BLOCKCHAIN

Certified Journal

Submitted in partial fulfilment of the

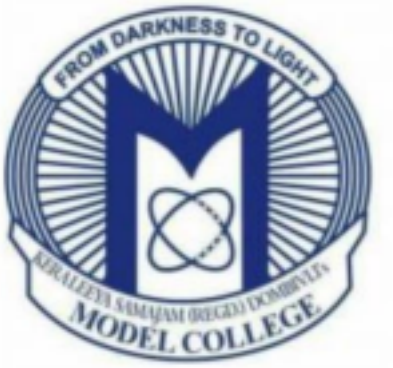
Requirements for the award of the Degree of

MASTER OF SCIENCE

(INFORMATION_TECHNOLOGY)

By

Anjali Rameshwar Nimje



DEPARTMENT OF INFORMATION TECHNOLOGY

KERALEEYA SAMAJAM (REGD.) DOMBIVLI'S

MODEL COLLEGE (AUTONOMOUS)

Re-Accredited 'A' Grade by NAAC

(Affiliated to University of Mumbai)

FOR THE YEAR

(2023-24)

Keraleeya Samajam(Regd.) Dombivli's

MODEL COLLEGE

Re-Accredited Grade "A" by NAAC

Kanchan Goan Village, Khambalpada, Thakurli East – 421201
Contact No – 7045682157, 7045682158. www.model-college.edu.in

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr. /Miss _____

*Studying in Class*_____ *Seat No.* _____

*Has completed the prescribed practicals in the subject*_____

*During the academic year*_____

Date : _____

External Examiner

Internal Examiner
M.Sc. Information Technology

INDEX

SR.NO	TOPIC	DATE	SIGNATURE
1	**Practical 1 : Write the following programs for Blockchain in Python :** (I). A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it (II). A transaction class to send and receive money and test it .		
2	**Practical 2 : Write the following programs for Blockchain in Python :** (I).Create multiple transactions and display them . (II). Create a blockchain, a genesis block and execute it .		
3	**Practical 3 : Write the following programs for Blockchain in Python :** (I).Create a mining function and test it . (II).Add blocks to the miner and dump the blockchain .		
4	**Practical 4 : Implement and demonstrate the use of the following in Solidity :** (I).Variable (II)Operators (III).Loops (IV).Decision Making (V).Strings		
5	**Practical 5 : Implement and demonstrate the use of the following in Solidity :** (I).Arrays (II).Enums (III).Structs (IV).Mappings (V).Conversations (VI).Ether Units (VII).Special Variables		
6	**Practical 6: Implement and demonstrate the use of the following in Solidity :**		

	(I).Functions (II).View Functions (III).Pure Functions (IV).Function Overloading (V).Mathematical Functions (VI).Cryptographic Functions		
7	**Practical 7 : Implement and demonstrate the use of the following in Solidity :** (I).Contracts (II).Inheritance (III).Constructors (IV).Abstract Contracts V).Interfaces		
8	**Practical 8 : Implement and demonstrate the use of the following in Solidity :** (I).Error Handling		

PRACTICAL NO : 1

(I) A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it

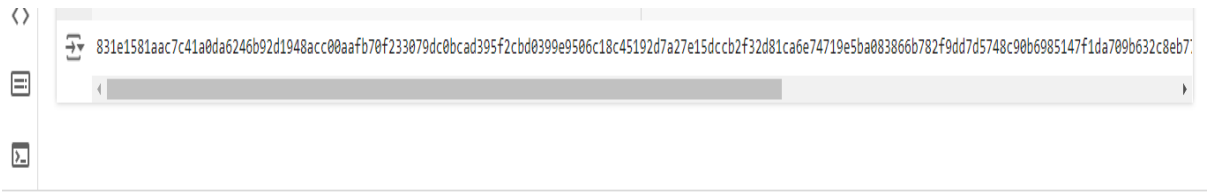
(II) A transaction class to send and receive money and test it .

CODE:

```
import hashlib
import random
import string
import json
import binascii
import numpy as np
import pandas as pd
import pylab as pl
import logging
import datetime
import collections
pip install pycryptodome
# following imports are required by PKI
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
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')
  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': identity,
      'recipient': self.recipient,
      'value': self.value,
      'time' : self.time})
    def sign_transaction(self):
     private_key = self.sender._private_key
   signer = PKCS1_v1_5.new(private_key)
     h = SHA.new(str(self.to_dict()).encode('utf8'))
     return binascii.hexlify(signer.sign(h)).decode('ascii')
Dinesh = Client()
    Ramesh = Client()
t = Transaction(Dinesh,Ramesh.identity,5.0)
signature = t.sign_transaction()
print (signature)
```

OUTPUT



PRACTICAL NO : 2

(I).Create multiple transactions and display them .

(II). Create a blockchain, a genesis block and execute it .

CODE:

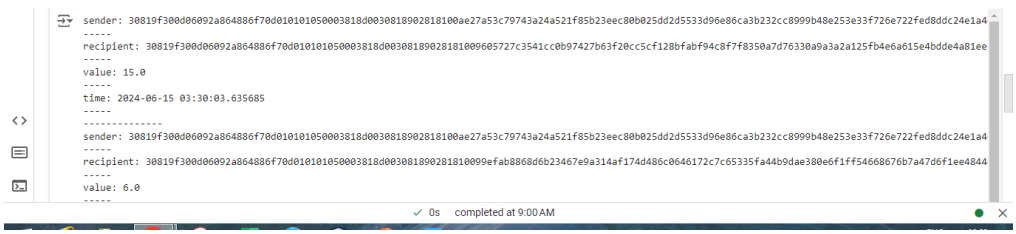
```
def display_transaction(transaction):
  #for transaction in transactions:
  dict = transaction.to_dict()
  print ("sender: " + dict['sender'])
  print ('-----')
  print ("recipient: " + dict['recipient'])
  print ('-----')
  print ("value: " + str(dict['value']))
  print ('-----')
  print ("time: " + str(dict['time']))
  print ('-----')
transactions = []
Dinesh = Client()
Ramesh = Client()
Seema = Client()
Vijay = Client()
t1 = Transaction(
  Dinesh,
  Ramesh.identity,
  15.0
)
t1.sign_transaction()
transactions.append(t1)
t2 = Transaction(
  Dinesh,
  Seema.identity,
```

