# INDEX

| Sr. No. | Date | Topic | Page No. | Signa ture |
|---|---|---|---|---|
| 1 | | Write the following programs for Blockchain in Python:<br>a. A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it.<br>b. A transaction class to send and receive money and test it.<br>c. Create multiple transactions and display them.<br>d. Create a blockchain, a genesis block and execute it.<br>e. Create a mining function and test it.<br>f. Add blocks to the miner and dump the blockchain. | | |
| 2 | | Install and configure Go Ethereum and the Mist browser.Develop and test a sample application. | | |
| 3 | | Implement and demonstrate the use of the following in Solidity: Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. | | |
| 4 | | Implement and demonstrate the use of the following in Solidity: Functions, Function Modifiers, View Functions, Pure Functions, Fallback Function, Mathematical Functions, Cryptographic Functions. | | |
| 5 | | Implement and demonstrate the use of the following in Solidity: Withdrawal Pattern, Restricted Access. | | |
| 6 | | Implement and demonstrate the use of the following in Solidity: Contracts, Inheritance, Constructors, Abstract Contracts,Interfaces. | | |
| 7 | | Implement and demonstrate the use of the following in Solidity: Libraries, Assembly, Events, Error handling. | | |
| 8 | | Demonstrate the use of Bitcoin Core API. | | |

# Practical No: 1

Write the following programs for Blockchain in Python

## Practical 1 a)

**Aim:** To create a simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it.

## Code:

BC_Practical 1a.py - C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 1a.py (3.11.2)

File Edit Format Run Options Window Help

```
#!pip install pycryptodome

import binascii
import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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')
Demo = Client()
print (Demo.identity)
```

## Output:

Command Prompt

```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Cvman>pip install pycryptodome
Collecting pycryptodome
  Using cached pycryptodome-3.17-cp35-abi3-win_amd64.whl (1.7 MB)
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.17

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Cvman>python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\cvman\appdata\local\programs\python\python311\lib\site-packages (22.3.1)
Collecting pip
  Using cached pip-23.0.1-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
Successfully installed pip-23.0.1

C:\Users\Cvman>pip install pycryptodome
Requirement already satisfied: pycryptodome in c:\users\cvman\appdata\local\programs\python\python311\lib\site-packages
(3.17)

C:\Users\Cvman>
```

IDLE Shell 3.11.2

File Edit Shell Debug Options Window Help

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Cvman/AppData/Local/Programs/Python/Python311/BC_Practical 1a.py
30819f300d06092a864886f70d010101050003818d0030818902818100d77357d8f88e597fae8a9c7b71359bfdc14400a
0cec1bc2ed612e20be00580333fe31824764ebf8dfab6c170e6c75ce301e49d5dc49fedb029f8bf66be2cc6d0dbd2a292
8f10d15431056c51783da584c65e4670d093c49553e2fda8b1e4c43cbb310d880d7e792c0200f9104a709ad40552d3fd7
02163102398a27e9752be350203010001
>>>
```

## Practical 1 b)

**Aim:** To create a transaction class to send and receive money and test it.

**Code:**

BC_Practical 1b.py - C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 1b.py (3.11.2)

File Edit Format Run Options Window Help

```
import collections
import datetime
import binascii
#!pip install pycryptodome
import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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')

Raja = Client()
Rani = Client()
t = Transaction(Raja, Rani.identity, 5.0)

signature = t.sign_transaction()
print (signature)
```

**Output:**

```
======= RESTART: C:/Users/Cvman/AppData/Local/Programs/Python/Python311/BC_Practical 1b.py ======
82e31c6b22d13c36c9b24ce8156603f4870a32934b05144615cb894ddbcaaaf7d8ad7ee7fc55d480599dcc15333a1cf0c
7a4b0aa52fe0d036301f9e870bd6f35ea56e1fe09a5321ba32177564b419bfaafc69494cfa2fca37b83d5fdbbf1241f05
4ade074b953ea07fadab8c20eb007edee1d50d2239432a37ffcaec4e6f0171
>>>
```

## Practical 1 c)

**Aim:** To create multiple transactions and display them.

**Code:**

BC_Practical 1c.py - C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 1c.py (3.11.2)

File Edit Format Run Options Window Help

```
import collections
import datetime
import binascii
#!pip install pycryptodome
import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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')

    import hashlib

    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
```

```
class Block:
    def __init__(self):
        self.verified_transactions = []
        self.previous_block_hash = ""
        self.Nonce = ""

    last_block_hash = ""

    def display_transaction(transaction):
        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 ('-----')

    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 ('====================================')
    last_transaction_index = 0

    transactions = []

    Raja = Client()
    Rani = Client()
    Seema = Client()
    Reema = Client()
```

```
t1 = Transaction(Raja, Rani.identity, 15.0)
t1.sign_transaction()
transactions.append(t1)

t2 = Transaction(Raja,Seema.identity,6.0)
t2.sign_transaction()
transactions.append(t2)

t3 = Transaction(Rani,Reema.identity,2.0)
t3.sign_transaction()
transactions.append(t3)

t4 = Transaction(Seema,Rani.identity,4.0)
t4.sign_transaction()
transactions.append(t4)

t5 = Transaction(Reema,Seema.identity,7.0)
t5.sign_transaction()
transactions.append(t5)

t6 = Transaction(Rani,Seema.identity,3.0)
t6.sign_transaction()
transactions.append(t6)

t7 = Transaction(Seema,Raja.identity,8.0)
t7.sign_transaction()
transactions.append(t7)

t8 = Transaction(Seema,Rani.identity,1.0)
t8.sign_transaction()
transactions.append(t8)

t9 = Transaction(Reema,Raja.identity,5.0)
t9.sign_transaction()
transactions.append(t9)

t10 = Transaction(Reema,Rani.identity,3.0)
t10.sign_transaction()
transactions.append(t10)

for transaction in transactions:
    display_transaction (transaction)
    print ('--------------')
```

## Output:

```
>>>
============= RESTART: C:/Users/Cvman/AppData/Local/Programs/Python/Python311/BC_Practical 1c.py ============
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b02fd85a87c5032cc75474d23f8e25b4358e915165150d459a256c14d47d2bf1e
f9f5e289caa04a0eb7a49aa4cb6432bc5f60d81a26113043eee80d2bb403833a7f087d8132e1a9e0cc5d29818693be88eba23c6612cfecb0ade80517b22ccb97248
2ca879a893e4566a9fde5bd03f3cf8b302eaf7e6da57c3a508c2b6c301c10203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100bdfbca63fdb3a4aae6ddfb47bef494cc1142ff668f1d29e0f48248f2c67667
4f81a89125f07bb04d55ff46a8eeae2ed1277be59fe5cae5783d8ff2dff68d449d1acf75ab83a9c6baba3c4fe97108d2e238df0b168b8fcab68ce2abe6ab355b7a6
80aeb2b279afa9fea1a5480c14af01f160e20884dedc7c048d530d9959f39d10203010001
-----
value: 15.0
-----
time: 2023-03-20 17:22:28.841684
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b02fd85a87c5032cc75474d23f8e25b4358e915165150d459a256c14d47d2bf1e
f9f5e289caa04a0eb7a49aa4cb6432bc5f60d81a26113043eee80d2bb403833a7f087d8132e1a9e0cc5d29818693be88eba23c6612cfecb0ade80517b22ccb97248
2ca879a893e4566a9fde5bd03f3cf8b302eaf7e6da57c3a508c2b6c301c10203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100ac3dbd38a8f12574ebee5bccab4696fe3c1e425a4ea913e1493bcdbd36c1ea
3c8a865e445f39fa293a35429d81a21df87160869102e8ccbe6d374fa142e9ffa1e65877fe5e32dcfab57ae9e63ecdd0e90441c33f1f1064d1c0f39ef9672c976b6
7dfd805f6ec4b3837853c1325f1b3b652ca6ef3879fe9ee907d5e2e4eb377e70203010001
-----
value: 6.0
-----
time: 2023-03-20 17:22:28.841684
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100bdfbca63fdb3a4aae6ddfb47bef494cc1142ff668f1d29e0f48248f2c676674f8
1a89125f07bb04d55ff46a8eeae2ed1277be59fe5cae5783d8ff2dff68d449d1acf75ab83a9c6baba3c4fe97108d2e238df0b168b8fcab68ce2abe6ab355b7a680a
eb2b279afa9fea1a5480c14af01f160e20884dedc7c048d530d9959f39d10203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100be08744c2dcae71ee7c25f0c75cc5b56d311451693c75c7513feb0052f73cd
5978228ee8a25715ea7628eb131cb027be1e3106e8d5cbf65c93cd451ab43aa0c50881574ac5598e04f649da99e798c120d8850db4e45c106d91630b20772c4655f
bed433493c9438cd7401c528c5a8771ff2d066330e0e80f069b39da78a38abf0203010001
-----
value: 2.0
-----
time: 2023-03-20 17:22:28.841684
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100ac3dbd38a8f12574ebee5bccab4696fe3c1e425a4ea913e1493bcdbd36c1ea3c8
a865e445f39fa293a35429d81a21df87160869102e8ccbe6d374fa142e9ffa1e65877fe5e32dcfab57ae9e63ecdd0e90441c33f1f1064d1c0f39ef9672c976b67df
d805f6ec4b3837853c1325f1b3b652ca6ef3879fe9ee907d5e2e4eb377e70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100bdfbca63fdb3a4aae6ddfb47bef494cc1142ff668f1d29e0f48248f2c67667
4f81a89125f07bb04d55ff46a8eeae2ed1277be59fe5cae5783d8ff2dff68d449d1acf75ab83a9c6baba3c4fe97108d2e238df0b168b8fcab68ce2abe6ab355b7a6
80aeb2b279afa9fea1a5480c14af01f160e20884dedc7c048d530d9959f39d10203010001
-----
value: 4.0
-----
```

