Implement and demonstrate the use of solidity programming**

**3A. Loops in Solidity**

**For Loop Code:**

|  |
| --- |
| pragma solidity ^0.6.0; contract Conditionloop{ uint[] public numbers = [1,2,3,4,5,6,7,8,9,19];  function countEventNumbers() public view returns (uint){ uint count = 0;  for(uint i = 0; i < numbers.length; i++){ if(isEvenNumber(numbers[i])){ count++;  } } return count;  } function isEvenNumber(uint \_number) public view returns(bool){ if(\_number %2 == 0){ return true;  } return false;  }  } |

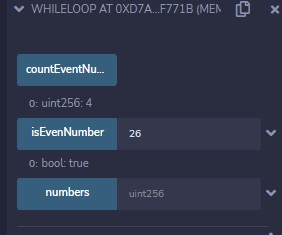
**Output:**



**While Loop Code:**

|  |
| --- |
| pragma solidity ^0.6.0;  contract Whileloop  { |
| uint[] public numbers = [1,2,3,4,5,6,7,8,9,19];  function countEventNumbers() public view returns (uint)  { uint count = 0; uint n = 0; while(n < numbers.length)  { if(isEvenNumber(numbers[n])){ count++;  } n = n+1;  } return count;  } function isEvenNumber(uint \_number) public view returns(bool){ if(\_number %2 == 0){ return true;  } return false;  }  } |

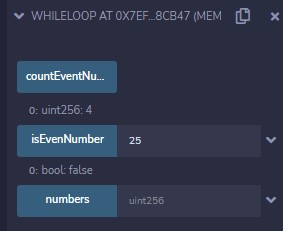
**Output:**



**Do while Loop Code:**

|  |
| --- |
| pragma solidity ^0.6.0;    contract Whileloop  { uint[] public numbers = [1,2,3,4,5,6,7,8,9,19];    function countEventNumbers() public view returns (uint)  { uint count = 0; |
| uint n = 0; do { if(isEvenNumber(numbers[n])){ count++;  } n = n+1;  }while(n < numbers.length); return count;  } function isEvenNumber(uint \_number) public view returns(bool){ if(\_number %2 == 0){ return true;  } return false;  }  } |

**Output:**



**3B.** **Decision making in Solidity**

**If statement, If……else statement and If…. else…. if statement Code:**

|  |
| --- |
| pragma solidity ^0.6.0;    contract DecisionCode{ uint i =8;    function only\_if() public view returns(bool)  { if(i<10) {  return true;  } return false;  } function if\_else() public view returns(bool) |
| { if(i % 2 == 0)  { return true;  } else { return false;  }  } string result; function if\_else\_if() public returns(string memory)  { if(i <10)  { result = "less than 10";  } else if(i == 10)  { result = "equal to 10";  } else { result = "greater than 10";  } return result;  }  } |

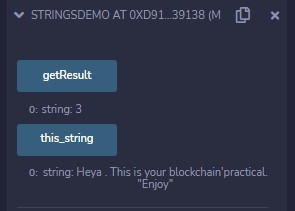
**Output:**



**3C. String Application Code:**

|  |
| --- |
| pragma solidity ^0.6.0;  contract Stringsdemo{  constructor() public {  } function this\_string() public view returns(string memory){ string memory str="Heya . \tThis is your blockchain\'practical. \n  \"Enjoy\"";  return str;  } function getResult() public view returns(string memory){ uint a = 1; uint b = 2; uint result = a+b; return integerToString(result);  } function integerToString(uint \_i) internal pure returns (string memory){ if(\_i == 0)  { return "0";  } uint j = \_i; uint len;  while(j != 0)  { len++; j /= 10;  }    bytes memory bstr = new bytes(len); uint k = len -1;    while (\_i != 0)  { bstr[k--] = byte(uint8(48 + \_i % 10));  \_i /= 10;  } return string(bstr);  }  } |