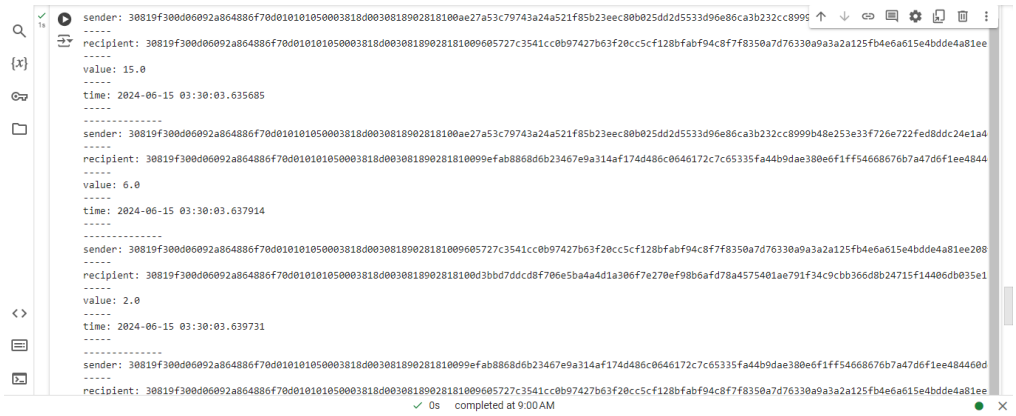
```
  6.0
)
t2.sign_transaction()
transactions.append(t2)
t3 = Transaction(
  Ramesh,
  Vijay.identity,
  2.0
)
t3.sign_transaction()
transactions.append(t3)
t4 = Transaction(
  Seema,
  Ramesh.identity,
  4.0
)
t4.sign_transaction()
transactions.append(t4)
t5 = Transaction(
  Vijay,
  Seema.identity,
 7.0
)
t5.sign_transaction()
transactions.append(t5)
t6 = Transaction(
  Ramesh,
  Seema.identity,
  3.0
)
t6.sign_transaction()
transactions.append(t6)
t7 = Transaction(
  Seema,
```

```
  Dinesh.identity,
  8.0
)
t7.sign_transaction()
transactions.append(t7)
t8 = Transaction(
  Seema,
  Ramesh.identity,
  1.0
)
t8.sign_transaction()
transactions.append(t8)
t9 = Transaction(
  Vijay,
  Dinesh.identity,
  5.0
)
t9.sign_transaction()
transactions.append(t9)
t10 = Transaction(
  Vijay,
  Ramesh.identity,
  3.0
)
t10.sign_transaction()
transactions.append(t10)
for transaction in transactions:
  display_transaction (transaction)
  print ('----------')
class Block:
  def __init__(self):
    self.verified_transactions = []
    self.previous_block_hash = ""
    self.Nonce = ""
```

```
last_block_hash = ""
Dinesh = Client()
t0 = Transaction (
  "Genesis",
  Dinesh.identity,
  500.0
)
block0 = Block()
block0.previous_block_hash = None
Nonce = None
block0.verified_transactions.append (t0)
digest = hash (block0)
last_block_hash = digest
TPCoins = []
def dump_blockchain (self):
  print ("Number of blocks in the chain: " + str(len (self)))
  for x in range (len(TPCoins)):
    block_temp = TPCoins[x]
    print ("block # " + str(x))
    for transaction in block_temp.verified_transactions:
      display_transaction (transaction)
      print ('--------------')
    print ('======================================')
TPCoins.append (block0)
dump_blockchain(TPCoins)
```

OUTPUT:




sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee
value: 15.0
time: 2024-06-15 03:30:03.635685
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
value: 6.0
time: 2024-06-15 03:30:03.637914
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460d
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee
0s completed at 9:00 AM


recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208
value: 4.0
time: 2024-06-15 03:30:03.641381
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
value: 7.0
time: 2024-06-15 03:30:03.642927
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
value: 3.0
time: 2024-06-15 03:30:03.644313
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460d
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee
value: 4.0
time: 2024-06-15 03:30:03.641381
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
value: 7.0
time: 2024-06-15 03:30:03.642927
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
value: 3.0
time: 2024-06-15 03:30:03.644313
sender: 30819f300d06092a864886f70d0101010500038 18d00308189028181009efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460d
recipient: 30819f300d06092a864886f70d0101010500038 18d00308189028181009ae27a53c79743a24a521f85b23eec80b025dd2d5533d96e86ca3b232cc8999b48e253e33f726e722fed8ddc24e
0s completed at 9:00 AM

PRACTICAL NO: 3

(I).Create a mining function and test it .

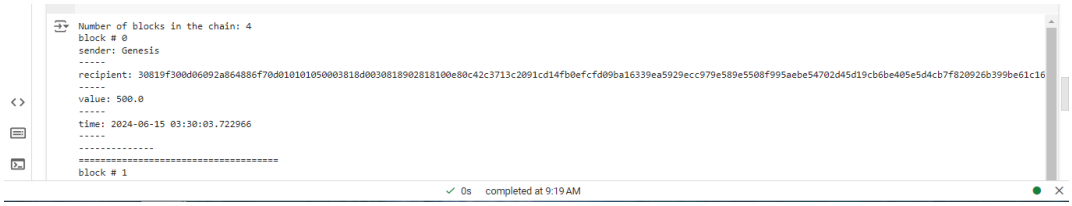
(II).Add blocks to the miner and dump the blockchain .

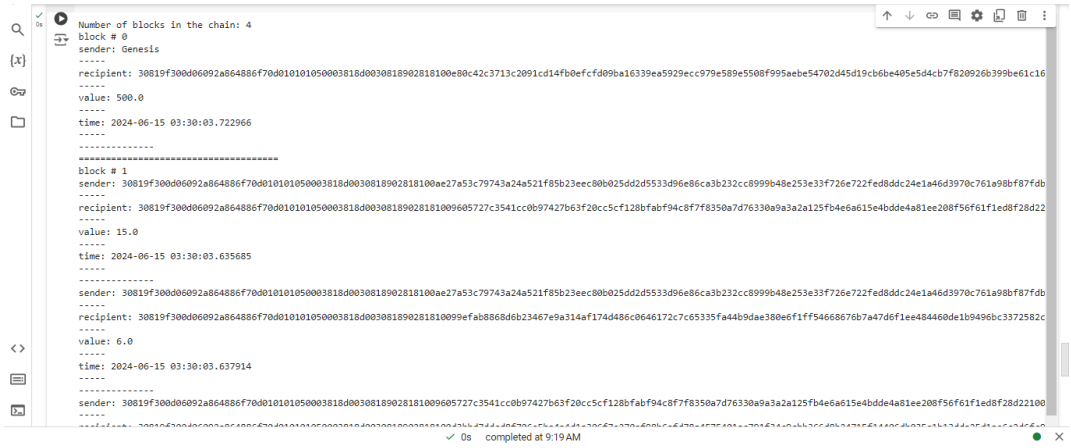
CODE:

```
def sha256(message):
  return hashlib.sha256(message.encode('ascii')).hexdigest()
def mine(message, difficulty=1):
   assert difficulty >= 1
   prefix = '1' * difficulty
   for i in range(1000):
      digest = sha256(str(hash(message)) + str(i))
      if digest.startswith(prefix):
         print ("after " + str(i) + " iterations found nonce: "+ digest)
      return digest
last_transaction_index = 0
block = Block()
for i in range(3):
   temp_transaction = transactions[last_transaction_index]
   # validate transaction
   # if valid
   block.verified_transactions.append (temp_transaction)
   last_transaction_index += 1 mine ("test message", 2)
block.previous_block_hash = last_block_hash
block.Nonce = mine (block, 2)
digest = hash (block)
TPCoins.append (block)
last_block_hash = digest
# Miner 2 adds a block
block = Block()
for i in range(3):
   temp_transaction = transactions[last_transaction_index]
   # validate transaction
```

```
    # if valid
    block.verified_transactions.append (temp_transaction)
    last_transaction_index += 1
  block.previous_block_hash = last_block_hash
  block.Nonce = mine(block, 2)
  digest = hash (block)
  TPCoins.append (block)
  last_block_hash = digest
  # Miner 3 adds a block
  block = Block()
  for i in range(3):
    temp_transaction = transactions[last_transaction_index]
    #display_transaction (temp_transaction)
    # validate transaction
    # if valid
   block.verified_transactions.append (temp_transaction)
    last_transaction_index += 1
  block.previous_block_hash = last_block_hash
  block.Nonce = mine (block, 2)
  digest = hash (block)
  TPCoins.append (block)
  last_block_hash = digest
  dump_blockchain(TPCoins)
```

