

**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](http://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
<b>1</b>	<p><b>Practical 1 : Write the following programs for Blockchain in Python :</b></p> <p>(I). A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it</p> <p>(II). A transaction class to send and receive money and test it .</p>		
<b>2</b>	<p><b>Practical 2 : Write the following programs for Blockchain in Python :</b></p> <p>(I).Create multiple transactions and display them .</p> <p>(II). Create a blockchain, a genesis block and execute it .</p>		
<b>3</b>	<p><b>Practical 3 : Write the following programs for Blockchain in Python :</b></p> <p>(I).Create a mining function and test it .</p> <p>(II).Add blocks to the miner and dump the blockchain .</p>		
<b>4</b>	<p><b>Practical 4 : Implement and demonstrate the use of the following in Solidity :</b></p> <p>(I).Variable</p> <p>(II)Operators</p> <p>(III).Loops</p> <p>(IV).Decision Making</p> <p>(V).Strings</p>		
<b>5</b>	<p><b>Practical 5 : Implement and demonstrate the use of the following in Solidity :</b></p> <p>(I).Arrays</p> <p>(II).Enums</p> <p>(III).Structs</p> <p>(IV).Mappings</p> <p>(V).Conversations</p> <p>(VI).Ether Units</p> <p>(VII).Special Variables</p>		
<b>6</b>	<p><b>Practical 6: Implement and demonstrate the use of the following in Solidity :</b></p>		

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

831e1581aac7c41a0da6246b92d1948acc00aafb70f233079dc0bcd395f2cbd0399e9506c18c45192d7a27e15dccbf32d81ca6e74719e5ba083866b782f9dd7d5748c90b6985147f1da709b632c8eb7

## 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 = ""

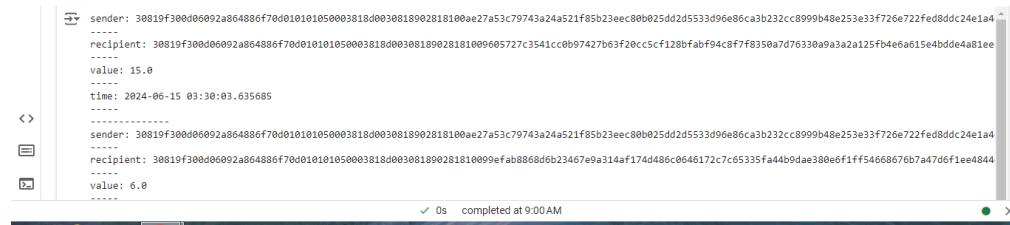
Dinesh = Client()

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: 30819f300d06092a864886f70d010101050003818d0030818902818100ae27a53c79743a24a521f85b23eec80b025dd2d5533d906e86ca3b232cc8999b48e253e33f726e722fed8ddc24e1a4
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181000605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7fb350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee
-----
value: 15.0
-----
time: 2024-06-15 03:30:03.635685
-----
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100ae27a53c79743a24a521f85b23eec80b025dd2d5533d906e86ca3b232cc8999b48e253e33f726e722fed8ddc24e1a4
-----
recipient: 30819f300d06092a864886f70d010101050003818d003081890281810099efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
-----
value: 6.0
-----
```

The terminal output displays two transactions. The first transaction is from Dinesh with a value of 500.0, and the second is a smaller transaction with a value of 6.0. Each transaction is shown with its sender, recipient, time, and a long hexidecimal hash.

```
recipient: 30819f300d06092a864886f7d010101050003818d0030818902818100965727c3541cc0b97427b63f28cc5cf128bfabf94c8f7f8350a7d7633e
value: 4.0
time: 2024-06-15 03:30:03.641381
-----
sender: 30819f300d06092a864886f7d010101050003818d0030818902818100d3bbd7ddcd8f706e5ba44d1a306f7e270ef98b6afdf78a4575401ae791f3c9cb366d8b24715f14406db035elb13
recipient: 30819f300d06092a864886f7d010101050003818d0030818902818100999fa88686d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
value: 7.0
time: 2024-06-15 03:30:03.642927
-----
sender: 30819f300d06092a864886f7d010101050003818d0030818902818100965727c3541cc0b97427b63f28cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208
recipient: 30819f300d06092a864886f7d010101050003818d0030818902818100999fa88686d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee4844
value: 3.0
time: 2024-06-15 03:30:03.644313
-----
sender: 30819f300d06092a864886f7d010101050003818d0030818902818100999fa88686d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484450d
recipient: 30819f300d06092a864886f7d010101050003818d0030818902818100999fa88686d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484450e
```

## 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: 30819f300d06092a864886f70d01010105000381d0030818902818100e80c42c3713c2091cd14fb0efcf09ba16339ea5929eccc979e580e5500f995aebe54702d45d19cb6be405e5d4cb7f820926b399be61c16
value: 500.0
-----
time: 2024-06-15 03:30:03.722966
-----
-----
=====
block # 1

```

The terminal window also includes standard file navigation icons (back, forward, search, etc.) and a progress bar at the bottom indicating the command completed at 9:19 AM.

```
blocks
  Number of blocks in the chain: 4
  block # 0
    sender: Genesis
    -----
    recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100e80<42c3713c2091cd14fb0efcf09ba16339ea5929ecc979e589e5508f995aebe54702d45d19cb6be405e5d4cb7f820926b399be61c16
    -----
    value: 500.0
    -----
    time: 2024-06-15 03:30:03.722966
    -----
    =====
    block # 1
    sender: 30819f300d06092a864886f70d010101050003818d0030818902818100e27a5c79743a24a521f85b23ecc80b025dd2d5533d96e86ca3b232cc8999b48e253e33f726e722fed8ddc24e1a46d3970c761a98bf87fdb
    -----
    recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100e605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208f56f61f1ed8f28d22
    -----
    value: 15.0
    -----
    time: 2024-06-15 03:30:03.635685
    -----
    sender: 30819f300d06092a864886f70d010101050003818d0030818902818100e27a5c79743a24a521f85b23ecc80b025dd2d5533d96e86ca3b232cc8999b48e253e33f726e722fed8ddc24e1a46d3970c761a98bf87fdb
    -----
    recipient: 30819f300d06092a864886f70d010101050003818d003081890281810099efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460de1b9496bc3372582c
    -----
    value: 6.0
    -----
    time: 2024-06-15 03:30:03.637914
    -----
    sender: 30819f300d06092a864886f70d010101050003818d00308189028181009605727c3541cc0b97427b63f20cc5cf128bfabf94c8f7f8350a7d76330a9a3a2a125fb4e6a615e4bdde4a81ee208f56f61f1ed8f28d22100
    -----
    value: 0s completed at 9:19AM
```