**Output:**



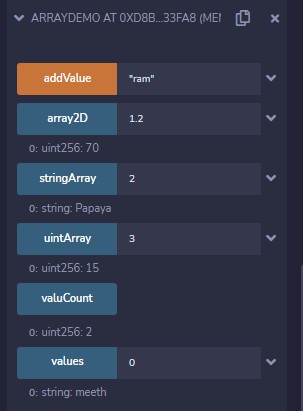
## Practical no 4

**Aim : Implement and demonstrate the use of solidity programming**

**4A. Arrays Code:**

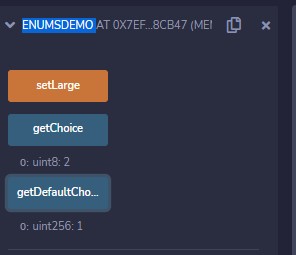
|  |
| --- |
| pragma solidity ^0.6.0;  contract Arraydemo  { uint[] public uintArray = [22,10,1,15];  string[] public stringArray = ["Apple","Watermelon","Papaya","Kiwi","Blue Berry"]; uint[][] public array2D = [ [10,20,30], [90,80,70] ]; string[] public values; function addValue(string memory \_value) public  { values.push(\_value);  } function valuCount() public view returns(uint)  { return values.length;  }  } |

**Output:**



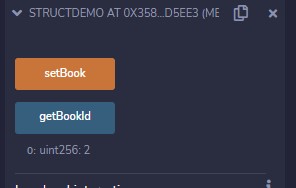
**4B. Enums Code:**

|  |  |
| --- | --- |
| pragma solidity ^0.6.0;  contract Enumsdemo  {  enum FreshJuiceSize{ SMALL, MEDIUM, LARGE}  FreshJuiceSize choice;  FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;  function setLarge() public  { choice = FreshJuiceSize.LARGE;  } function getChoice() public view returns (FreshJuiceSize)  { return choice;  } function getDefaultChoice() public pure returns (uint)  { return uint(defaultChoice);  }  } | |
| **Output:** |  |



**4C. Structure Code:**

|  |  |
| --- | --- |
| pragma solidity ^0.6.12;  contract StructDemo  { struct Book | |
| { string title; string author; uint book\_id;  }  Book book; function setBook() public{ book = Book("Learn Java", "TP",1); book = Book("Learn C#","CP",2);  } function getBookId() public view returns(uint)  { return book.book\_id;  }  } | |
| **Output:** |  |

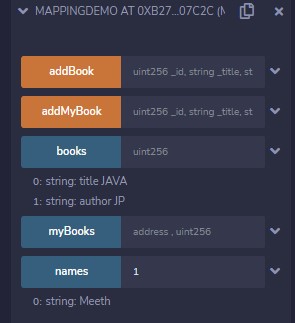


## Practical no : 5

**Aim : Implement and demonstrate the use of solidity programming.**

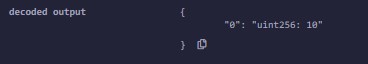
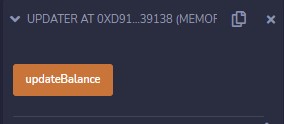
**5A. Mappings Code:**

|  |  |
| --- | --- |
| pragma solidity ^0.6.0;  contract Mappingdemo  { mapping(uint => string) public names; mapping(uint => Book) public books; mapping(address => mapping(uint => Book)) public myBooks;  struct Book{ string title; string author;  } constructor() public  { names[1] = "Meeth"; names[2] = "Ram"; names[3] = "Shyam";  } function addBook(uint \_id, string memory \_title, string memory \_author) public { books[\_id] = Book(\_title, \_author);  }  function addMyBook(uint \_id, string memory \_title, string memory \_author) public { myBooks[msg.sender][\_id] = Book(\_title,\_author);  }  } | |
| **Output:** |  |



**5B. Special Variables Code:**

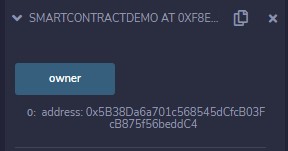
|  |  |
| --- | --- |
| pragma solidity ^0.6.0; contract SpecialVariabledemo  { mapping(address => uint) public balances;  function updateBalance(uint newBalance) public  { balances[msg.sender] = newBalance;  } } contract Updater  { function updateBalance() public returns(uint) {  SpecialVariabledemo ledgerBalance = new SpecialVariabledemo(); ledgerBalance.updateBalance(10); return ledgerBalance.balances(address(this));  }  } | |
| **Output:** |  |