```
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100ac3dbd38a8f12574ebee5bccab4696fe3c1e425a4ea913e1493bcdbd36c1ea3c8
a865e445f39fa293a35429d81a21df87160869102e8ccbe6d374fa142e9ffa1e65877fe5e32dcfab57ae9e63ecdd0e90441c33f1f1064d1c0f39ef9672c976b67df
d805f6ec4b3837853c1325f1b3b652ca6ef3879fe9ee907d5e2e4eb377e70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b02fd85a87c5032cc75474d23f8e25b4358e915165150d459a256c14d47d2b
f1ef9f5e289caa04a0eb7a49aa4cb6432bc5f60d81a26113043eee80d2bb403833a7f087d8132e1a9e0cc5d29818693be88eba23c6612cfecb0ade80517b22ccb97
2482ca879a893e4566a9fde5bd03f3cf8b302eaf7e6da57c3a508c2b6c301c10203010001
-----
value: 8.0
-----
time: 2023-03-20 17:22:28.841684
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100ac3dbd38a8f12574ebee5bccab4696fe3c1e425a4ea913e1493bcdbd36c1ea3c8
a865e445f39fa293a35429d81a21df87160869102e8ccbe6d374fa142e9ffa1e65877fe5e32dcfab57ae9e63ecdd0e90441c33f1f1064d1c0f39ef9672c976b67df
d805f6ec4b3837853c1325f1b3b652ca6ef3879fe9ee907d5e2e4eb377e70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100bdfbca63fdb3a4aae6ddfb47bef494cc1142ff668f1d29e0f48248f2c67667
4f81a89125f07bb04d55ff46a8eeae2ed1277be59fe5cae5783d8ff2dff68d449d1acf75ab83a9c6baba3c4fe97108d2e238df0b168b8fcab68ce2abe6ab355b7a6
80aeb2b279afa9fea1a5480c14af01f160e20884dedc7c048d530d9959f39d10203010001
-----
value: 1.0
-----
time: 2023-03-20 17:22:28.841684
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100be08744c2dcae71ee7c25f0c75cc5b56d311451693c75c7513feb0052f73cd597
8228ee8a25715ea7628eb131cb027be1e3106e8d5cbf65c93cd451ab43aa0c50881574ac5598e04f649da99e798c120d8850db4e45c106d91630b20772c4655fbed
433493c9438cd7401c528c5a8771ff2d066330e0e80f069b39da78a38abf0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b02fd85a87c5032cc75474d23f8e25b4358e915165150d459a256c14d47d2b
f1ef9f5e289caa04a0eb7a49aa4cb6432bc5f60d81a26113043eee80d2bb403833a7f087d8132e1a9e0cc5d29818693be88eba23c6612cfecb0ade80517b22ccb97
2482ca879a893e4566a9fde5bd03f3cf8b302eaf7e6da57c3a508c2b6c301c10203010001
-----
value: 5.0
-----
time: 2023-03-20 17:22:28.857309
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100be08744c2dcae71ee7c25f0c75cc5b56d311451693c75c7513feb0052f73cd597
8228ee8a25715ea7628eb131cb027be1e3106e8d5cbf65c93cd451ab43aa0c50881574ac5598e04f649da99e798c120d8850db4e45c106d91630b20772c4655fbed
433493c9438cd7401c528c5a8771ff2d066330e0e80f069b39da78a38abf0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100bdfbca63fdb3a4aae6ddfb47bef494cc1142ff668f1d29e0f48248f2c67667
4f81a89125f07bb04d55ff46a8eeae2ed1277be59fe5cae5783d8ff2dff68d449d1acf75ab83a9c6baba3c4fe97108d2e238df0b168b8fcab68ce2abe6ab355b7a6
80aeb2b279afa9fea1a5480c14af01f160e20884dedc7c048d530d9959f39d10203010001
-----
value: 3.0
-----
time: 2023-03-20 17:22:28.857309
-----
--------------
```

## Practical 1 d)

**Aim:** To create multiple transactions and display them.

**Code:**

BC_Practical 1d.py - C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 1d.py (3.11.2)

File Edit Format Run Options Window Help

```
import collections
import datetime
import binascii
#!pip install pycryptodome
import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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')

class Block:
    def __init__(self):
        self.verified_transactions=[]
        self.previous_block_hash=""
        self.Nonce=""

    last_block_hash=""
```

```
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('-----')

Rutuja = Client()
t0 = Transaction (
    "Genesis",
    Rutuja.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))

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:**

```
= RESTART: C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC
_Practical 1d.py
Number of blocks in the chain: 1
block # 0
sender:Genesis
-----
recipient:30819f300d06092a864886f70d010101050003818d0030818902818100
ebce501f177f40c6f813f886c46cf0b652203aaf5c161a52fc881d8e1ffb419e1413
fe7a1bcb0e61be44bbf8c89b7eb81db095c76dd041d26d8755b668de88cf79f7b386
c25912ad84c4f99314c7e0d013176e9927b6d12b7c9aee3a6517557c4ecf996e9b9a
44da4ec23a56d2413f279b8d9a4df98cc7b69e9dd5021359df9d0203010001
-----
value:500.0
-----
time:2023-03-25 19:57:13.114738
-----
--------------
====================================
```

**Practical 1 e)**

**Aim:** To create a mining function and test it.

**Code:**

BC_Practical 1e.py - C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 1e.py (3.11.2)

File Edit Format Run Options Window Help

```
import hashlib

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)+"iterationsfoundnonce:"+digest)
            return digest

mine("testmessage",2)
```

**Output:**

```
= RESTART: C:/Users/Cvman/AppData/Local/Programs/Python/Python311/BC_Practical 1e.py
after 165 iterations found nonce: 11c7ca07ab761d0f45e20ede827f738a4815fb27cccd6dd77a2d35c02731d808
>>>
========== RESTART: C:/Users/Cvman/AppData/Local/Programs/Python/Python311/BC_Practical 1e.py ==========
after 105 iterations found nonce: 1154f040b5973b68e29b8a28761f6950bf8d7f02f3f5c065fc6ffb832dba58b1
>>>
```

## Practical 1 f)

**Aim:** To add blocks to the miner and dump the blockchain.

**Code:**

BC_Practical 1f.py - C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 1f.py (3.11.2)

File Edit Format Run Options Window Help

```
import collections
import datetime
import binascii
#!pip install pycryptodome
import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
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')

import hashlib
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)+"iterationsfoundnonce:"+digest)
            return digest
```

```
class Block:
    def __init__(self):
        self.verified_transactions=[]
        self.previous_block_hash=""
        self.Nonce=""

last_block_hash=""
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('-----')

TPCoins=[]
def dump_blockchain(self):
    print("Numberofblocksinthechain:"+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('====================================')

last_transaction_index=0

transactions=[]

Raja=Client()
Rani=Client()
Seema=Client()
Reema=Client()
```

```
t1=Transaction(Raja,Rani.identity,15.0)
t1.sign_transaction()
transactions.append(t1)

t2=Transaction(Raja,Seema.identity,6.0)
t2.sign_transaction()
transactions.append(t2)

t3=Transaction(Rani,Reema.identity,2.0)
t3.sign_transaction()
transactions.append(t3)

t4=Transaction(Seema,Rani.identity,4.0)
t4.sign_transaction()
transactions.append(t4)

t5=Transaction(Reema,Seema.identity,7.0)
t5.sign_transaction()
transactions.append(t5)

t6=Transaction(Rani,Seema.identity,3.0)
t6.sign_transaction()
transactions.append(t6)

t7=Transaction(Seema,Raja.identity,8.0)
t7.sign_transaction()
transactions.append(t7)

t8=Transaction(Seema,Rani.identity,1.0)
t8.sign_transaction()
transactions.append(t8)

t9=Transaction(Reema,Raja.identity,5.0)
t9.sign_transaction()
transactions.append(t9)

t10=Transaction(Reema,Rani.identity,3.0)
t10.sign_transaction()
transactions.append(t10)
```

```
#Minerladdsablock

block=Block()
for i in range(3):
    temp_transaction=transactions[last_transaction_index]
    #validatetransaction
    #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

#Miner2 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

#Miner3 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

dump_blockchain(TPCoins)
```