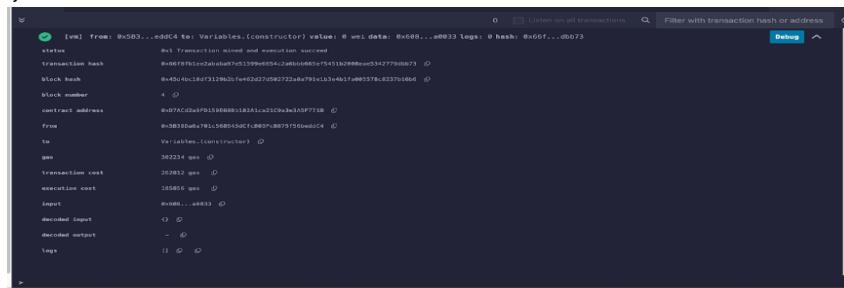
```
blocks
  recipient: 30819f300d06092a864886f70d010101050003818d003081890281810099efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460de1b9496bc3372582c126
  -----
  value: 6.0
  -----
  time: 2024-06-15 03:30:03.637914
  -----
  sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d3bbd7ddcd8f706e5ba44d1a306f7e270ef98b6af78a4575401ae791f34c9ccb366db24715f14406db035e1b13dda25d1ae6c2d6fc9
  -----
  recipient: 30819f300d06092a864886f70d010101050003818d003081890281810099efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460de1b9496bc3372582c
  -----
  value: 2.0
  -----
  time: 2024-06-15 03:30:03.639731
  -----
  sender: 30819f300d06092a864886f70d010101050003818d003081890281810099efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460de1b9496bc3372582c126
  -----
  recipient: 30819f300d06092a864886f70d010101050003818d003081890281810099efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460de1b9496bc3372582c
  -----
  value: 4.0
  -----
  time: 2024-06-15 03:30:03.641381
  -----
  sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d3bbd7ddcd8f706e5ba44d1a306f7e270ef98b6af78a4575401ae791f34c9ccb366db24715f14406db035e1b13dda25d1ae6c2d6fc9495
  -----
  recipient: 30819f300d06092a864886f70d010101050003818d003081890281810099efab8868d6b23467e9a314af174d486c0646172c7c65335fa44b9dae380e6f1ff54668676b7a47d6f1ee484460de1b9496bc3372582c
  -----
  value: 7.0
  -----
  value: 0s completed at 9:19AM
```

## 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;
}
```

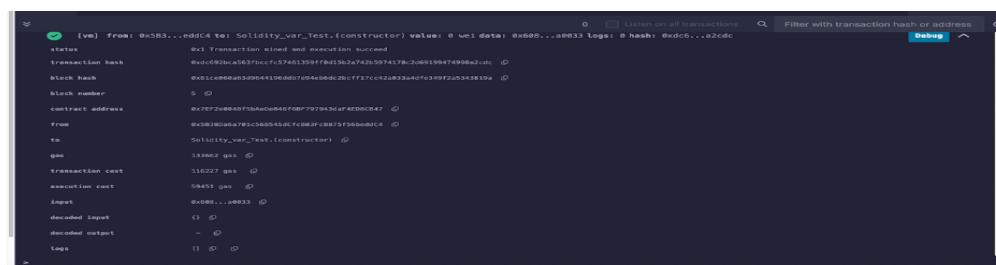


// Solidity program to demonstrate state variables