OUTPUT:




Number of blocks in the chain: 4
block # 0
sender: Genesis
recipient: 30819f300d06092a864886f70d0101010500038...
value: 500.0
time: 2024-06-15 03:30:03.722966
block # 1
sender: 30819f300d06092a864886f70d0101010500038...
recipient: 30819f300d06092a864886f70d0101010500038...
value: 15.0
time: 2024-06-15 03:30:03.635685
sender: 30819f300d06092a864886f70d0101010500038...
recipient: 30819f300d06092a864886f70d0101010500038...
value: 6.0
time: 2024-06-15 03:30:03.637914
sender: 30819f300d06092a864886f70d0101010500038...
0s completed at 9:19 AM

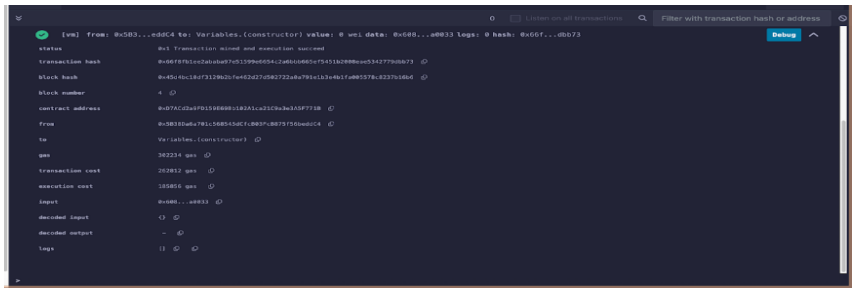

recipient: 30819f300d06092a864886f70d0101010500038...
value: 6.0
time: 2024-06-15 03:30:03.637914
sender: 30819f300d06092a864886f70d0101010500038...
recipient: 30819f300d06092a864886f70d0101010500038...
value: 2.0
time: 2024-06-15 03:30:03.639731
block # 2
sender: 30819f300d06092a864886f70d0101010500038...
recipient: 30819f300d06092a864886f70d0101010500038...
value: 4.0
time: 2024-06-15 03:30:03.641381
sender: 30819f300d06092a864886f70d0101010500038...
recipient: 30819f300d06092a864886f70d0101010500038...
value: 7.0
0s completed at 9:19 AM

PRACTICAL NO : 4

Implement and demonstrate the use of the following in Solidity :

(I).Variable

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.6.12 <0.9.0;
contract Variables
{
string public d1="AnjaliAnushkaRupa";
function foo() public pure returns(int){
int a=10;
return a;
}
uint public timestamp = block.timestamp;
}
```



// **Solidity program to demonstrate state variables**

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;
// Creating a contract
contract Solidity_var_Test {
   // Declaring a state variable
   uint8 public state_var;

   // Defining a constructor
   constructor() public
   {
       state_var = 16;
   }
}
```

OUTPUT:

// **Solidity program to show Global variables**

```
pragma solidity >=0.6.12 <0.9.0;
contract Test {
address public admin;
constructor() public {
admin = msg.sender;
}
}
```

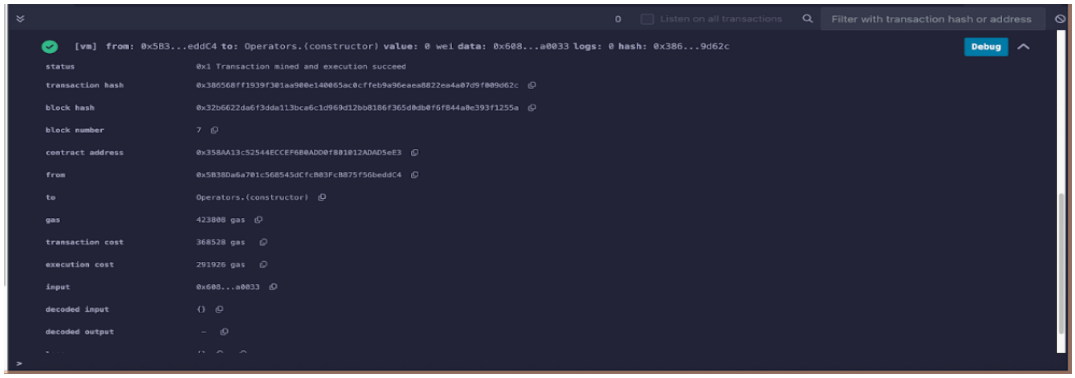
OUTPUT:

(II).Operators

// Solidity contract to demonstrate Arithemetic Operator

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;
contract Operators {
   uint16 public a = 20;
   uint16 public b = 10;
   uint public sum = a + b;
   uint public diff = a - b;
   uint public mul = a * b;
   uint public div = a / b;
   uint public mod = a % b;
   uint public dec = --b;
   uint public inc = ++a;
}
```

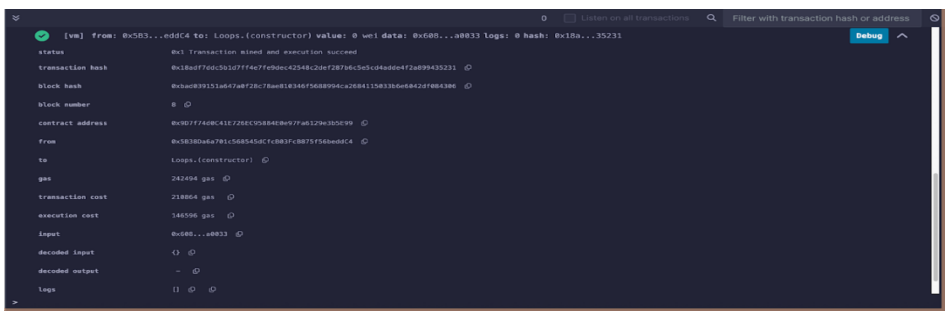
OUTPUT :



(III).Loops

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.6.12 <0.9.0;
contract Loops
{
int public j=0;
int public k=0;
int public l=0;
function forLoop() public returns (int)
{
for(int i=0; i<5;i++)
{
j++;
}
return j;
}
function whileLoop() public returns (int)
{
int i=0;
while(i<5)
{
k++;
i++;
}
return k;
}
function dowhileLoop() public returns (int)
{
int i=0;
do
{
l++;
i++;
}
while(i<5);
return 1;
}
}
```