**Output:**

```
= RESTART: C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 1
f.py
after266iterationsfoundnonce:11e61930905dfd9d58db0c2b63f4ee8921f9c2fb0f1cbf6e751
0917d16b16018
after369iterationsfoundnonce:115401fa5ee967b9f99260b146a2af8189934e854fc9d665fa9
8aa6507fd2a74
after444iterationsfoundnonce:11f2d02284f24b1ae4951a00ea5f5f854f51732e27735a28510
87e69b6c3f761
Numberofblocksinthechain:3
block#0
sender:30819f300d06092a864886f70d010101050003818d003081890281810092676295783906
7
37423558114730b7760877729cf28027f1710aa6fcad15cb35480a8c811d88acc4d3b17990127579
22eddf239ea50862ba19757bac2c7b56c649cd33f6bddace334fcf3c8e7acc256bd82637d88076ec
d0e6c5345a4fe3e81e8ac954d8da1f2b93d4ba8b6d5d59de342e6e5f42d171bf85e61f9e0ef281ca
f0203010001
-----
recipient:30819f300d06092a864886f70d010101050003818d00308189028181009f76b520cdf2
8b980f745fee13d554191ae75311ba7ed3110feed96828e66294022936e333727653a5853d79df76
6be64ca654d6171c8b54bc210cccb77c40f0c0bebd9978043aaacc84aa0e3d1a5fccc348c0e47844
b6324e5c93ce82e6e901b06bb17c7257adf0a04dee04cee8afa60300daa715d89cabbdb745a8c12d
c6350203010001
-----
value:15.0
-----
time:2023-03-25 20:00:58.070912
-----
--------------
=====================================
sender:30819f300d06092a864886f70d010101050003818d003081890281810092676295783906
7
37423558114730b7760877729cf28027f1710aa6fcad15cb35480a8c811d88acc4d3b17990127579
22eddf239ea50862ba19757bac2c7b56c649cd33f6bddace334fcf3c8e7acc256bd82637d88076ec
d0e6c5345a4fe3e81e8ac954d8da1f2b93d4ba8b6d5d59de342e6e5f42d171bf85e61f9e0ef281ca
f0203010001
-----
```

# Practical No: 2

**Practical No: 2**

**Aim:** Install and configure Go Ethereum and the Mist browser. Develop and test a sample application.

**Code:**

Installing GETH (Go Ethereum) :-

Step 1: Go to website https://geth.ethereum.org/downloads/
Step 2: From stable releases Geth 1.5.8 (kind = installer)
Step 3: once downloaded run it then click next
Step 4: Select Geth and Development tools click next
Step 5: Select location to install click next
Step 6: Once Installation is finished Click Close and its done

Installing Mist Browser :-

Step 1: https://github.com/ethereum/mist/releases
Step 2: Under Ethereum Wallet and Mist 0.8.9 - "The Wizard" download mist-installer-0-8- 9.exe
Step 3: For installation click, I agree -> next -> install

Run Mist :-

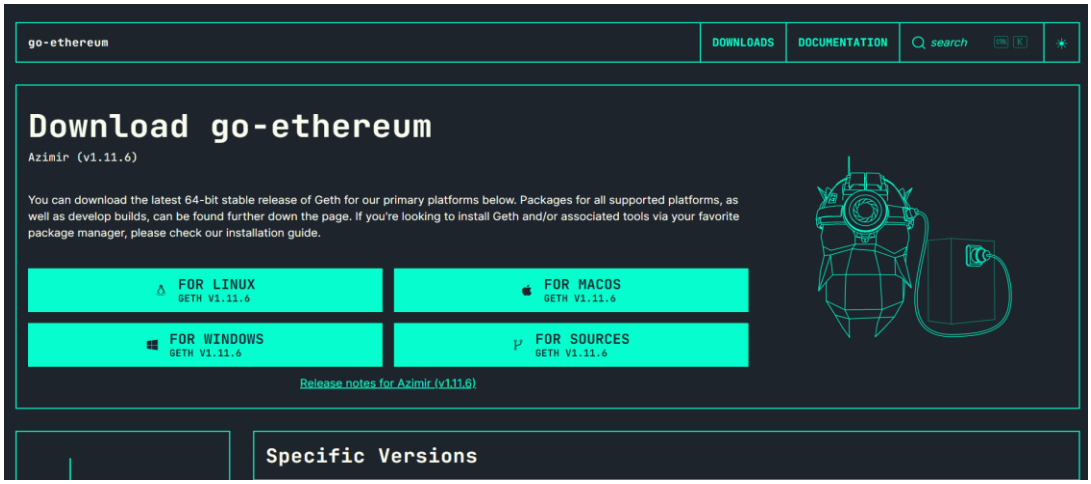
Step 1: Open the Mist from the start menu
Step 2: It will start downloading Blockchain data once you open it
Step 3: Once it finishes downloading it is ready to use

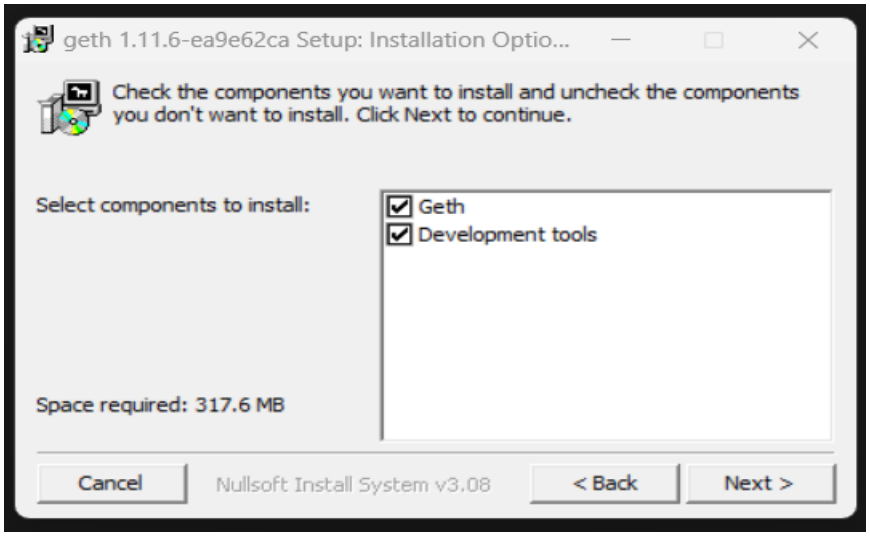
Run Geth :-

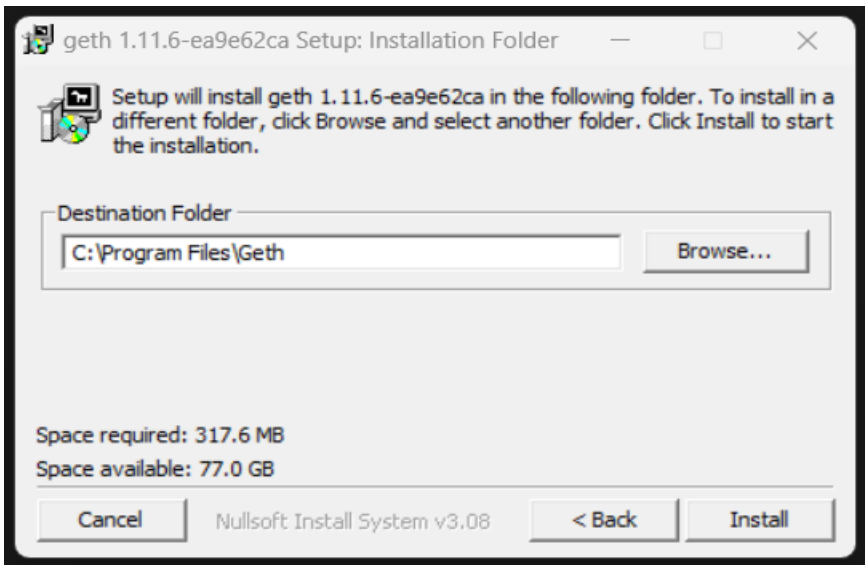
Step 1: Open CMD
Step 2: Type GETH and press enter
Step 3: After it finishes loading press ctrl+c to exit the process.
Step 4: Now it's ready to use