```
// 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;
    }
}
```

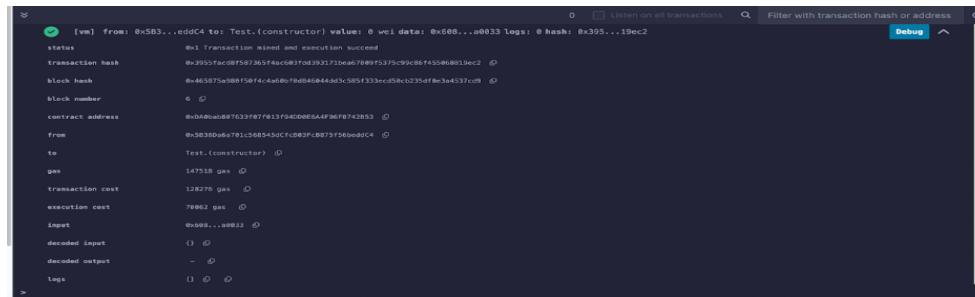
**OUTPUT:**



## // 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:



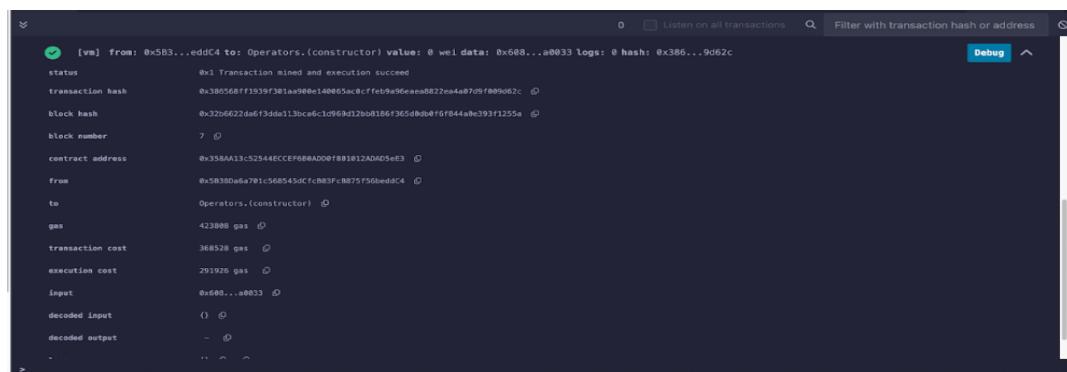
The screenshot shows the Truffle UI interface with the transaction details for the Test contract's constructor. The transaction hash is 0x39557fac0ff5f873b5f4acc08f3fd393171beac6788bf75375c99c88f43598e8d19ec2. The status indicates it was mined and succeeded. The gas used was 147518, and the transaction cost was 128276 gas.

## (II).Operators

### // Solidity contract to demonstrate Arithmetic 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 :

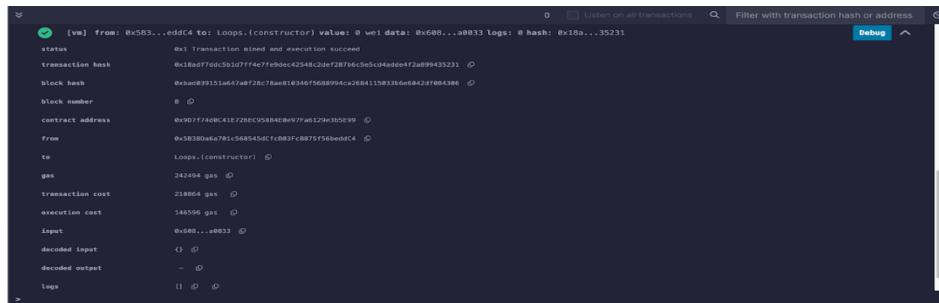


The screenshot shows the Truffle UI interface with the transaction details for the Operators contract's constructor. The transaction hash is 0x395568ff1939f7381aa988e140865ac8cffeb9a95eaa0822ea4a87d9f089d62c. The status indicates it was mined and succeeded. The gas used was 423808, and the transaction cost was 368528 gas.

### (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 l;
    }
}
```

### OUTPUT:



The screenshot shows a transaction details interface for a Solidity contract named 'Loops'. The transaction hash is 0x5b3...addc4, and it was successful. The gas used was 242404 gas, and the transaction cost was 218064 gas. The input data was 0x60...a0833, which corresponds to the constructor signature. The output is empty, indicated by '-'. The logs field is also empty.

Field	Value
status	0x1 Transaction mined and execution succeed
transaction hash	0x5b3...addc4
block hash	0xbacd839151ad47a#f2Bc7Ba#81e346f56688994c#268415833b6e#842d#08430#
block number	0
contract address	0xc0Df77460cA1C726Ec05884Cb97fa6129e3D5e99
from	0x5b3B0da8e781c568545dCt#883Fc#8875f56bedd4
to	Loops.(constructor)
gas	242404 gas
transaction cost	218064 gas
execution cost	144596 gas
input	0x60...a0833
decoded input	<>
decoded output	-
logs	[]

## (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:**

```
(bin) From: 0x0000...0000 to: 0x0000...0000 (constructor) value: 0 wei data: 0x0000...0000 logs: 0 hash: 0x0000...0000
status: 0x0000000000000000000000000000000000000000000000000000000000000000
transaction hash: 0x000000000000000000000000000000000000000000000000000000000000000
block hash: 0x000000000000000000000000000000000000000000000000000000000000000
block number: 0
contract address: 0x0000000000000000000000000000000000000000
from: 0x0000000000000000000000000000000000000000
to: 0x0000000000000000000000000000000000000000
gas: 135296 gas
transaction cost: 137578 gas
execution cost: 66096 gas
Input: 0x0000...cde832
decoded Input: -
decoded Output: -
Logs: 0x0000...0000
```

// 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:**

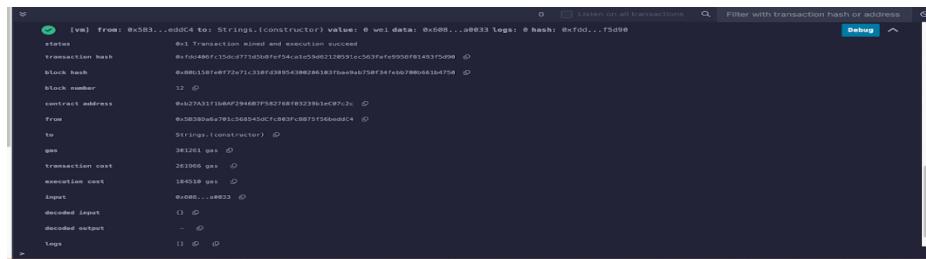
```
(bin) From: 0x0000...0000 to: 0x0000...0000 (constructor) value: 0 wei data: 0x0000...0000 logs: 0 hash: 0x0000...0000
status: 0x000000000000000000000000000000000000000000000000000000000000000
transaction hash: 0x000000000000000000000000000000000000000000000000000000000000000
block hash: 0x000000000000000000000000000000000000000000000000000000000000000
block number: 0
contract address: 0x0000000000000000000000000000000000000000
from: 0x0000000000000000000000000000000000000000
to: 0x0000000000000000000000000000000000000000
gas: 135296 gas
transaction cost: 137578 gas
execution cost: 66060 gas
Input: 0x0000...cde832
decoded Input: -
decoded Output: -
Logs: 0x0000...0000
```