OUTPUT:



(IV).Decision Making

// **Solidity program to demonstrate the use of 'if statement'**

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;
contract ifstatementts {
    // Declaring state variable
    uint i = 10;
    // Defining function to demonstrate use of 'if statement'
    function decision_making() public view returns(bool) {
        if (i < 10) {
            return true;
        } else {
            return false;
        }
    }
}
```

OUTPUT:

// **Solidity program to demonstrate the use of 'if...else' statement**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ifelsestat {
    uint public i = 10;
    bool public even;
    function decision_making() public {
        if (i % 2 == 0) {
            even = true;
        } else {
            even = false;
        }
    }
    function getResult() public view returns (bool) {
        return even;
    }
}
```

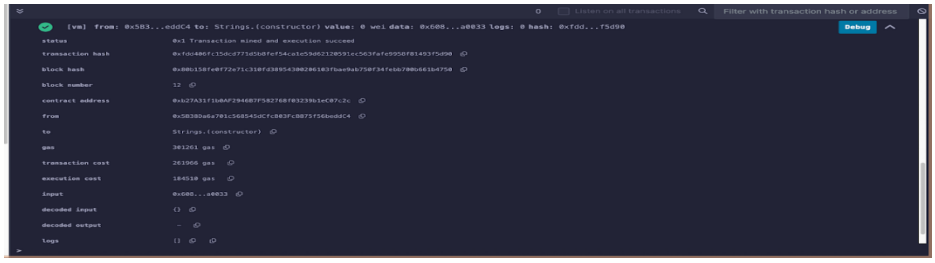
OUTPUT:

(V) Strings

// Solidity program to demonstrate strings (by default)

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;
contract Strings
{
string str1 = "ANJALI";
string str2 = "ANUSHKA";
function getstr1() public view returns(string memory)
{
return str1;
}
function getstr2() public view returns(string memory)
{
return str2;
}
}
```

OUTPUT:



// Solidity program to demonstrate strings (input from user)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract strings {
string public str;
constructor(string memory str_in) {
str = str_in;
}
    function str_out() public view returns (string memory) {
        return str;
    }
}
```

OUTPUT :

PRACTICAL NO : 5

Implement and demonstrate the use of the following in Solidity :

(I).Arrays

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Types {
    // Declaring state variables
    int[5] public data;
    uint[6] public data1;
    // Function to initialize the arrays
    function array_example() public {
        data = [int(50), -63, 77, -28, 90];
        data1 = [uint(10), 20, 30, 40, 50, 60];
    }
    // Function to return the values of the arrays
    function getresult() public view returns (int[5] memory,
uint[6] memory) {
        return (data, data1);
    }
}
```

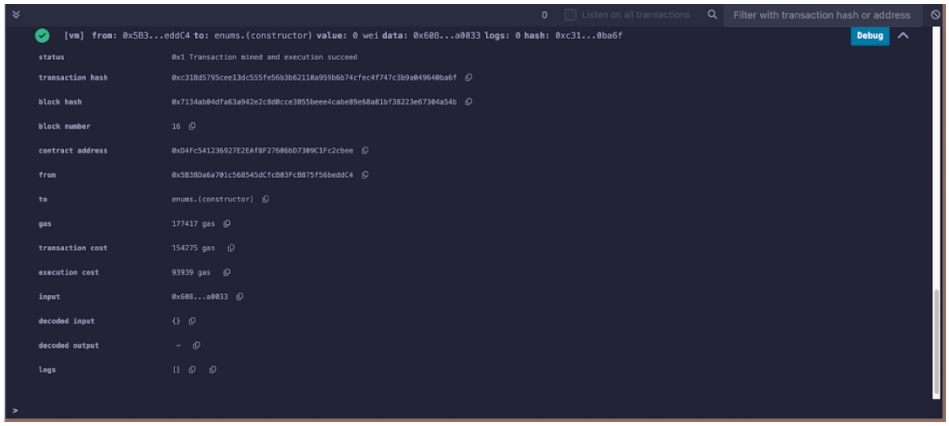
OUTPUT:

(II). Enums

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract test {
   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);
    }
}
```

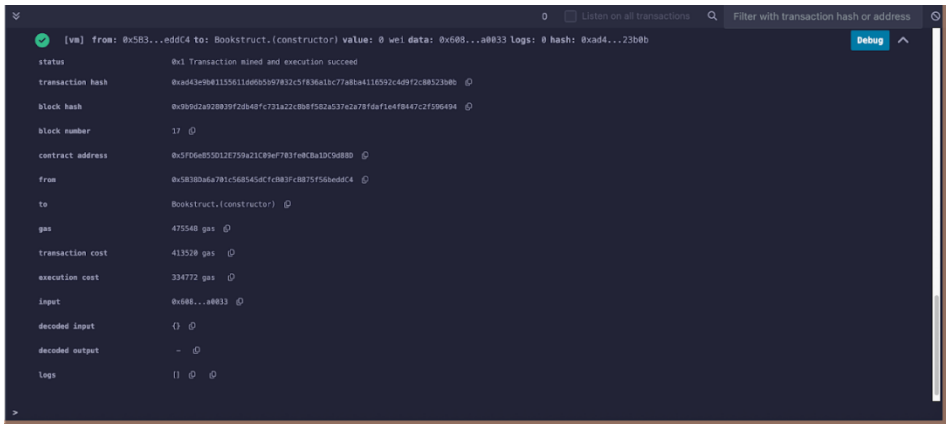
OUTPUT:



(III).Structs

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Bookstruct {
    struct Book {
        string title;
        string author;
        uint book_id;
    }
    Book public book;
    function setBook() public {
        book = Book('Learn Java', 'TP', 1);
    }
    function getBookId() public view returns (uint) {
        return book.book_id;
    }
}
```

OUTPUT:



(IV).Mappings

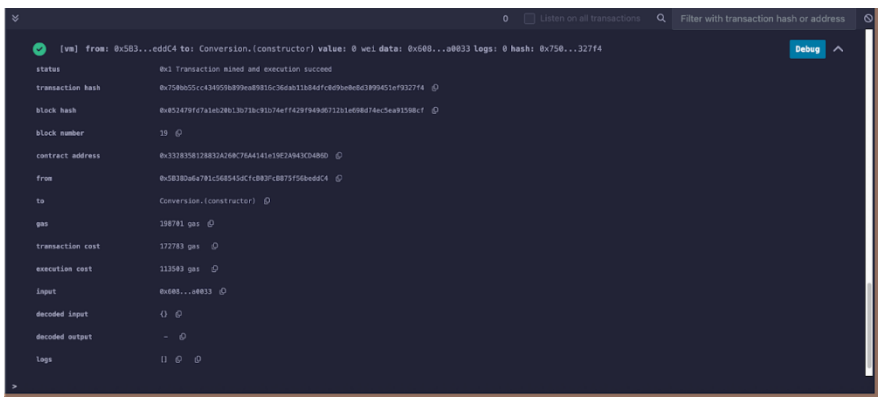
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract LedgerBalancemapping {
   mapping(address => uint) public balance;
   function updateBalance() public returns(uint) {
       balance[msg.sender] = 20;
       return balance[msg.sender];
   }
}
```

OUTPUT:

(V).Conversion

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Conversion
{
uint a=10;
uint8 b=10;
uint16 c=10;
function add() public view returns (uint)
{
uint d = a + uint(b) + uint(c);
return d;
}
}
```

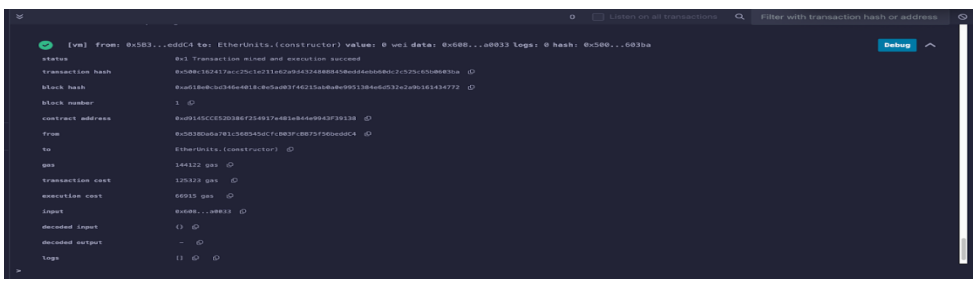
OUTPUT:



(VI).Ether Units

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract EtherUnits

{

function units() public pure returns  (bool,bool,bool)

{

bool a=false;

bool b=false;

bool c=false;

if ((1 ether) * 10**18 == 1000 * 10**15) // 1 Ether equals to
1000 Finney

a=true;

if ((1 ether) * 10**18 == 1000 * 10**12) // 1 Ether equals to
1000 Szabo

b=true;

if (1 ether == 10000000000000000000)   // 1 Ether equals to
10000000000000000000 Wei

c=true;

return (a,b,c);

}

}
```

OUTPUT:



(VII).Special Variables

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract SpecialVariables

{

uint data1=0;

uint data2=0;

uint data3=0;

function set() public

{

data1=block.difficulty;

data2=block.timestamp;

data3=block.number;

}

function get() public view returns (uint, uint, uint)

{

return(data1, data2, data3);

}

}
```

OUTPUT:

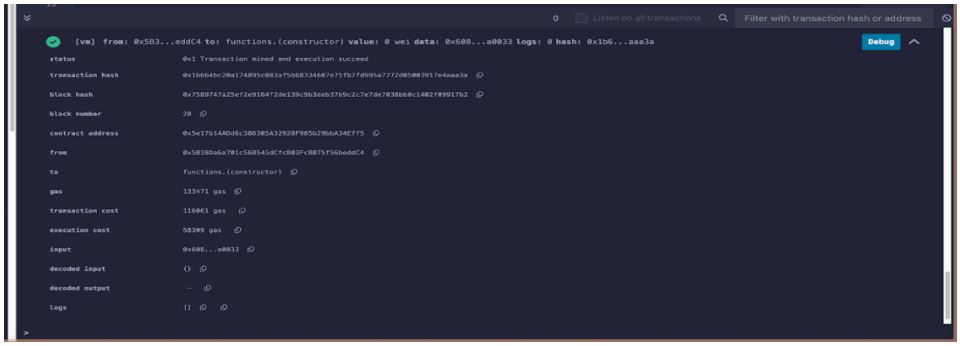
PRACTICAL NO : 6

Implement and demonstrate the use of the following in Solidity :

(I).Functions

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract functions {
    function testpgmresult() public pure returns (uint) {
        uint a = 1000; // local variable
        uint b = 2000;
        uint result = a + b;
        return result;
    }
}
```

OUTPUT:



(II).View Functions

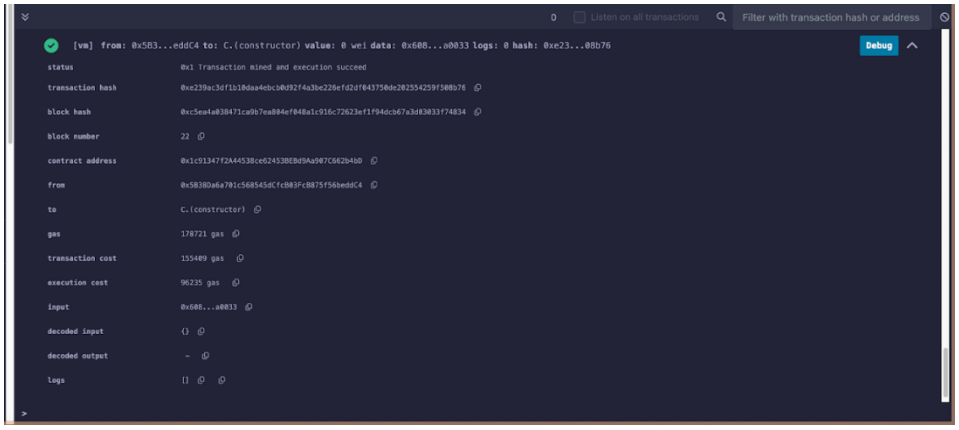
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract viewfunction {
    function getResult() public pure returns (uint product,
uint sum) {
        uint a = 1; // local variable
        uint b = 2;
        product = a * b;
        sum = a + b;
    }
}
```

OUTPUT:

(III).Pure Functions

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
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;
    }
}
// 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 getDataFromC() public view returns (uint) {
        return c.info();
    }
}
```