**Output:**

**INSTALLING GETH.**

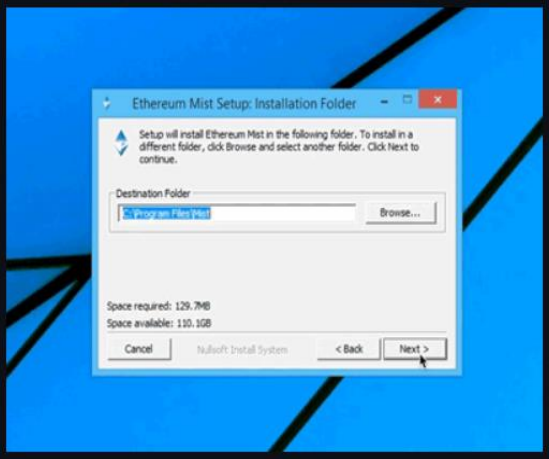


## INSTALLING MIST BROWSER.

# Ethereum Wallet and Mist 0.8.9 - "The Wizard"

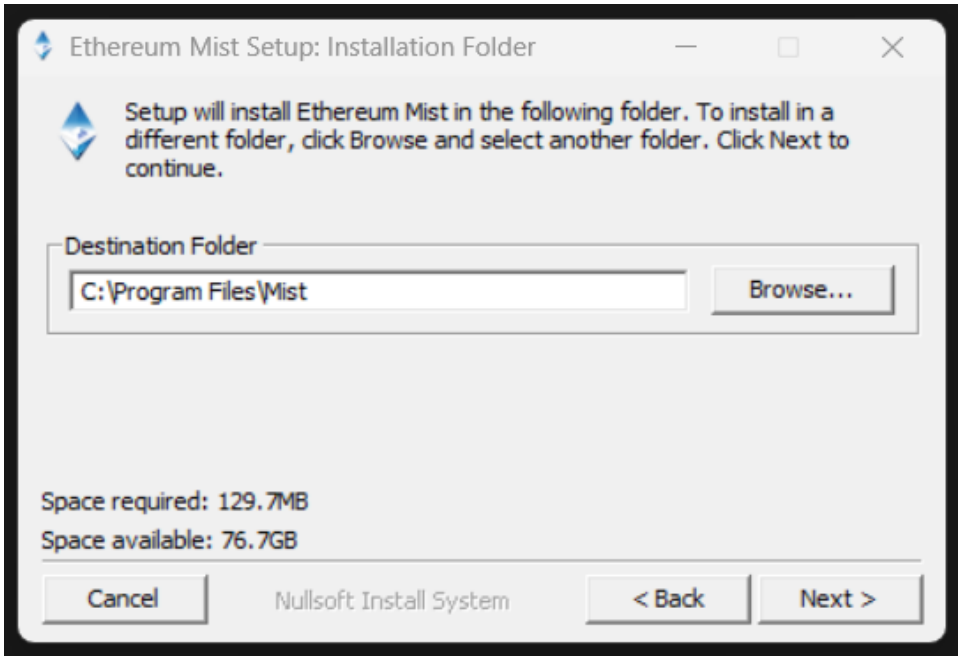



- **Full fledged Windows Installers**: This version includes the new installer for Windows created by **@tgerring**, which lets you choose the directory to install Mist in, as well as the `data-dir` of the ethereum node. It's one installer for both 32 and 64-bit computers.

> *Note that the `data-dir` is set as a parameter in Mist shortcut properties, at the installation stage; not in Mist's preferences.*

- **App Signing**: Mist for Mac OS X is now signed by the Ethereum Foundation.
- **Solidity Compiler**: Now featuring version 0.4.8.

| | | |
|---|---|---|
| Ethereum-Wallet-linux32-0-8-9.deb | 37.3 MB | Feb 2, 2017 |
| Ethereum-Wallet-linux32-0-8-9.zip | 54.3 MB | Feb 2, 2017 |
| Ethereum-Wallet-linux64-0-8-9.deb | 36.5 MB | Feb 2, 2017 |
| Ethereum-Wallet-linux64-0-8-9.zip | 53.3 MB | Feb 2, 2017 |
| Ethereum-Wallet-macosx-0-8-9.dmg | 55.7 MB | Feb 2, 2017 |
| Ethereum-Wallet-win32-0-8-9.zip | 54 MB | Feb 2, 2017 |
| Ethereum-Wallet-win64-0-8-9.zip | 78.2 MB | Feb 2, 2017 |
| mist-installer-0-8-9.exe | 129 MB | Feb 2, 2017 |
| Mist-linux32-0-8-9.deb | 36.7 MB | Feb 2, 2017 |
| Mist-linux32-0-8-9.zip | 52.2 MB | Feb 2, 2017 |
| Mist-linux64-0-8-9.deb | 35.9 MB | Feb 2, 2017 |
| Mist-linux64-0-8-9.zip | 51.1 MB | Feb 2, 2017 |
| Mist-macosx-0-8-9.dmg | 54 MB | Feb 2, 2017 |
| Mist-win32-0-8-9.zip | 51.8 MB | Feb 3, 2017 |
| Mist-win64-0-8-9.zip | 76 MB | Feb 3, 2017 |
| Source code (zip) | | Feb 2, 2017 |


Ethereum Mist Setup: Installation Folder
Setup will install Ethereum Mist in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\Mist
Browse...
Space required: 129.7MB
Space available: 76.7GB
Cancel
Nullsoft Install System
< Back
Next >

# Practical No: 3

Implement and demonstrate the use of the following in Solidity:

**Practical 3 a) Variables**

**Code:**

```
pragma solidity ^0.8.0;
contract SolidityTest {
    uint storedData; // State variable
    constructor() public {    82425 gas 60200 gas
        storedData = 10;
        }
        function getResult() public view returns(uint){    infinite gas
            uint a = 1; // local variable
            uint b = 2;
            uint result = a + b;
            return result; //access the local variable
            }
            }
```

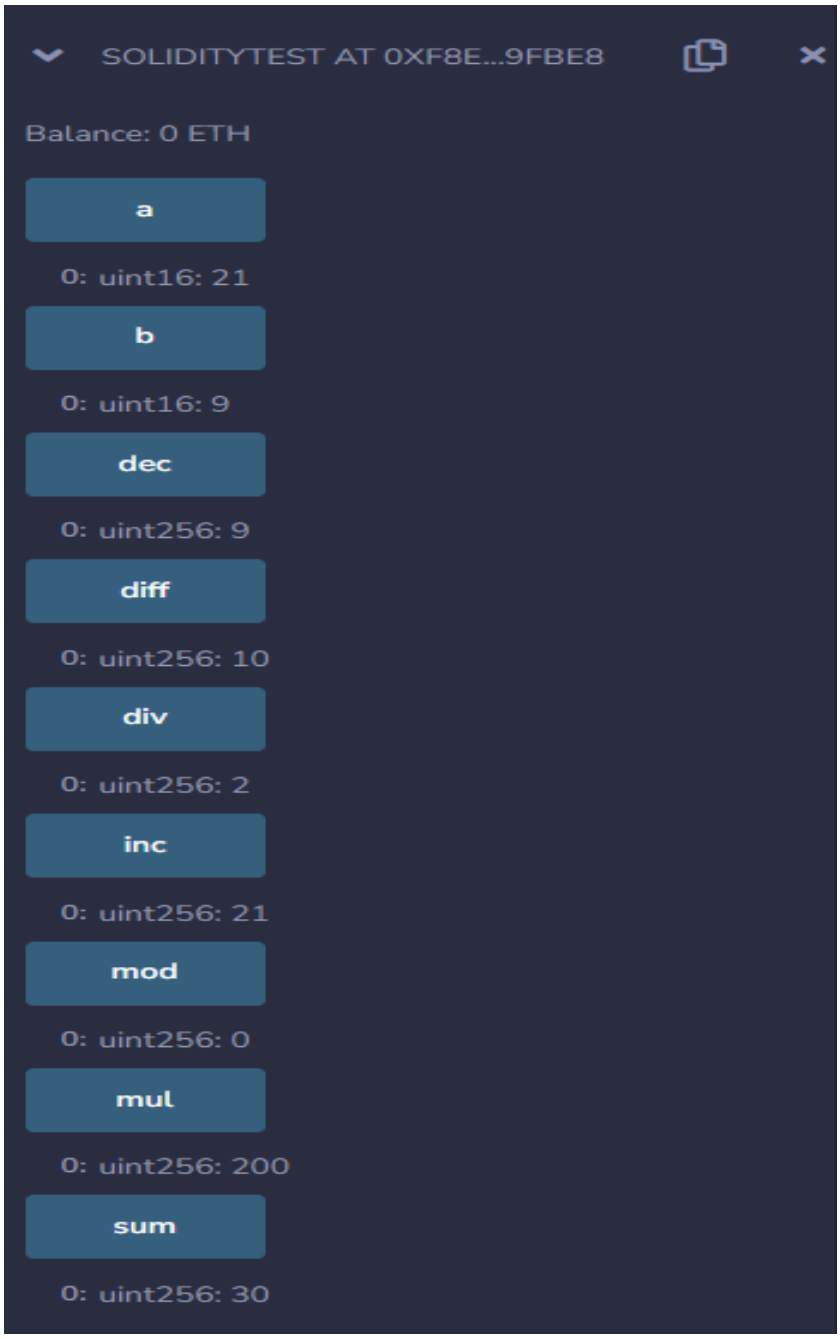
**Output:**

**Practical 3 b) Operators**

**Code:**

```
pragma solidity ^0.8.0;
// Creating a contract
contract SolidityTest {
    // Initializing variables
    uint16 public a = 20;
    uint16 public b = 10;
    // Initializing a variable
    // with sum
    uint public sum = a + b;
    // Initializing a variable
    // with the difference
    uint public diff = a - b;
    // Initializing a variable
    // with product
    uint public mul = a * b;
    // Initializing a variable
    // with quotient
    uint public div = a / b;
    // Initializing a variable
    // with modulus
    uint public mod = a % b;
    // Initializing a variable
    // decrement value
    uint public dec = --b;
    // Initializing a variable
    // with increment value
    uint public inc = ++a;
    }
```