## (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:

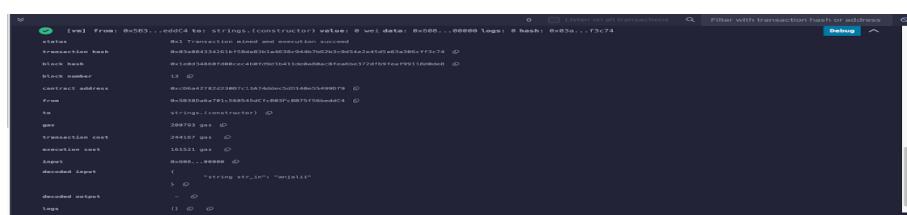


The screenshot shows the Truffle UI interface. In the top bar, it says "[vm] From: 0x5b3...addc4 to: Strings.(constructor) value: 0 wei data: 0x000...a0003 logs: 0 hash: 0xf0d...15d90". Below this, there's a table with columns: status, transaction hash, block hash, block number, contract address, from, to, gas, transaction cost, execution cost, input, decoded input, decoded output, and logs. The 'status' row indicates "0x1 Transaction mined and execution succeeded". The 'gas' row shows 381261 gas used. The 'logs' row contains the output of the function call, which is "ANJALI".

// 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:



The screenshot shows the Truffle UI interface. In the top bar, it says "[vm] From: 0x5b3...addc4 to: strings.(constructor) value: 0 wei data: 0x000...00000 logs: 0 hash: 0x0...1f3c74". Below this, there's a table with columns: status, transaction hash, block hash, block number, contract address, from, to, gas, transaction cost, execution cost, input, decoded input, decoded output, and logs. The 'status' row indicates "0x1 Transaction mined and execution succeeded". The 'gas' row shows 288750 gas used. The 'logs' row contains the output of the function call, which is "ANJALI".

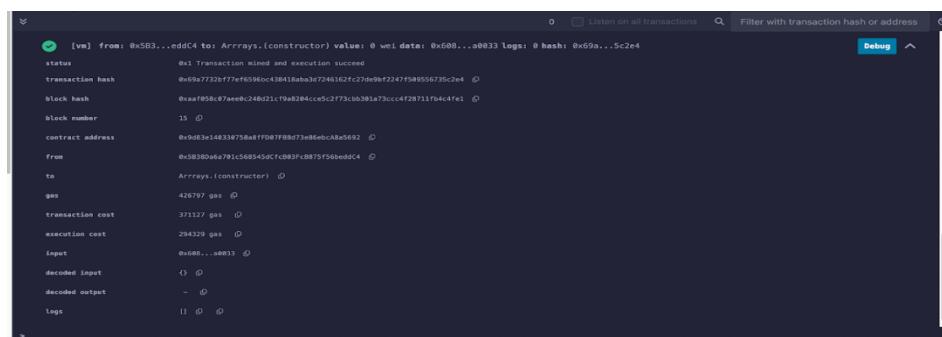
## PRACTICAL NO : 5

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:



The screenshot shows a transaction details page. Key information includes:  
- Status: 0x1 Transaction mined and execution succeed  
- Transaction hash: 0x09a7732bf77ef65960c38418aba3d7246102fc37de9bf2247f509556735c2e4  
- Block hash: 0xae0f05dc87aeeb0c248d21c9fb8284cc5c2f73cb38a73cc4f28711fb4c4fe1  
- Block number: 15  
- Contract address: 0xb03e148338758a87FD07F88d073e88ebcA8d0692  
- From: 0x5B30a6d781c569545dCt083fC8875f566edc4  
- To: Arrays.(constructor)  
- Gas: 426797 gas  
- Transaction cost: 371127 gas  
- Execution cost: 294329 gas  
- Input: 0x60...a003  
- Decoded input: ()  
- Decoded output: -  
- Logs: 0x00...a003

### (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:

VM status: 0x1 Transaction mined and execution succeed  
 transaction hash: 0xc108d5795ceec13d55f5ef5602b6211a9593b6074fc4f747c3b9a049640a6f  
 block hash: 0x7134ab040fa63a942e2cb80c;e385tbeen4cab09e68a815f38223e67384a54b  
 block number: 16  
 contract address: 0x04fc541236927e21af8f2760bb7389c1fc2cbee  
 from: 0x5B30ba6a791c568545dCfc0803fc0875f56beddC4  
 to: enums.(constructor)  
 gas: 177417 gas  
 transaction cost: 154275 gas  
 execution cost: 93939 gas  
 input: 0x600...a0033  
 decoded input: ()  
 decoded output: -  
 logs: ()