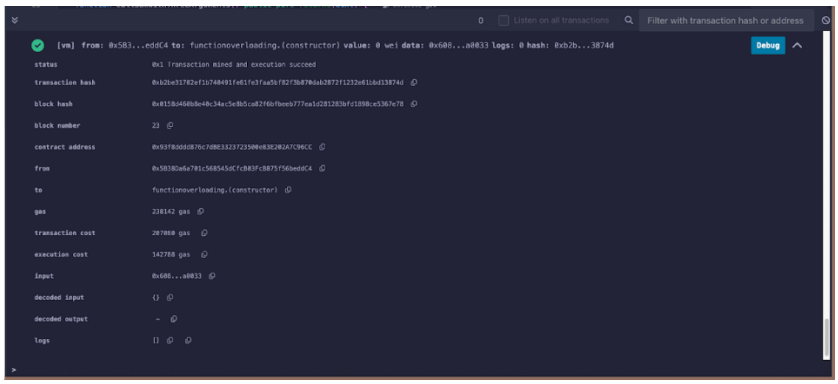
OUTPUT:



(IV).Function Overloading

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract functionoverloading {
    function getSum(uint a, uint b) public pure returns(uint)
{
        return a + b;
    }
    function getSumWithThreeArguments(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 getSumWithThreeArguments(1, 2, 3);
    }
}
```

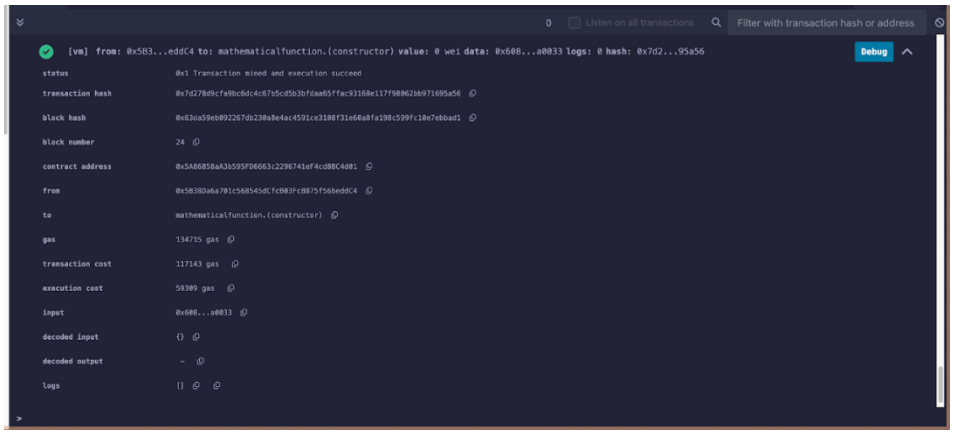
OUTPUT:



(V).Mathematical Functions

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract mathematicalfunction {
    function callAddMod() public pure returns(uint) {
        return addmod(4, 5, 3);  // Computes (4 + 5) % 3 = 6 %
3 = 0
    }
    function callMulMod() public pure returns(uint) {
        return mulmod(4, 5, 3);  // Computes (4 * 5) % 3 = 20
% 3 = 2
    }
}
```

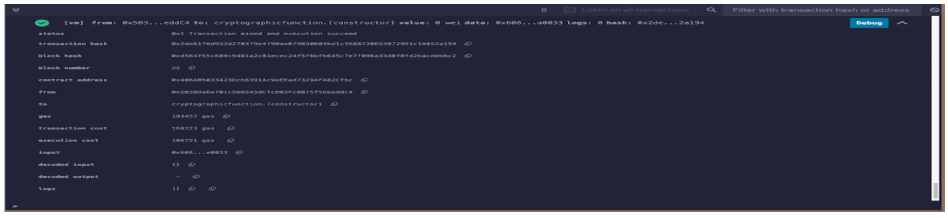
OUTPUT:



(VI).Cryptographic Functions

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract cryptographicfunction {
    function callSha256() public pure returns(bytes32 result)
{
        return sha256("ronaldo");
    }
    function callKeccak256() public pure returns(bytes32
result) {
        return keccak256("ronaldo");
    }
}
```

OUTPUT:



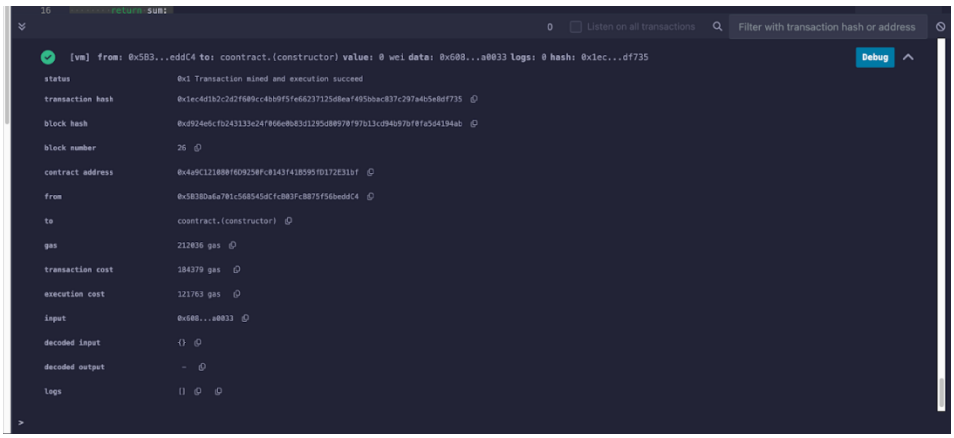
PRACTICAL NO:7

Implement and demonstrate the use of the following in Solidity :

(I).Contracts

```
pragma solidity >=0.6.12 <0.9.0;
// Defining a contract
contract coontract {
   // Declaring state variables
   uint public var1;
   uint public var2;
   uint public sum;
   // Defining public function that sets the value of the
state variables
   function set(uint x, uint y) public {
       var1 = x;
       var2 = y;
       sum = var1 + var2;
   }
   // Defining function to print the sum of state variables
   function get() public view returns (uint) {
       return sum;
   }
}
```

OUTPUT:

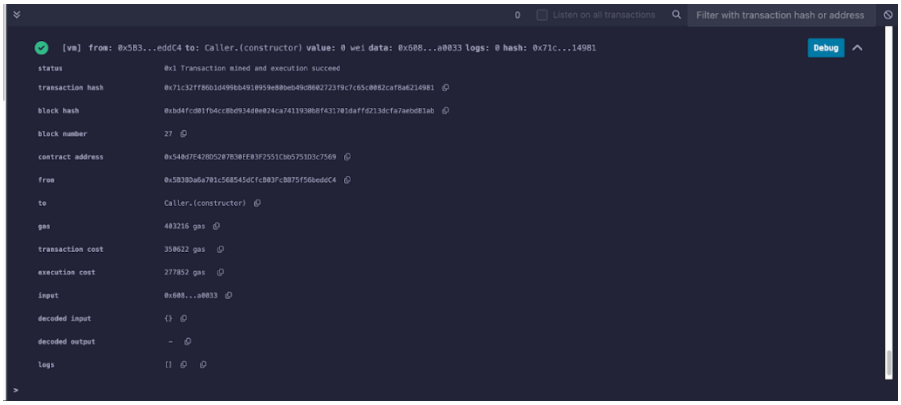


(II).Inheritance

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
// Defining parent contract
contract Parent {
    // Declaring internal state variable
    uint internal sum;
    // Defining external function to set value of internal
state variable sum
    function setValue() external {
        uint a = 10;
        uint b = 20;
```

```
        sum = a + b;
    }
}
// Defining child contract inheriting from parent
contract Child is Parent {
    // Defining external function to return value of internal
state variable sum
    function getValue() external view returns(uint) {
        return sum;
    }
}
// Defining caller contract
contract Caller {
    // Creating child contract object
    Child public cc = new Child();
    // Defining function to call setValue and getValue
functions
    function testInheritance() public {
        cc.setValue();
    }
    function getResult() public view returns(uint) {
        return cc.getValue();
    }
}
```