**Output:**

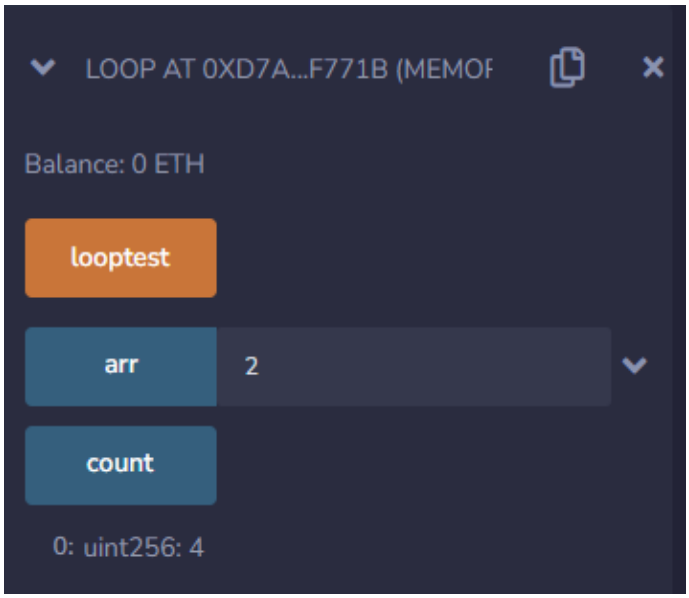

SOLIDITYTEST AT 0XF8E...9FBE8
Balance: 0 ETH
a
0: uint16: 21
b
0: uint16: 9
dec
0: uint256: 9
diff
0: uint256: 10
div
0: uint256: 2
inc
0: uint256: 21
mod
0: uint256: 0
mul
0: uint256: 200
sum
0: uint256: 30

**Practical 3 c) Loops**

**Code:**

```
pragma solidity ^0.8.0;
contract loop{
    uint[4] public arr;
    uint public count;
    function looptest() public{    infinite gas
        while(count<arr.length){
            arr[count] = count;
            count++;
            }
            }
            }
```

**Output:**



## Practical 3 d) Decision Making

### Code:

```
// 'if...else' statement
pragma solidity ^0.8.0;
contract decisionMaking{
    function check(int a) public pure returns(string memory){
        string memory value;
        if(a>0)
        {
            value = "Greater than zero";
            }
            else if(a==0)
            {
                value = "Equal to zero";
                }
                else
                {
                    value = "Less than zero";
                    }
                    return value;
                    }
                    }
```

### Output:

## Practical 3 e) Strings and Enum

### Code:

```
pragma solidity ^ 0.8.0;
// Creating a contract
contract stringtest {
    // Initializing String variable
    string public str = "GeeksforGeeks";
    // Defining an enumerator
    enum my_enum { geeks_, _for, _geeks }
    // Defining a function to return
    // values stored in an enumerator
    function Enum() public pure returns(    432 gas
        my_enum) {
            return my_enum._geeks;
            }
            }
```

### Output:

STRINGTEST AT 0X358...D5EE3 (

Balance: 0 ETH

Enum

0: uint8: 2

str

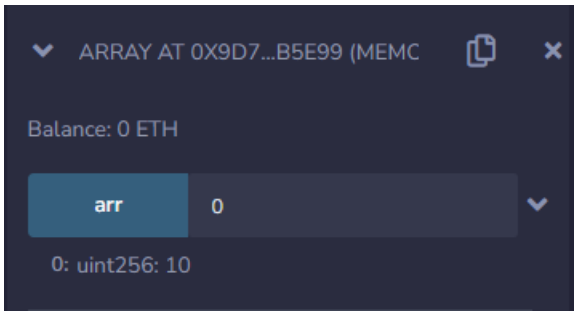
0: string: GeeksforGeeks

## Practical 3 f) Array

**Code:**

```
pragma solidity ^0.8.0;
contract Array{
    uint[4] public arr =[10,20,30,40];
    }
```

**Output:**



## Practical 3 g) Structs

### Code:

```
pragma solidity ^0.8.0;
// Creating a contract
contract structtest {
    struct Book {
        string name;
        string writter;
        uint id;
        bool available;
        }
        Book book1;
        Book book2
        = Book("Building Ethereum DApps",
        "Roberto Infante ",
        2, false);
        function set_book_detail() public {    infinite gas
            book1 = Book("Introducing Ethereum and Solidity",
            "Chris Dannen",
            1, true);
            }
            function book_info(    infinite gas

            )public view returns (
                string memory, string memory, uint, bool) {
                    return(book2.name, book2.writter,
                    book2.id, book2.available);
                    }
                    function get_details(    infinite gas

                    ) public view returns (string memory, uint) {
                        return (book1.name, book1.id);
                        }
                        }
```

### Output:

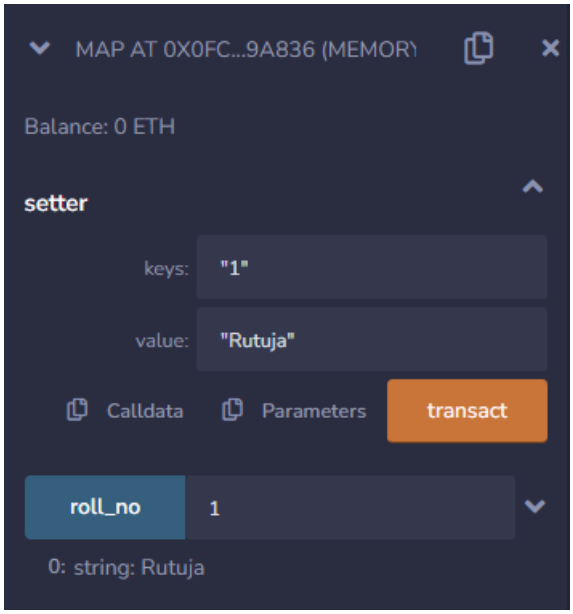
```
STRUCTTEST AT 0XD2A...FD005

Balance: 0 ETH

set_book_deta

book_info

0: string: Building Ethereum DApps
1: string: Roberto Infante
2: uint256: 2
3: bool: false

get_details

0: string: Introducing Ethereum and Solidity
1: uint256: 1
```

## Practical 3 h) Mapping

### Code:

```
pragma solidity ^0.8.0;
contract Map
{
    mapping(uint=>string) public roll_no;
    function setter(uint keys, string memory value) public
    {
        roll_no[keys]= value;
        }
        }
```

### Output:



## Practical 3 i) Conversion

### Code:

```
pragma solidity ^0.8.0;
contract Converse
{
    uint a = 10;
    uint8 b = 20;
    uint16 c = 30;
    function add() public view returns(uint)    infinite gas
    {
        uint d = a + uint(b) + uint(c);
        return d;
        }
        }
```

### Output:

CONVERSE AT 0XCD6...99DF9 (M

Balance: 0 ETH

add

0: uint256: 60

# Practical No: 4

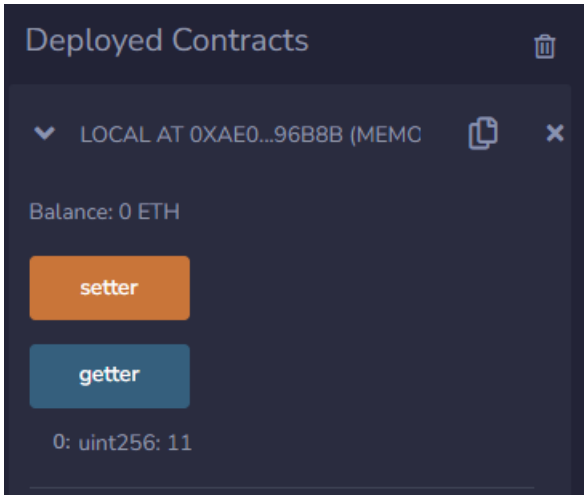
Implement and demonstrate the use of the following in Solidity:

**Practical 4 a) Functions**

**Code:**

```
pragma solidity ^0.8.0;
contract local
{
    uint age = 10;
    function getter() public view returns(uint)    2437 gas
    {
        return age;
        }
        function setter() public    infinite gas
        {
            age = age+1;
            }
            }
```

**Output:**

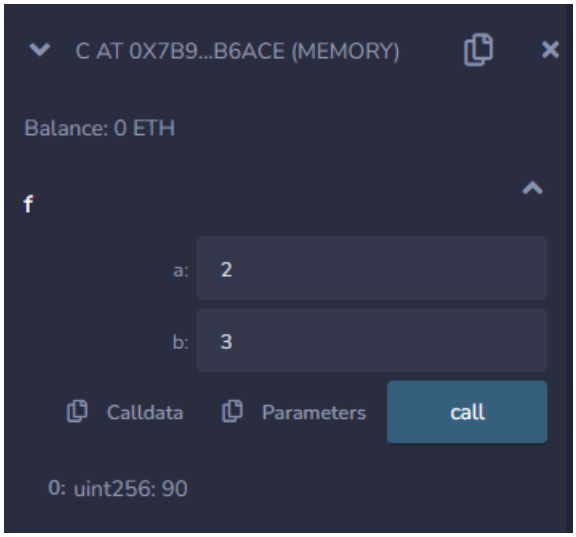


## Practical 4 b) Pure and View Functions

### Code:

```
// Pure
pragma solidity ^0.8.0;
contract C {
    function f(uint a, uint b) public pure returns (uint) {    infinite gas
        return a * (b + 42);
        }
        }
        // View
        contract local
        {
            uint age = 10;
            function getter() public view returns(uint)    2415 gas
            {
                return age;
                }}
```