## (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;
    }
}

```

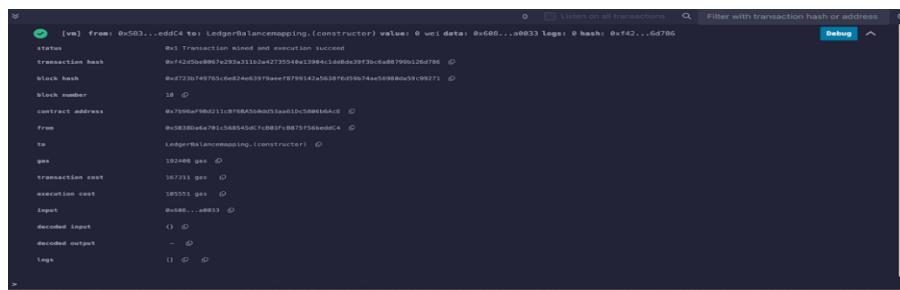
## OUTPUT:

VM status: 0x1 Transaction mined and execution succeed  
 transaction hash: 0xad43e9081155011d6050597832c7f838a1c77a83a4110592cd972c085230b0  
 block hash: 0x9b9c2a920039f2d04fc731a22c808f582a537e2a78fdaf4edfb447c2f596494  
 block number: 17  
 contract address: 0x5f06eb55012e759a21c09ef783fe0Ba1Dc9d880  
 from: 0x5B30ba6a791c568545dCfc0803fc0875f56beddC4  
 to: Bookstruct.(constructor)  
 gas: 475540 gas  
 transaction cost: 413520 gas  
 execution cost: 334772 gas  
 input: 0x600...a0033  
 decoded input: ()  
 decoded output: -  
 logs: ()

## (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:



[vm] from: 0x503...edd4 to: LedgerBalancemapping.(constructor) value: 0 wei data: 0x608...a0833 logs: 0 hash: 0xf42...6d786

status Ex1 Transaction mined and execution succeeded

transaction hash 0xf42c05e0887e29b311b2e42735548e13994c1cdd8d3a39f3bc8ab87995b126d786

block hash 0x7230746765c6824e63979aeef7f795142a5638f6d59974ee56088ba593c9271

block number 18

contract address 0x700af98bd211c87f8a558db0d53aaed13c508916d4c8

from 0xd380ade4781c5685454dcfc0887fc0875f56bedd4

to LedgerBalancemapping.(constructor)

gas 192488 gas

transaction cost 167231 gas

execution cost 188551 gas

input 0x608...a0833

decoded input ()

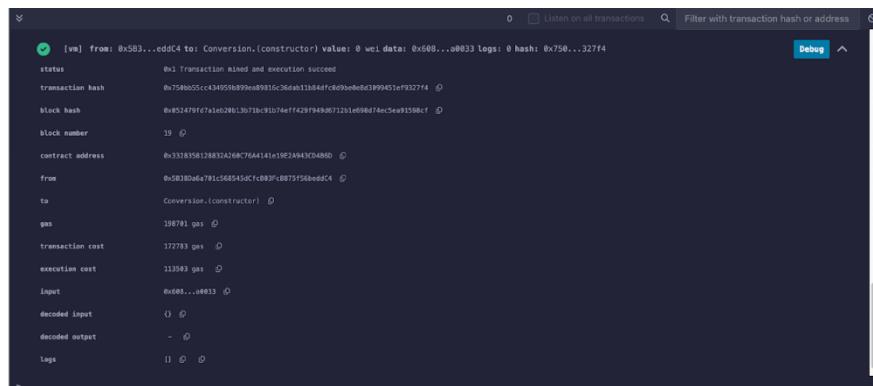
decoded output -

logs ()

## (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:



[vm] from: 0x503...edd4 to: Conversion.(constructor) value: 0 wei data: 0x608...a0833 logs: 0 hash: 0x750...327f4

status Ex1 Transaction mined and execution succeeded

transaction hash 0x750bb5cc434959b999ea0915c36daab1b84dfc8d99eeb6399945ef932774

block hash 0x8d2479f67a1e028b1b071bc91b74ef7429f94906712b1ed698d74ec5ea91598cf

block number 19

contract address 0x37383581288312a7e8c7844141e19f24943c0488d

from 0x5030ad6791c5685454dcfc0887fc0875f56bedd4

to Conversion.(constructor)

gas 198791 gas

transaction cost 172783 gas

execution cost 113843 gas

input 0x608...a0833

decoded input ()

decoded output -

logs ()

## (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 == 10000000000000000000000000000000) // 1 Ether equals to 1000000000000000000000000 Wei

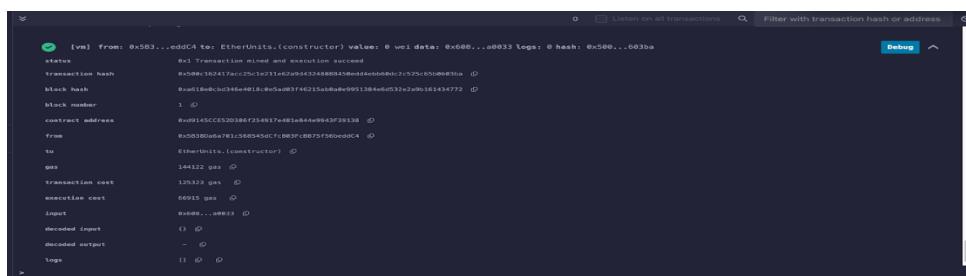
c=true;

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:

The screenshot shows a transaction details page from a blockchain explorer. The transaction hash is 0x895c731780f1dfb24d518080898798dfe7746e25f69e8e30cf54bd4f3e01ee. It was mined by block 0x5235b1320437fe995cb67997a0dc3079390aa300b1726a7ed8ec44e3fe7fc33 at block number 2. The transaction cost was 155372 gas, and the execution cost was 57712 gas. The input was 0x600...a0033, and the output was 0x05...e01ee. The logs were empty.