**5C. To Create a Complete Solidity Smart Contract Code:**

|  |
| --- |
| pragma solidity ^0.6.0; contract SmartContractDemo { address payable public owner;    enum Statuses { Vacant, Occupied } Statuses currentStatus; event Occupy(address \_occupied, uint \_value);  constructor() public { owner = msg.sender; currentStatus = Statuses.Vacant; |
| } modifier onlyWhileVacant { require(currentStatus == Statuses.Vacant, "Currently Occupied");  \_; } modifier costs(uint \_amount) { require(msg.value >= \_amount, "Not enough Ether provided!");  \_; } receive() external payable onlyWhileVacant costs(2 ether) { currentStatus = Statuses.Occupied; owner.transfer(msg.value);  emit Occupy(msg.sender, msg.value);  }  } |

**Output:**

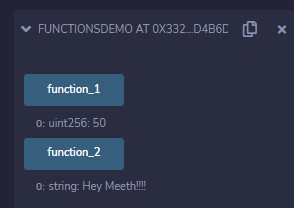


**Practical no : 6**

**Aim : Implement and demonstrate the use of solidity programming.**

**6A. Functions Code:**

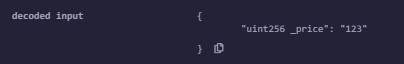
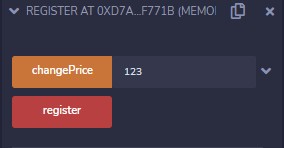
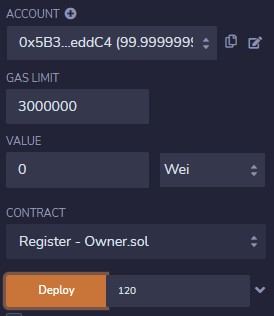
|  |  |
| --- | --- |
| pragma solidity ^0.6.0;  contract Functionsdemo  { function function\_1() public view returns(uint)  { uint a =28; uint b =22; uint result = a+b; return result;  } function function\_2() public view returns(string memory)  { string memory result = "Hey Meeth!!!!"; return result;  }  } | |
| **Output:** |  |



**6B. Function Modifiers Code:**

|  |
| --- |
| pragma solidity ^0.6.0;    contract Owner { address owner; constructor() public { owner = msg.sender;  } modifier onlyOwner { require(msg.sender == owner); |
| \_; } modifier costs(uint price) { if (msg.value >= price) {  \_;  }  } } contract Register is Owner { mapping (address => bool) registeredAddresses; uint price; constructor(uint initialPrice) public { price = initialPrice; }  function register() public payable costs(price) { registeredAddresses[msg.sender] = true;  } function changePrice(uint \_price) public onlyOwner { price = \_price;  }  } |

**Output:**



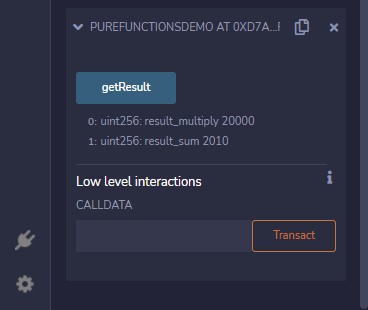
**6C. View functions and Pure functions Code:**

pragma solidity ^0.6.12;

contract PureFunctionsdemo{ function getResult() public pure returns(uint result\_multiply, uint result\_sum){ uint a = 2000; uint b =10; result\_multiply = a\*b; result\_sum = a+b;

} **}**

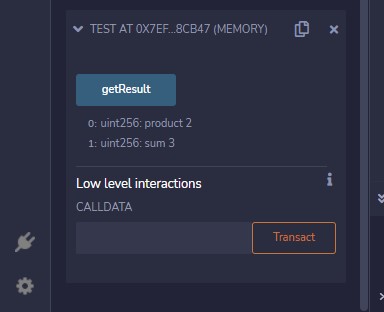
**Output:**



**View Function Code:**

|  |
| --- |
| pragma solidity ^0.6.12;  contract Test {  function getResult() public view returns(uint product, uint sum){ uint a = 1; uint b = 2; product = a \* b; sum = a + b;  }  } |