OUTPUT:

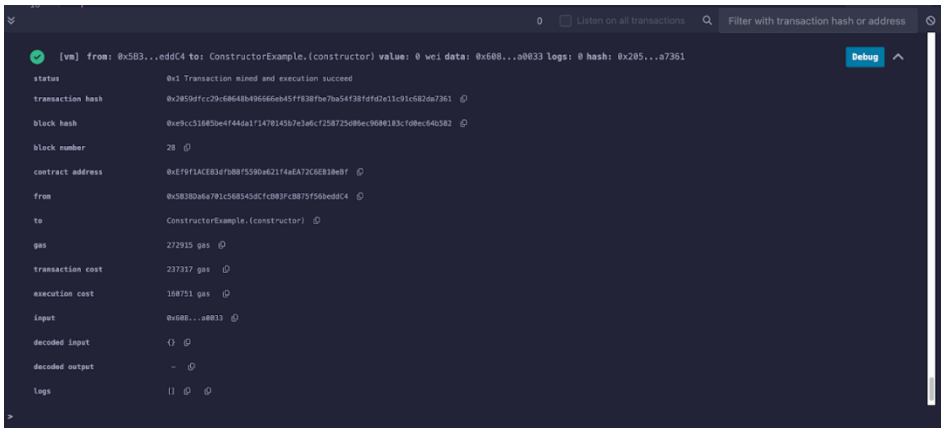


(III).Constructors

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

// Creating a contract
contract ConstructorExample {

    // Declaring state variable
    string public str;

    // Creating a constructor to set value of 'str'
```

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

    // Defining function to return the value of 'str'
    function getValue() public view returns (string memory) {
        return str;
    }
}
```

OUTPUT:

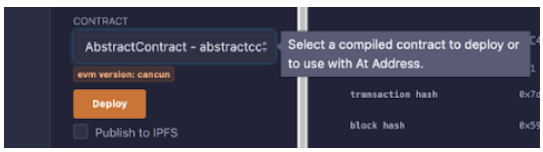


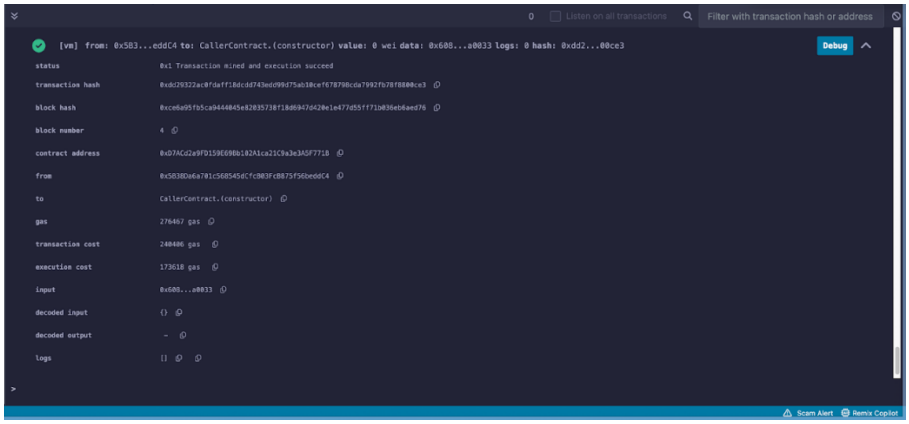
(IV).Abstract Contract

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

abstract contract AbstractContract

{

function getResult() public virtual pure returns(uint);

}

contract DerivedContract is AbstractContract

{

function getResult() public override pure returns(uint)
```

```
{
uint a = 1;
uint b = 2;
uint result = a + b;
return result;
}
}
contract CallerContract
{
AbstractContract abs;

function TestAbstract() public
{
abs = new DerivedContract();
}
function getValues() public view returns (uint)
{
return abs.getResult();
}
}
```

OUTPUT:



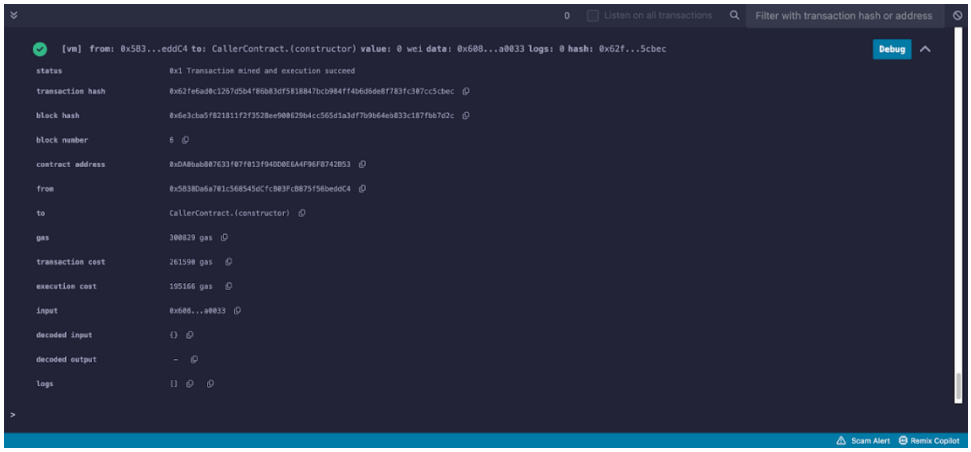


(V).Interfaces

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
interface Calculator
{
function getResult() external pure returns(uint);
}
contract TestContract is Calculator
{
function getResult() external pure returns(uint)
{
uint a = 1;
uint b = 2;
uint result = a + b;
return result;
}
}
```

```
contract CallerContract

{

TestContract tc = new TestContract();

function getValues() public view returns(uint)

{

return tc.getResult();

}

}
```

OUTPUT:



PRACTICAL NO:8

Implement and demonstrate the use of the following in Solidity :