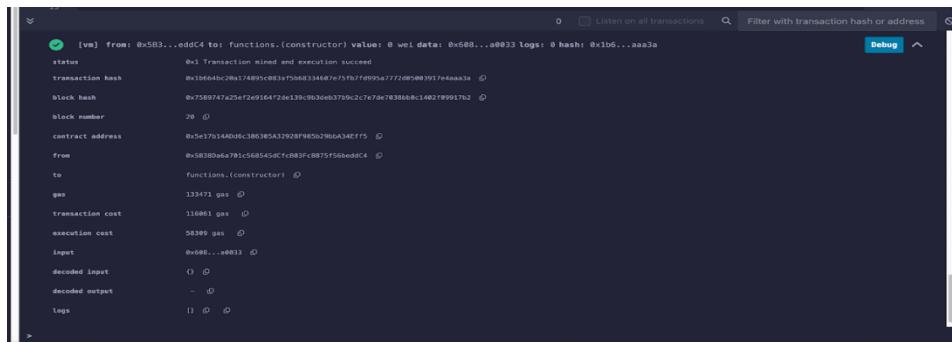
## PRACTICAL NO : 6

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:

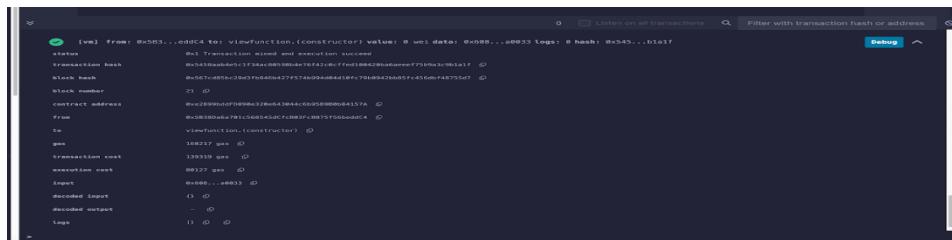


The screenshot shows a transaction details page from a blockchain interface. The transaction is identified by its hash: 0x5b3...edd4. It was mined and executed successfully. The transaction cost was 116861 gas, and the execution cost was 56399 gas. The input data was 0x60...a033. The transaction was sent from address 0x5b3...edd4 to the contract's constructor. The logs field is empty.

### (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:



The screenshot shows a transaction details page from a blockchain interface. The transaction is identified by its hash: 0x5b3...edd4. It was mined and executed successfully. The transaction cost was 108217 gas, and the execution cost was 89339 gas. The input data was 0x60...a033. The transaction was sent from address 0x5b3...edd4 to the viewfunction's constructor. The logs field is empty.

### (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:**

[vm] from: 0x5B3...eddC4 to: C.(constructor) value: 0 wei data: 0x600...a0033 logs: 0 hash: 0xe23...08b76	
status	0x1 Transaction mined and execution succeed
transaction hash	0xe23909c3df1b10d0aa4ebc10d92f4ax3be228ed2d3f043750d20254259f508b76
block hash	0xc5e4a038471ca9b7ea84e08a81c916c72623ef1f94dc67a3d0303374834
block number	22
contract address	0x1c0147f2444538ce6245308E0d9a097c562b410
from	0x5B30a6a781c568545dCfcB#3FcB#75f56bedd4
to	C.(constructor)
gas	170721 gas
transaction cost	155469 gas
execution cost	96235 gas
input	0x600...a0033
decoded input	{}
decoded output	-
logs	[]

## (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:

[vm] from: 0x5B3...eddC4 to: functionoverloading.(constructor) value: 0 wei data: 0x600...a0033 logs: 0 hash: 0xb2b...3874d	
status	0x1 Transaction mined and execution succeed
transaction hash	0xb2e03782ef1074491fed1fe3fa09f82f3b97eab28721232e610bd13874d
block hash	0x015b46d0bde4bc34c5e15ca2f60fb0e9777ea1c281283bf1098cc5367e78
block number	23
contract address	0x53f7b6dd587b7d083323723594e30202a7c96cc
from	0x5B30a6a781c568545dCfcB#3FcB#75f56bedd4
to	functionoverloading.(constructor)
gas	23142 gas
transaction cost	207000 gas
execution cost	147788 gas
input	0x600...a0033
decoded input	{}
decoded output	-
logs	[]

## (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:

```
0x1 Transaction mined and execution succeed
0xd7d59e9fa1bc6cd4c675cc5b3bfcaad5fffc9316be117f98862bb71695a56
0x63da59eb092267d8230a8e4ac4591c3188f31e60a8fa198c599fc18e7ebbad1
24
0x5A6058aA3b599f06663c2296741ef4cc0BC4d81
0x5838abaa787c568545dC1cb83Fc088f5f5bbedd4
mathematicalfunction.(constructor)
1347155 gas
117423 gas
59389 gas
0x608...a0033
-
[]
```

## (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:

```
0x1 Transaction mined and execution succeed
0x2e06176d32c78770c4708ec47316808b0f135d878053872951c1ae52e104
0xd5d4f55c089c0481ac2d1cc2e247f5fbfb4a77e7896431dd7f8fc2baedee2
25
0x5A6058aA3b599f06663c2296741ef4cc0BC4d81
0x5838abaa787c568545dC1cb83Fc088f5f5bbedd4
cryptographicfunction.(constructor)
133607 gas
368023 gas
368051 gas
0x608...a0033
-
[]
```

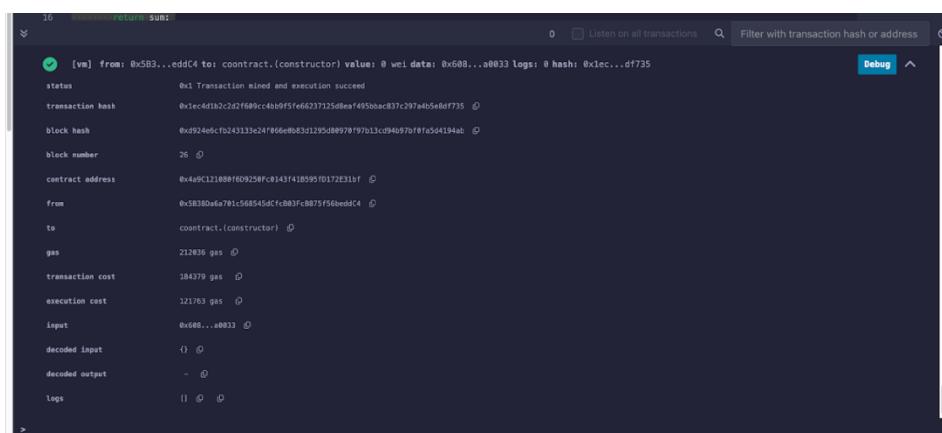
## PRACTICAL NO:7

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:



status	0x1 Transaction mined and execution succeeded
transaction hash	0x1ec4d1b2c2df689cc8b0f5fe682371250earf4950bae837c97a4b5e8dff735
block hash	0xd924edcfb243133c247866e0b351285d8897897b13cd94097b0f8fa5d4194eb
block number	26
contract address	0x4a9c121888f609250fc8143f418595f0172e31bf
from	0x5030d6a781c568545dcfc803fc8075f50be0dC4
to	coontract.(constructor)
gas	212030 gas
transaction cost	184379 gas
execution cost	212763 gas
input	0x60...a0033
decoded input	{}
decoded output	-
logs	[]

### (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:

Field	Value
status	0x1 Transaction mined and execution succeed
transaction hash	0x1c1c2f7fb60d149fb0b491f959e880ebd49f0802723f7c7c65c0082ca78e216981
block hash	0xd4fc001fb4cc8bd934de024ca411930b8f4317bdaffd213dcfa7aebd01ab
block number	27
contract address	0x54bd7e42805267858ff08f2551cc0575103c7569
from	0x50303a6a781c568545cf0883fc0075f560ed0c4
to	Caller.(constructor)
gas	40218 gas
transaction cost	358622 gas
execution cost	277872 gas
input	0x608...a0833
decoded input	{}
decoded output	-
logs	[ ]

## (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:

[vm] from: 0x583...eddC4 to: ConstructorExample.(constructor) value: 0 wei data: 0x600...a0033 logs: 0x205...a7351

status 0x1 Transaction mined and execution succeed

transaction hash 0x2050dfcc29c68648b496665eb45ff838fbe7ba54f38fdcfde1c91c682da7361

block hash 0xe9c5188650e4f440a11478145b7e36cf258725d06ec9600183c708ec640582

block number 28

contract address 0xEf9f1AC83dfb88f5990a621faEA72C66B10e8f

from 0x58300da791c568545dC7cd83fc8875f56bcddC4

to ConstructorExample.(constructor)

gas 272915 gas

transaction cost 237317 gas

execution cost 168751 gas

input 0x600...a0033

decoded input {}

decoded output -

logs ()

## (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;

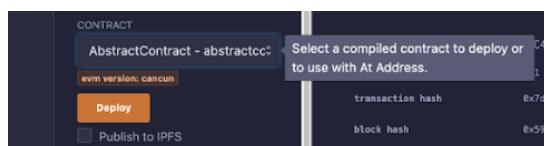
    function TestAbstract() public

    {
        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;

    }

}
```

```

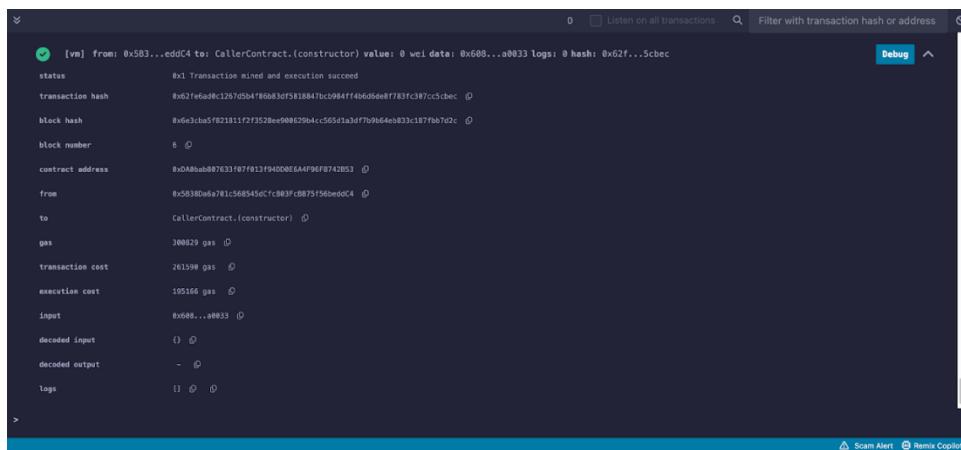
contract CallerContract

{
    TestContract tc = new TestContract();

    function getValues() public view returns(uint)
    {
        return tc.getResult();
    }
}

```

## OUTPUT:



The screenshot shows the transaction details for a successful call from the CallerContract to the TestContract. The transaction hash is 0x627ef0a0c1261d5d4f1fb083df818841cc0984ff466dbde8f783fc387cc5cbecc. The status is 0x1 Transaction mined and execution succeed. The contract address is 0xDABbb0087633f07f013f940DEEA4f9bf8742B53. The input was 0x600...a0033. The output is -. The logs are empty. The transaction cost was 261598 gas, and the execution cost was 105166 gas.

Parameter	Value
status	0x1 Transaction mined and execution succeed
transaction hash	0x627ef0a0c1261d5d4f1fb083df818841cc0984ff466dbde8f783fc387cc5cbecc
block hash	0x6e3cb05f821811f2f3528ee90882064:c565d1a3d7b9064e0833c187fb67d2c
block number	6
contract address	0xDABbb0087633f07f013f940DEEA4f9bf8742B53
from	0x50380ea791c5685450cfc983fc8875f565ed0C4
to	CallerContract.(constructor)
gas	300029 gas
transaction cost	261598 gas
execution cost	105166 gas
input	0x600...a0033
decoded input	-
decoded output	-
logs	[]

## 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";
    }

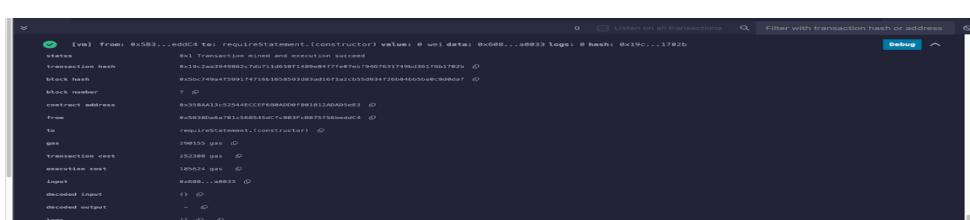
    // Defining function to use require statement