**Output:**

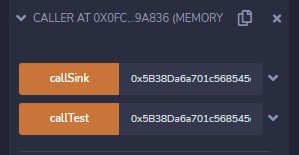


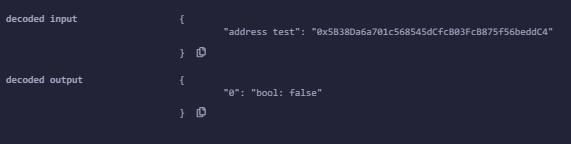
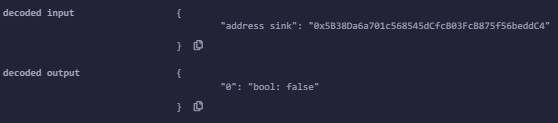
## Practical no : 7

**Aim : Implement and demonstrate the use of solidity programming.**

**7A. Fallback Functions Code:**

|  |  |
| --- | --- |
| pragma solidity ^0.6;  contract Test { uint public x ; fallback() external { }  } contract Sink { fallback() external payable { }  } contract Caller { function callTest(Test test) public returns (bool) {  (bool success,) = address(test).call(abi.encodeWithSignature("nonExistingFunction()")); require(success);  // test.x is now 1  address payable testPayable = address(uint160(address(test)));  // Sending ether to Test contract,  // the transfer will fail, i.e. this returns false here. return (testPayable.send(2 ether));  } function callSink(Sink sink) public returns (bool) { address payable sinkPayable = address(sink); return (sinkPayable.send(2 ether));  }  } | |
| **Output:** |  |

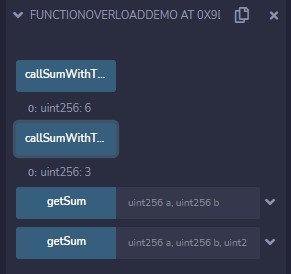




**7B. Function Overloading Code:**

|  |
| --- |
| pragma solidity ^0.6.12;  contract FunctionOverloaddemo{ function getSum(uint a, uint b) public pure returns(uint){ return a+b;  } function getSum(uint a, uint b, uint c) public pure returns(uint){ return a+b+c;  } function callSumWithTwoArguments() public pure returns(uint){ return getSum(1,2);  } function callSumWithThreeArguments() public pure returns(uint){ return getSum(1,2,3);  }  } |

**Output:**



**7C. Mathematical and Cryptographic Functions**

**Mathematical Functions Code:**

pragma solidity ^0.6.0;

contract MathFunctiondemo { function callAddMod() public pure returns(uint){ return addmod(4, 5, 3);

} function callMulMod() public pure returns(uint){ return mulmod(4, 5, 3);

} }

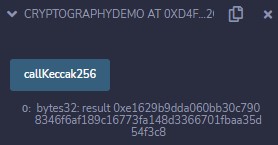
**Output:**



**Code:**

|  |
| --- |
| pragma solidity ^0.6.0;  contract Cryptographydemo {  function callKeccak256() public pure returns(bytes32 result){ return keccak256("ABC");  }  } |

**Output:**

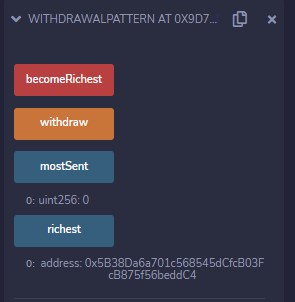


## Practical no : 8

**Aim : Implement and demonstrate the use of solidity programming**

**8A. Withdrawal Patterns Code:**

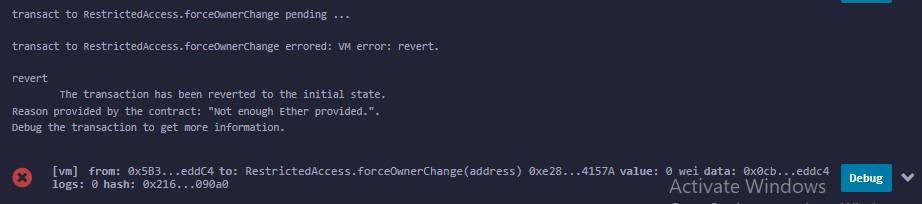
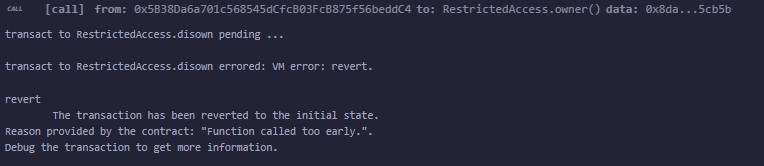
|  |  |
| --- | --- |
| pragma solidity ^0.6.0;  contract WithdrawalPattern { address public richest; uint public mostSent;  mapping (address => uint) pendingWithdrawals;  constructor() public payable { richest = msg.sender; mostSent = msg.value;  } function becomeRichest() public payable returns (bool) { if (msg.value > mostSent) { pendingWithdrawals[richest] += msg.value; richest = msg.sender; mostSent = msg.value; return true; } else {  return false;  } } function withdraw() public { uint amount = pendingWithdrawals[msg.sender]; pendingWithdrawals[msg.sender] = 0; msg.sender.transfer(amount);  }  } | |
| **Output** |  |