### Output:

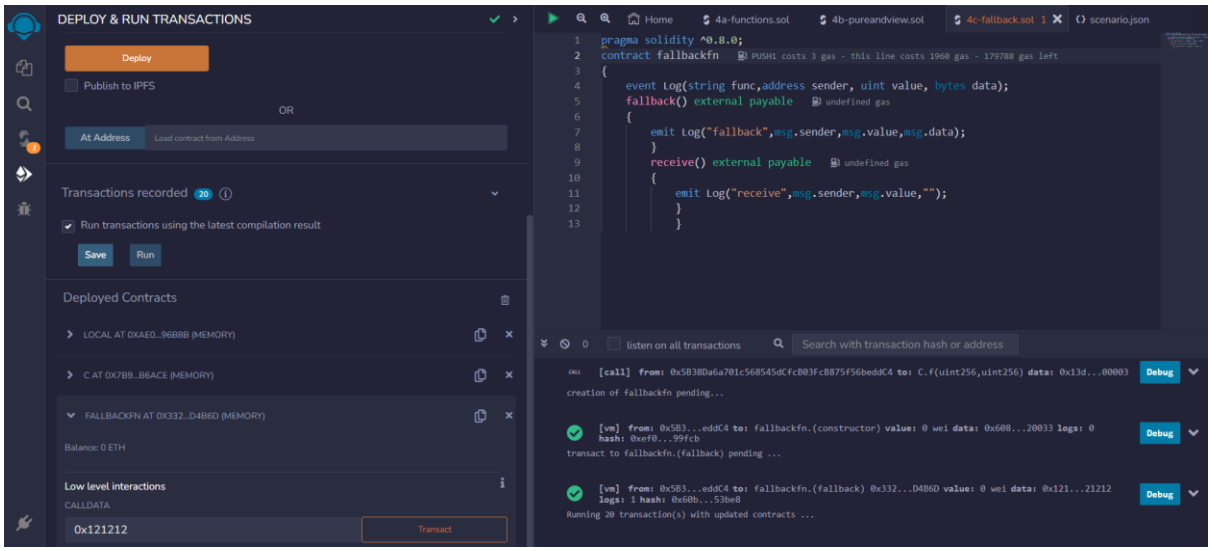


## Practical 4 c) Fallback Functions

### Code:

```
pragma solidity ^0.8.0;
contract fallbackfn
{
    event Log(string func,address sender, uint value, bytes data);
    fallback() external payable
    {
        emit Log("fallback",msg.sender,msg.value,msg.data);
        }
        receive() external payable
        {
            emit Log("receive",msg.sender,msg.value,"");
            }
            }
```

### Output:



## Practical 4 d) Mathematical Operations

### Code:

```
pragma solidity ^0.8.0;
contract C {
    function f() public pure returns (uint256 aMod, uint256 mMod) {
        uint256 x = 3;
        // Old code gen: add/mulmod(5, 4, 3)
        // New code gen: add/mulmod(4, 5, 5)
        aMod = addmod(++x, ++x, x);
        mMod = mulmod(++x, ++x, x);
        }
        }
```

### Output:

C AT 0XE28...4157A (MEMORY)

Balance: 0 ETH

f

0: uint256: aMod 0

1: uint256: mMod 2

## Practical 4 e) Cryptographic Functions

### Code:

```
pragma solidity ^0.8.13;
contract HashFunction {
    function hash(    infinite gas
        string memory _text,
        uint _num,
        address _addr
        ) public pure returns (bytes32) {
            return keccak256(abi.encodePacked(_text, _num, _addr));
            }
            function collision(string memory _text, string memory _anotherText)    infinite gas
            public
            pure
            returns (bytes32)
            {
                // encodePacked(AAA, BBB) -> AAABBB
                // encodePacked(AA, ABBB) -> AAABBB
                return keccak256(abi.encodePacked(_text, _anotherText));
                }
                }
                contract GuessTheMagicWord {
                    bytes32 public answer =
                    0x60298f78cc0b47170ba79c10aa3851d7648bd96f2f8e46a19dbc777c36fb0c00;
                    // Magic word is "Solidity"
                    function guess(string memory _word) public view returns (bool) {    infinite gas
                        return keccak256(abi.encodePacked(_word)) == answer;
                        }
                        }
```

### Output:

```
GUESSTHEMAGICWORD AT 0X1C9...2B4BD (MEMOI

Balance: 0 ETH

answer

0: bytes32: 0x60298f78cc0b47170ba79c10aa3851d7648bd9
            6f2f8e46a19dbc777c36fb0c00

guess

_word:  "Solidity"

Calldata   Parameters   call

0: bool: true
```

## Practical 4 f) Function Modifiers

### Code:

```
pragma solidity ^0.8.13;
contract FunctionModifier {
    // We will use these variables to demonstrate how to use
    // modifiers.
    address public owner;
    uint public x = 10;
    bool public locked;
    constructor() {    384546 gas 337800 gas
        // Set the transaction sender as the owner of the contract.
        owner = msg.sender;
        }
        modifier onlyOwner() {
            require(msg.sender == owner, "Not owner");
            _;
            }
            modifier validAddress(address _addr) {
                require(_addr != address(0), "Not valid address");
                _;
                }
                function changeOwner(address _newOwner) public onlyOwner validAddress(_newOwner) {    26973 gas
                    owner = _newOwner;
                    }
                    modifier noReentrancy() {
                        require(!locked, "No reentrancy");
                        locked = true;
                        _;
                        locked = false;
                        }
                        function decrement(uint i) public noReentrancy {    infinite gas
                            x -= i;
                            if (i > 1) {
                                decrement(i - 1);
                                }
                                }
                                }
```

### Output:

FUNCTIONMODIFIER AT 0X93F...C96

Balance: 0 ETH

changeOwner address _newOwner

decrement uint256 i

locked

0: bool: false

owner

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

x

0: uint256: 10

# Practical No: 5

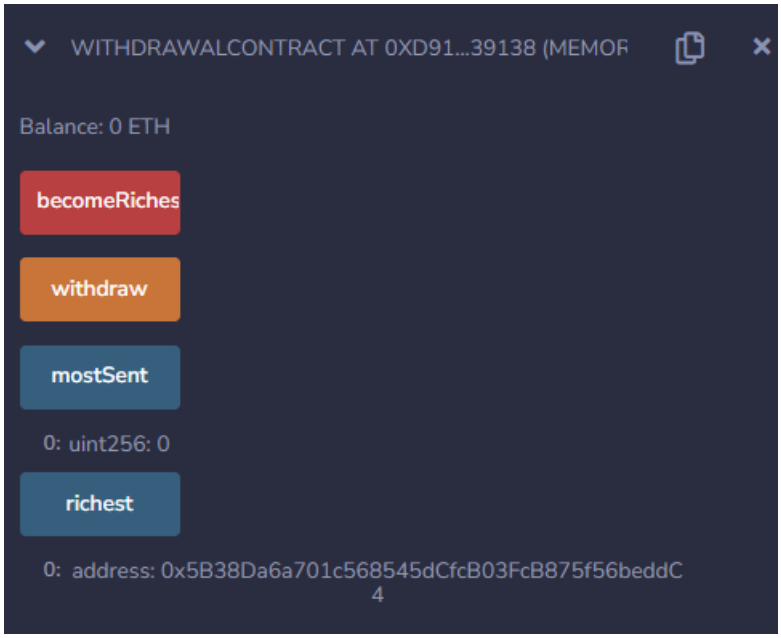
Implement and demonstrate the use of the following in Solidity:

**Practical 5 a) Withdrawal Pattern**

**Code:**

```
pragma solidity ^0.8.0;
contract WithdrawalContract {    PUSH1 costs 3 gas - this line costs 73 gas - 2978936 gas left
    address public richest;
    uint public mostSent;

    mapping (address => uint) pendingWithdrawals;
    error NotEnoughEther();

    constructor() payable {    244193 gas 197600 gas
        richest = msg.sender;
        mostSent = msg.value;
        }
        function becomeRichest() public payable {    infinite gas
            if (msg.value <= mostSent) revert NotEnoughEther();
            pendingWithdrawals[richest] += msg.value;
            richest = msg.sender;
            mostSent = msg.value;
            }
            function withdraw() public {    infinite gas
                uint amount = pendingWithdrawals[msg.sender];
                pendingWithdrawals[msg.sender] = 0;
                payable(msg.sender).transfer(amount);
                }
                }
```