    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:**

```

[vm] From: 0x503...ed0C4 to AssertStatement.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x48b...bf964
status 0x1 Transaction mined and execution succeed
transaction hash 0x400abf46c70976ab50e9c1599334186f6822f7fc625f99fe1ea7958c02b7f964
block hash 0xa9497553b2bjeclfaa2ce1d415482b01875663929a7730b46a27b0a55366f5d7
block number 8
contract address 0x90977400C413726EC0958840e97Fa6120e3b5E99
from 0x50300da781c500540cf7083fc0875f56bed0C4
to AssertStatement.(constructor)
gas 256432 gas
transaction cost 222984 gas
execution cost 158200 gss
input 0x608...a0033
decoded input () 
decoded output -()
logs []

```

## 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);

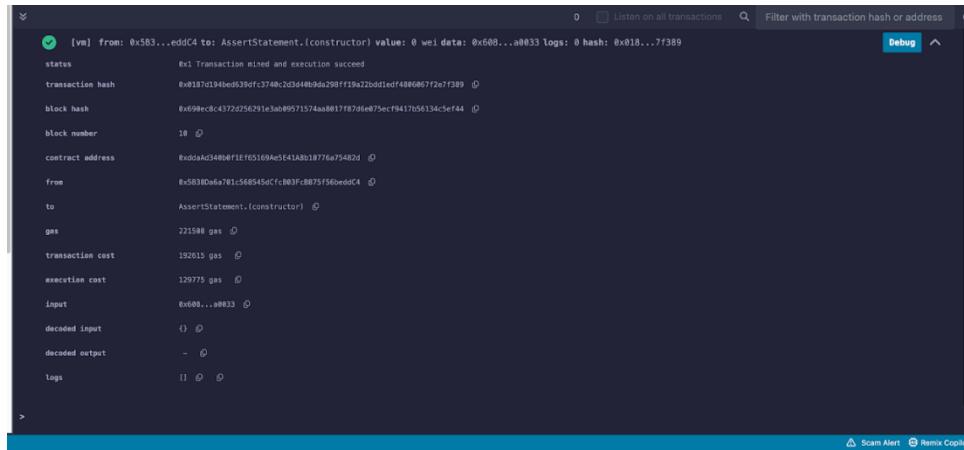
        result = true;
    }

    // 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:



The screenshot shows the Remix IDE interface with a transaction details panel. The transaction is successful, indicated by a green checkmark. Key details include:

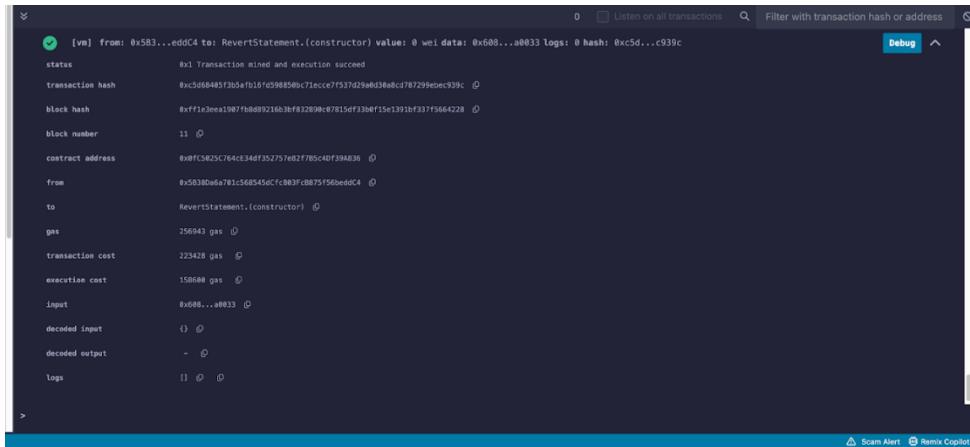
- from: 0x583...edd4
- to: AssertStatement.(constructor)
- value: 0 wei
- data: 0x60...a0033
- hash: 0x018...7f389
- status: 0x1 Transaction mined and execution succeed
- transaction hash: 0x817c194be0639dfc374ec203d46b0da208f19a22bd1edf408006772e7f389
- block hash: 0x698ec8c4372d256291e3b040571574aa0017f87dfe875ecf9417b56134c5ef44
- block number: 10
- contract address: 0xdca4d3480d71ef05169a5f41a8018776a75482d
- from: 0x58380a6a761c568845dcfc803fc8875f560e0edc4
- to: AssertStatement.(constructor)
- gas: 221598 gas
- transaction cost: 192615 gas
- execution cost: 129775 gas
- input: 0x00...a0033
- decoded input: ()
- decoded output: -
- logs: []

## 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:



The screenshot shows the Remix IDE interface with the transaction details for a revert statement. The transaction was mined and succeeded, with the following key details:

- From: 0x5b3...eddC4
- To: RevertStatement.(constructor)
- Value: 0 wei
- Data: 0x600...a0033
- Hash: 0xc5d...c939c
- Status: 0x1 (Transaction mined and execution succeeded)
- Block Hash: 0x5d66485f305efb10fd59885d0...71eccc7f537d294bd348c1787294dec939c
- Block Number: 11
- Contract Address: 0x9fcf025c764ce34df352757e82f785<40f394836
- From: 0x5b380d6a781c5685456cfcb803fc8075f56bedd0C4
- To: RevertStatement.(constructor)
- Gas: 25941 gas
- Transaction Cost: 223428 gas
- Execution Cost: 150600 gas
- Input: 0x608...a0033
- Decoded Input: ()
- Decoded Output: -
- Logs: []

At the bottom right, there are links for "Scam Alert" and "Remix Copilot".