**8B. Restricted Access Code:**

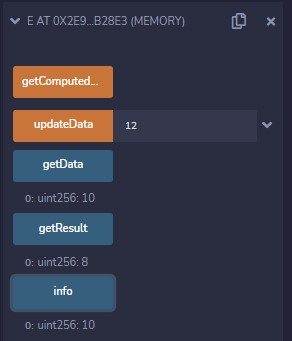
|  |
| --- |
| pragma solidity ^0.6.0;  contract RestrictedAccess{ address public owner = msg.sender; uint public creationTime = now;  modifier onlyBy(address \_account) { require( msg.sender == \_account,  "Sender not authorized."  );  \_; } function changeOwner(address \_newOwner) public onlyBy(owner) { owner = \_newOwner;  } modifier onlyAfter(uint \_time) { require(  now >= \_time,  "Function called too early."  );  \_; } function disown() public onlyBy(owner) onlyAfter(creationTime + 6 weeks) { delete owner;  } modifier costs(uint \_amount) { require( msg.value >= \_amount,  "Not enough Ether provided."  ); \_; if (msg.value > \_amount)  msg.sender.transfer(msg.value - \_amount);  } function forceOwnerChange(address \_newOwner) public payable costs(200 ether) {  owner = \_newOwner; if (uint(owner) & 0 == 1) return;  }  } |

**Output:**



**8C Contracts Code:**

|  |  |
| --- | --- |
| pragma solidity ^0.5.0;  contract C {  //private state variable uint private data;    //public state variable uint public info;    //constructor constructor() public { info = 10;  }  //private function  function increment(uint a) private pure returns(uint) { return a + 1; }    //public function function updateData(uint a) public { data = a; } function getData() public view returns(uint) { return data; } function compute(uint a, uint b) internal pure returns (uint) { return a + b; }  }  //External Contract contract D {  function readData() public returns(uint) {  C c = new C();  c.updateData(7); return c.getData(); | |
| }  }  //Derived Contract contract E is C { uint private result;  C private c;  constructor() public { c = new C();  } function getComputedResult() public { result = compute(3, 5);  } function getResult() public view returns(uint) { return result; } function getData() public view returns(uint) { return c.info(); } } | |
| **Output:** |  |