**Output:**



## Practical 5 b) Restricted Access

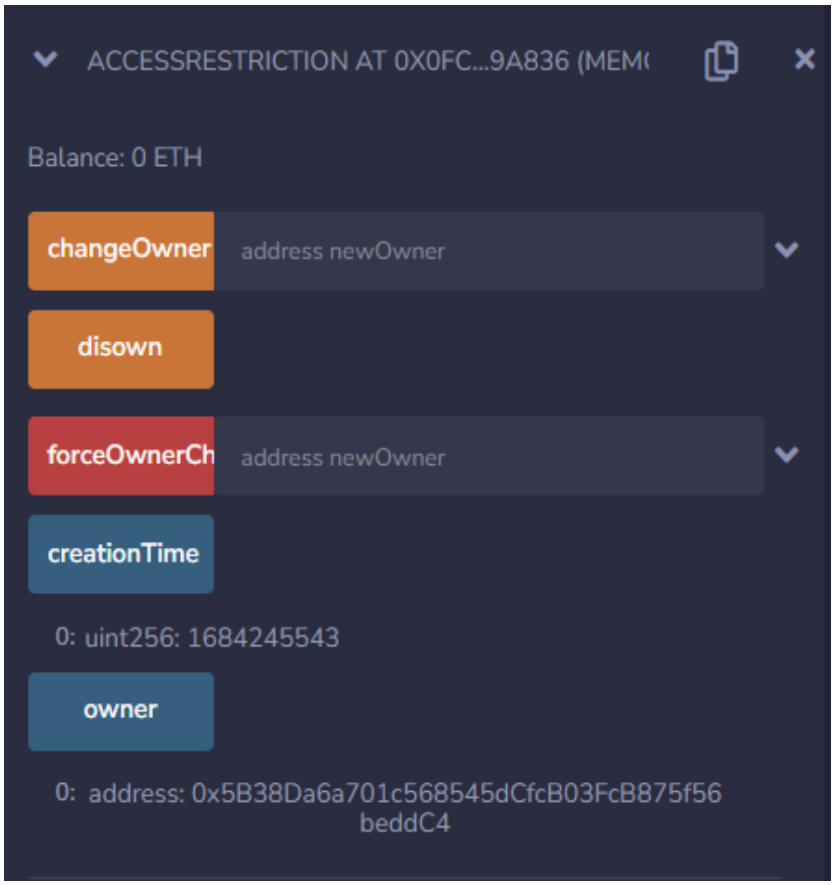
## Code:

Home 5a-withdraw.sol 5b-restricted.sol 1

default_workspace/5a-withdraw.sol

```
pragma solidity ^0.8.4;
contract AccessRestriction {
    address public owner = m
    uint public creationTime = block.timestamp;

    error Unauthorized();
    error TooEarly();
    error NotEnoughEther();

    modifier onlyBy(address account)
    {
        if (msg.sender != account)
        revert Unauthorized();
        _;
        }
        function changeOwner(address newOwner)      26938 gas
        public
        onlyBy(owner)
        {
            owner = newOwner;
            }
            modifier onlyAfter(uint time) {
                if (block.timestamp < time)
                revert TooEarly();
                _;
                }
                function disown()      infinite gas
                public
                onlyBy(owner)
                onlyAfter(creationTime + 6 weeks)
                {
                    delete owner;
                    }
                    modifier costs(uint amount) {
                        if (msg.value < amount)
                        revert NotEnoughEther();
                        _;
                        if (msg.value > amount)
                        payable(msg.sender).transfer(msg.value - amount);}
                        function forceOwnerChange(address newOwner)      infinite gas
                        public
                        payable
                        costs(200 ether)
                        {
                            owner = newOwner;
                            // just some example condition
                            if (uint160(owner) & 0 == 1)
                            // This did not refund for Solidity
                            // before version 0.4.0.
                            return;
                            // refund overpaid fees
                            }
                            }
```

**Output:**


ACCESSRESTRICTION AT 0X0FC...9A836 (MEM(
Balance: 0 ETH
changeOwner
address newOwner
disown
forceOwnerCh
address newOwner
creationTime
0: uint256: 1684245543
owner
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56
beddC4

# Practical No: 6

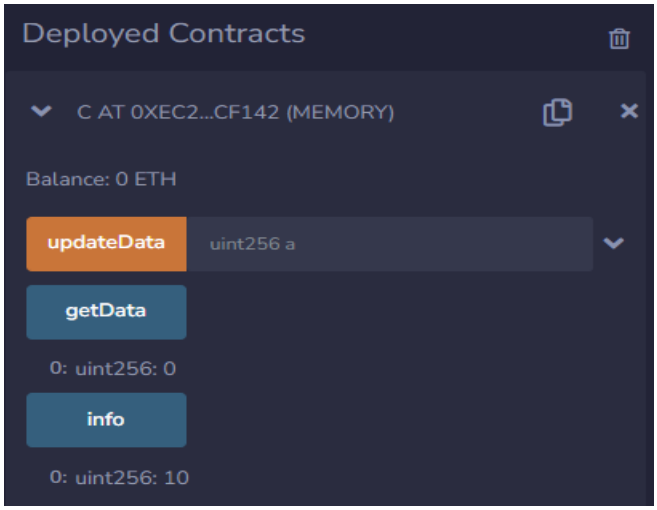
Implement and demonstrate the use of the following in Solidity:

## Practical 6 a) Contracts

### Code:

```
pragma solidity ^0.8.0;
contract C {
    //private state variable
    uint private data;
    //public state variable
    uint public info;
    //constructor
    constructor() public {    98837 gas 76600 gas
        info = 10;
    }
        //private function
        function increment(uint a) private pure returns(uint) { return a + 1; }    infinite gas
        //public function
        function updateData(uint a) public { data = a; }    22498 gas
        function getData() public view returns(uint) { return data; }    2459 gas
        function compute(uint a, uint b) internal pure returns (uint) { return a + b; }    infinite gas
        }
        //External Contract
        contract D {
            function readData() public returns(uint) {    infinite gas
                C c = new C();
                c.updateData(7);
                return c.getData();
                }
                }
                //Derived Contract
                contract E is C {
                    uint private result;
                    C private c;
                    constructor() public {    infinite gas 119200 gas
                        c = new C();
                    }
                        function getComputedResult() public {    infinite gas
                            result = compute(3, 5);
                            }
                            function getResult() public view returns(uint) { return result; }    2503 gas
                            }
```

### Output:

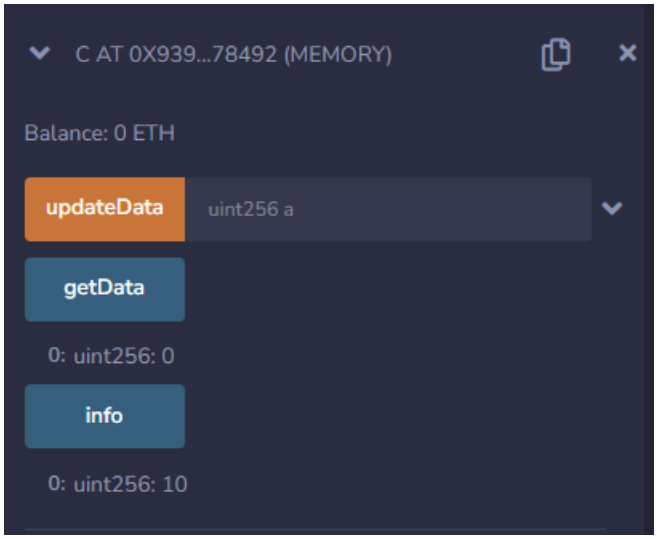


## Practical 6 b) Inheritance

### Code:

```
pragma solidity ^0.8.0;
contract C {
    //private state variable
    uint private data;
    //public state variable
    uint public info;
    //constructor
    constructor() public {    98837 gas 76600 gas
        info = 10;
    }
        //private function
        function increment(uint a) private pure returns(uint) { return a + 1; }    infinite gas
        //public function
        function updateData(uint a) public { data = a; }    22498 gas
        function getData() public view returns(uint) { return data; }    2459 gas
        function compute(uint a, uint b) internal pure returns (uint) { return a + b; }    infinite gas
        }
        //Derived Contract
        contract E is C {
            uint private result;
            C private c;
            constructor() public {    infinite gas 119200 gas
                c = new C();
            }
                function getComputedResult() public {    infinite gas
                    result = compute(3, 5);
                    }
                    function getResult() public view returns(uint) { return result; }    2503 gas
                    }
```

### Output:



## Practical 6 c) Constructors

### Code:

```
// creating a constructor
pragma solidity ^0.8.0;
// Creating a contract
contract constructorExample {
    // Declaring state variable
    string str;
    // Creating a constructor
    // to set value of 'str'
    constructor() public {      infinite gas 110400 gas
        str = "GeeksForGeeks";
    }
        // Defining function to
        // return the value of 'str'
        function getValue(      infinite gas
        ) public view returns (
            string memory) {
                return str;
                }
                }
```

### Output:

```
CONSTRUCTOREXAMPLE AT 0XB57...9DF30

Balance: 0 ETH

getValue

0: string: GeeksForGeeks
```

## Practical 6 d) Abstract Contracts

**Code:**

```
pragma solidity ^0.8.0;
contract Calculator {
    constructor() public{    71123 gas 71000 gas
    }
    function getResult() public view returns(uint){    infinite gas
        uint a = 1;
        uint b = 2;
        uint result = a + b;
        return result;
    }
}
```