(I).Error Handling

a.Require statement

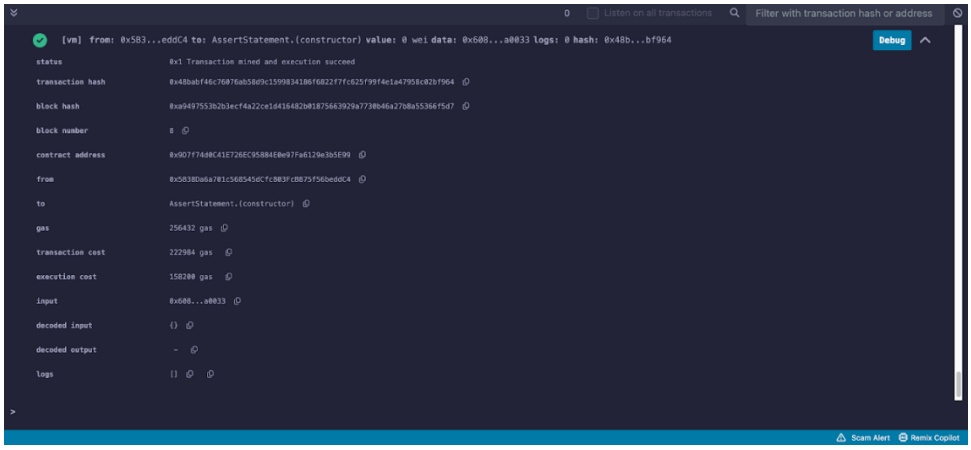
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract requireStatement {

    // Defining function to check input

    function checkInput(uint _input) public view returns(string
memory){

        require(_input >= 0, "invalid uint8");

        require(_input <= 255, "invalid uint8");

        return "Input is Uint8";

    }

    // Defining function to use require statement

    function Odd(uint _input) public view returns(bool){

        require(_input % 2 != 0);

        return true;

    }

}
```

OUTPUT:

b.Assert

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
// Creating a contract
contract AssertStatement {
    // Defining a state variable
    bool result;
    // Defining a function to check condition
    function checkOverflow(uint _num1, uint _num2) public {
        uint8 sum = uint8(_num1) + uint8(_num2);
        assert(sum <= 255);
        result = true;
    }
    // Defining a function to print result of assert statement
    function getResult() public view returns (string memory) {
        if (result) {
            return "No Overflow";
        } else {
            return "Overflow exists";
        }
    }
}
```

OUTPUT:

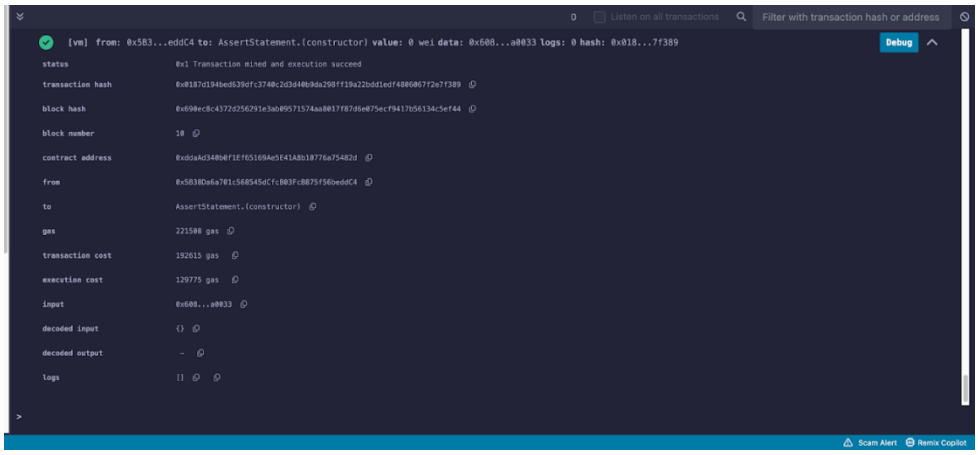


c.Assert

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract AssertStatement {

    // Defining a state variable

    bool result;

    // Defining a function to check condition

    function checkOverflow(uint8 sum) public {

        assert(sum <= 255);

        result = true;

    }

    // Defining a function to print result of assert statement

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

        if (result) {

            return "No Overflow";

        } else {

            return "Overflow exists";

        }
```

```
    }
}
```

OUTPUT:

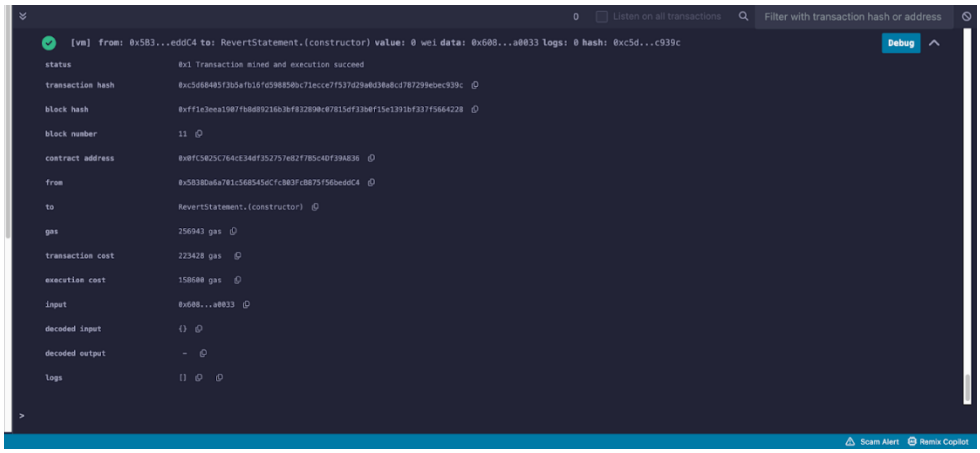


d.Revert

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

// Creating a contract

contract RevertStatement {

   // Defining a function to check condition

   function checkOverflow(uint _num1, uint _num2) public pure
returns (string memory, uint) {

        uint sum = _num1 + _num2;

        if (sum > 255) {

            revert("Overflow exists");

        } else {

            return ("No Overflow", sum);

        }
```

```
    }
}
```

OUTPUT:


0
Listen on all transactions
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: RevertStatement.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0xc5d...c939c
Debug
status
0x1 Transaction mined and execution succeed
transaction hash
0xc5d68405f3b5afb16fd598850c71ecce7f537d29a0d30a8cd787299e0ec939c
block hash
0xff1e3eea1907fb8d89216b3bf832890c07815df33b0f15e1391bf337f5664228
block number
11
contract address
0x0fC5025C764cE34df352757e82f7B5c4Df39A836
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
RevertStatement.(constructor)
gas
256943 gas
transaction cost
223428 gas
execution cost
150600 gas
input
0x608...a0033
decoded input
{}
decoded output
-
logs
[]
Scam Alert
Remix Copilot