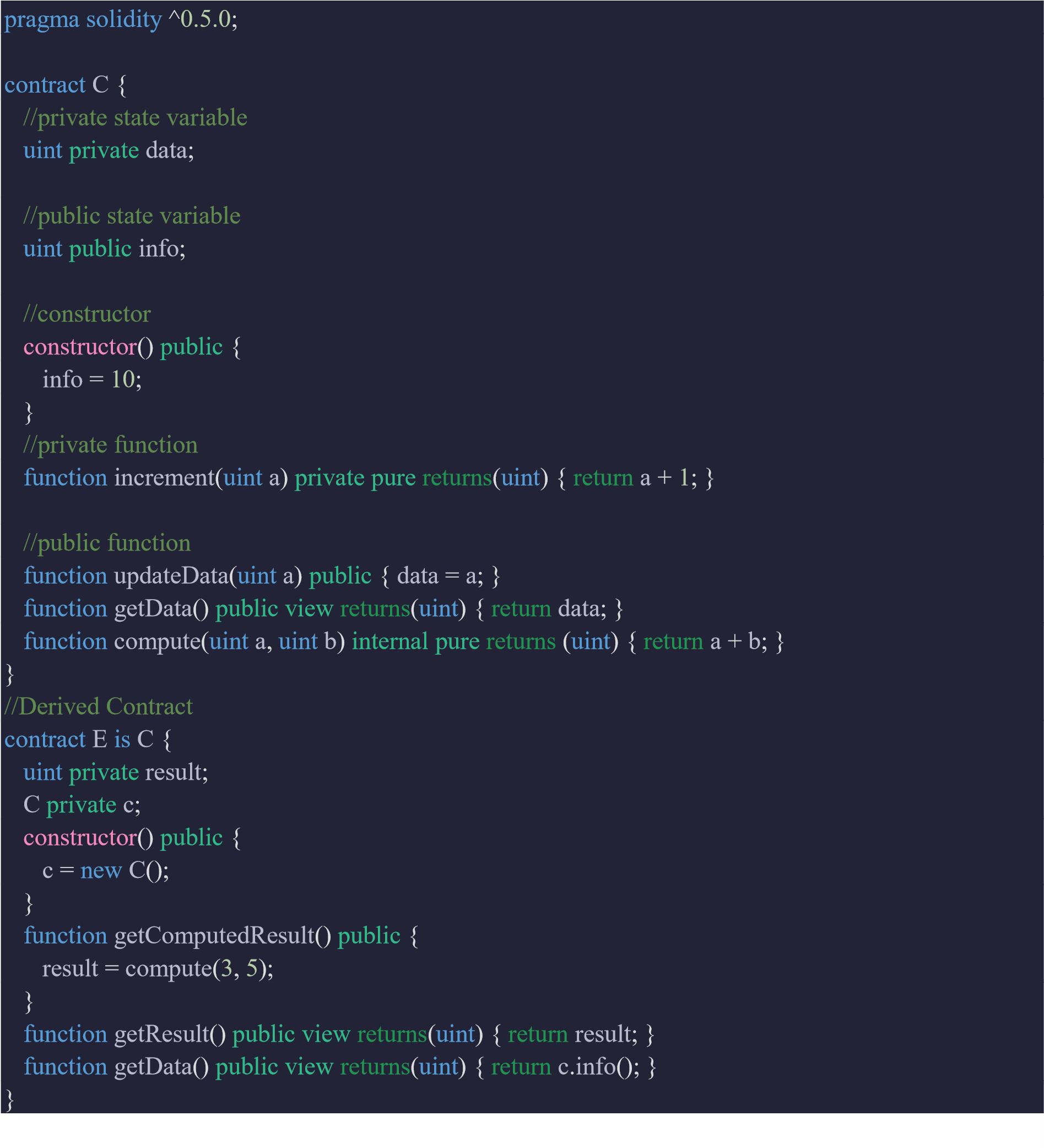


## Practical no : 9

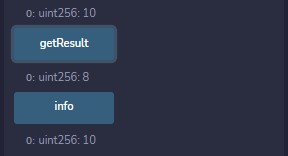
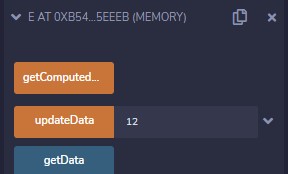
**Aim : Implement and demonstrate the use of solidity programming**

**9A. Inheritance**

**Code:**



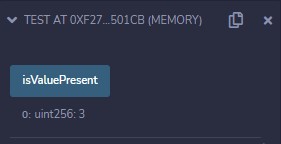
**Output:**



**9B. Libraries Code:**

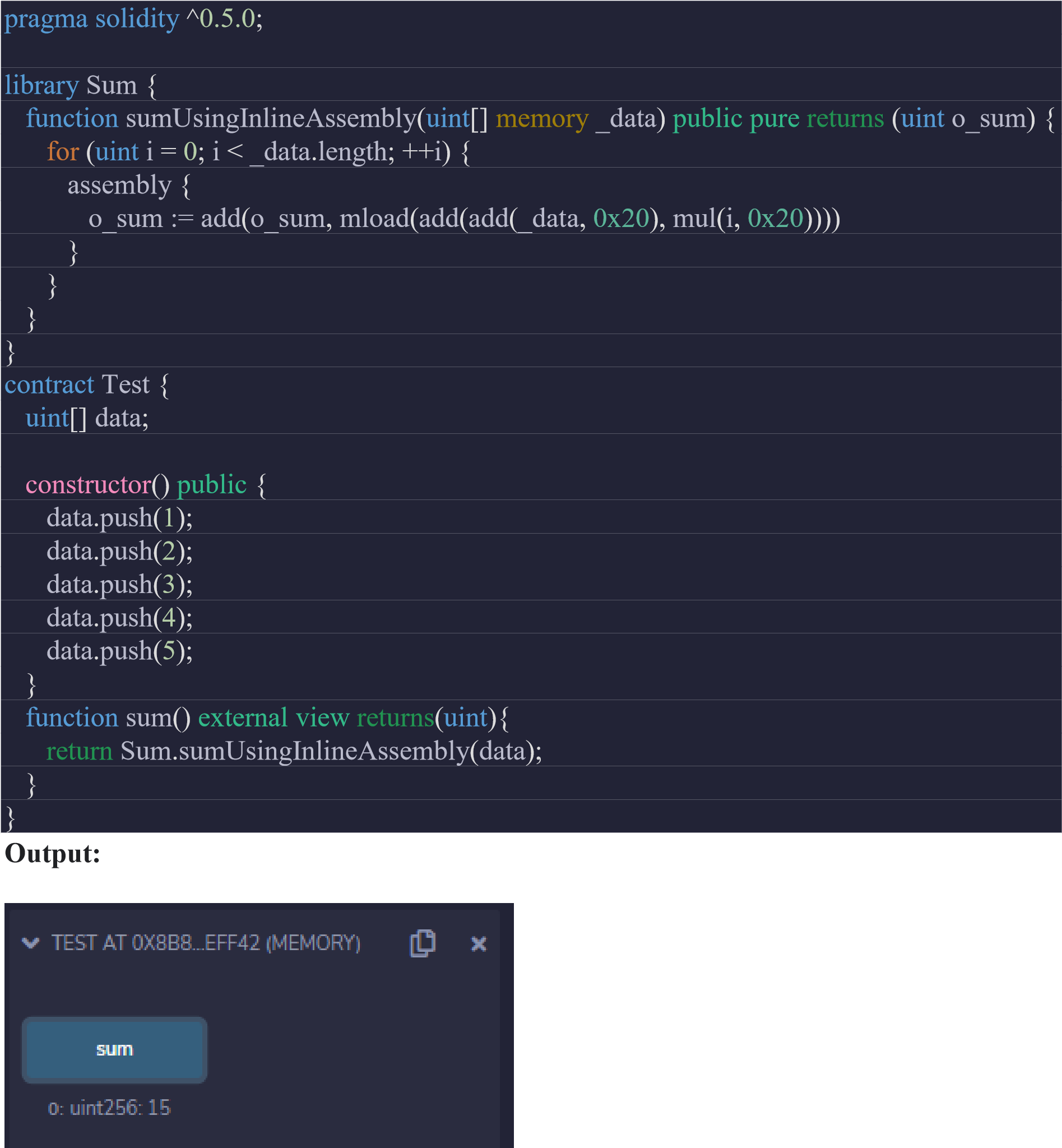
|  |
| --- |
| pragma solidity ^0.5.0;    library Search {  function indexOf(uint[] storage self, uint value) public view returns (uint) { for (uint i = 0; i < self.length; i++)if (self[i] == value) return i; return uint(-1);  } }  contract Test {  using Search for uint[]; uint[] data;  constructor() public { data.push(1); data.push(2); data.push(3); data.push(4); data.push(5);  }  function isValuePresent() external view returns(uint){ uint value = 4;    //Now data is representing the Library uint index = data.indexOf(value); return index;  }  } |

**Output:**



**9C. Assembly**

**Code:**



**9D. Error Handling**

**Code:**

pragma solidity ^0.5.0;

contract Vendor { address public seller; modifier onlySeller() {

require

(

msg

.

sender

==

seller

,

"Only seller can call this."

)

;

\_

;

}

function

sell

(

uint

amount

)

public

payable

onlySeller

{

if

(

amount

>

msg

.

value

/

2

ether

)

revert

(

"Not enough Ether provided."

)

;

// Perform the sell operation.

}

}

**Output:**