**Output:**

CALCULATOR AT 0XD91...39138

Balance: 0 ETH

getResult

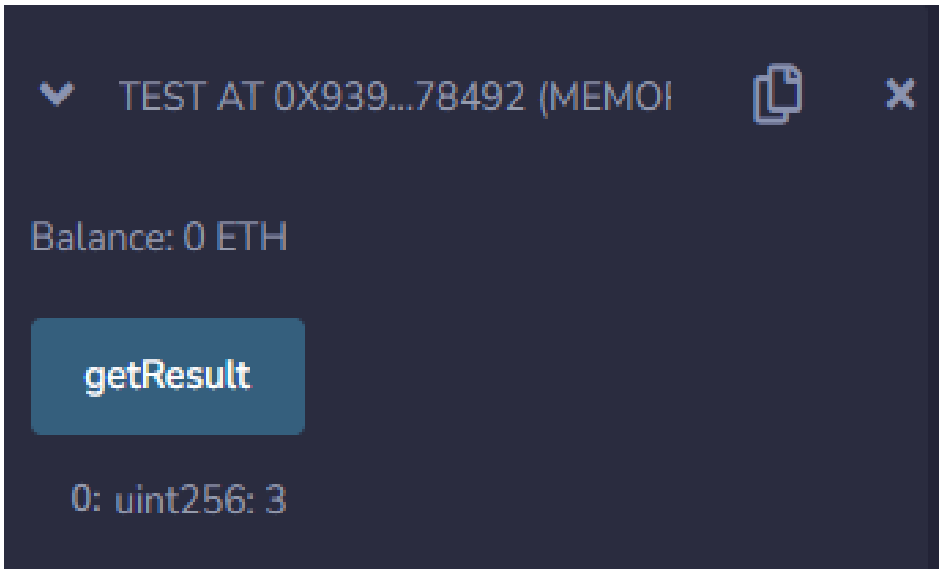
0: uint256: 3

## Practical 6 e) Interfaces

**Code:**

```
pragma solidity ^0.8.0;

interface Calculator {
    function getResult() external view returns(uint);
}
contract Test is Calculator {
    constructor() public {}
    function getResult() external view returns(uint){
        uint a = 1;
        uint b = 2;
        uint result = a + b;
        return result;
    }
}
```

**Output:**



**Practical No: 7**

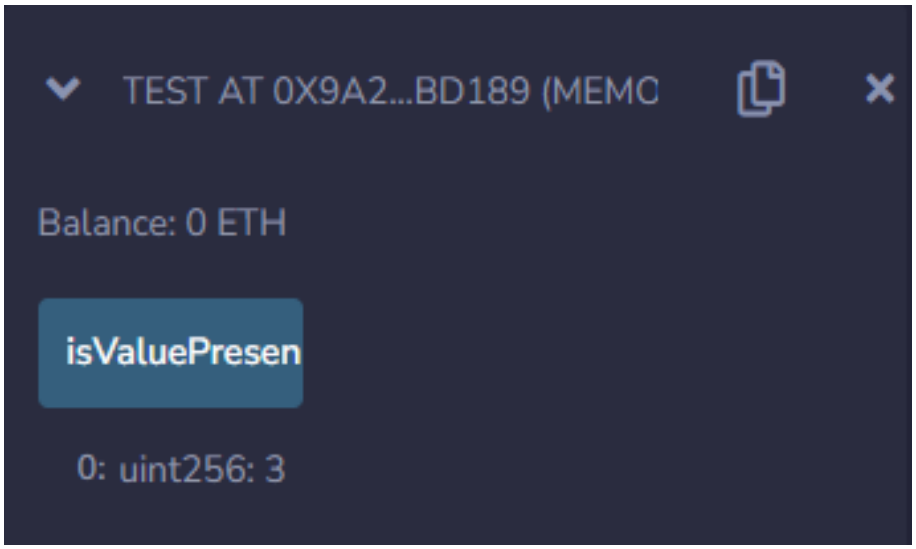
Implement and demonstrate the use of the following in Solidity:

**Practical 7 a) Libraries**

**Code:**

```
pragma solidity ^0.8.0;

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 {
    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;

        //search if value is present in the array using Library function
        uint index = Search.indexOf(data, value);
        return index;
    }
}
```

**Output:**

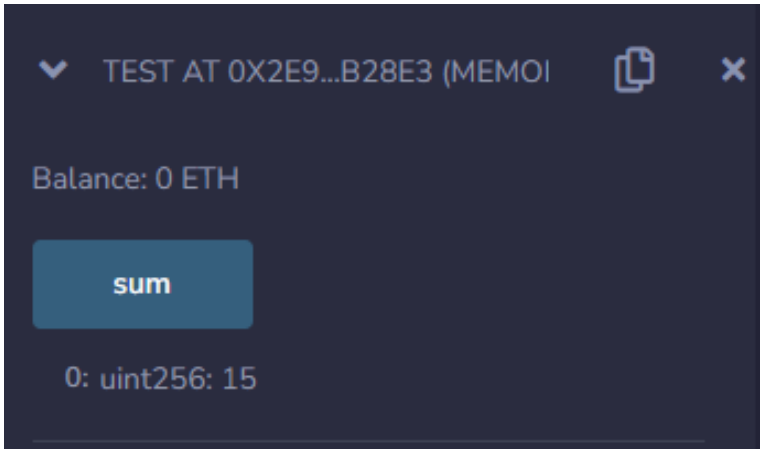


## Practical 7 b) Assembly

### Code:

```
pragma solidity ^0.8.0;
library Sum {
    function sumUsingInlineAssembly(uint[] memory _data) public pure returns (uint o_sum) {    infinite gas
        for (uint i = 0; i < _data.length; ++i) {
            assembly {
                o_sum := add(o_sum, mload(add(add(_data, 0x20), mul(i, 0x20))))
            }
        }
    }
}
contract Test {
    uint[] data;
    constructor() public {    374305 gas 142000 gas
        data.push(1);
        data.push(2);
        data.push(3);
        data.push(4);
        data.push(5);
    }
    function sum() external view returns(uint){    infinite gas
        return Sum.sumUsingInlineAssembly(data);
    }
}
```

### Output:

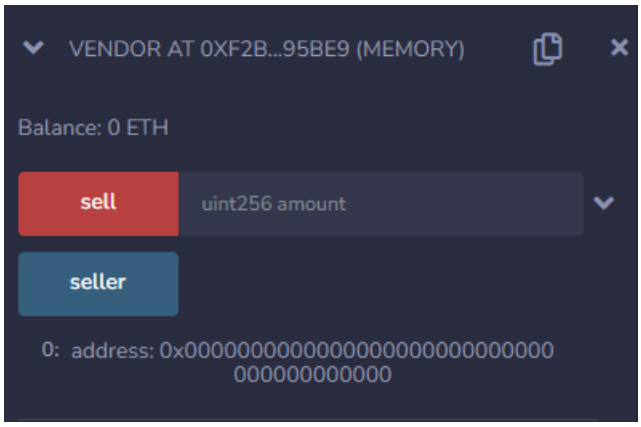


## Practical 7 c) Error Handling

### Code:

```
pragma solidity ^0.8.0;
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:



**Practical No: 8**

Demonstrate the use of Bitcoin Core API.

**Code:**

BC_Practical 8.py - C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 8.py (3.11.2)

File Edit Format Run Options Window Help

```
from bitcoinlib.wallets import Wallet
w = Wallet.create('Wallet3')
key1 = w.get_key()
print(key1.address)

# Send a small transaction to your wallet and use the scan() method to update transactions and UTXO's
w.scan()
print(w.info())
```

**Output:**

Command Prompt

```
C:\Users\Cvman\AppData\Local\Programs\Python\Python311>pip install bitcoinlib
Requirement already satisfied: bitcoinlib in c:\users\cvman\appdata\local\programs\python\python311\lib\site-packages
.6.9)
Requirement already satisfied: SQLAlchemy>=1.4.28 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-p
kages (from bitcoinlib) (2.0.13)
Requirement already satisfied: pycryptodome>=3.14.1 in c:\users\cvman\appdata\local\programs\python\python311\lib\site
ackages (from bitcoinlib) (3.17)
Requirement already satisfied: requests>=2.25.0 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-pac
ges (from bitcoinlib) (2.30.0)
Requirement already satisfied: ecdsa>=0.17 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-packages
from bitcoinlib) (0.18.0)
Requirement already satisfied: numpy>=1.21.0 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-packag
 (from bitcoinlib) (1.24.3)
Requirement already satisfied: six>=1.9.0 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-packages
rom ecdsa>=0.17->bitcoinlib) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\cvman\appdata\local\programs\python\python311\lib\
te-packages (from requests>=2.25.0->bitcoinlib) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-package
(from requests>=2.25.0->bitcoinlib) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-p
kages (from requests>=2.25.0->bitcoinlib) (2.0.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-p
kages (from requests>=2.25.0->bitcoinlib) (2023.5.7)
Requirement already satisfied: typing-extensions>=4.2.0 in c:\users\cvman\appdata\local\programs\python\python311\lib\
te-packages (from SQLAlchemy>=1.4.28->bitcoinlib) (4.5.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\cvman\appdata\local\programs\python\python311\lib\site-pac
```

```
IDLE Shell 3.11.2
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Cvman\AppData\Local\Programs\Python\Python311\BC_Practical 8.py
1FRv8FL8B7BM1uJMQUkBP8FtueSXhWgEZe
=== WALLET ===
 ID                          1
 Name                        Wallet3
 Owner
 Scheme                      bip32
 Multisig                    False
 Witness type                legacy
 Main network                bitcoin
 Latest update               2023-05-13 14:07:23.273095

= Wallet Master Key =
 ID                          1
 Private                     True
 Depth                       0

- NETWORK: bitcoin -
- - Keys
    6 m/44'/0'/0'/0/0          1FRv8FL8B7BM1uJMQUkBP8FtueSXhWgEZe        address index 0                    0.00000000 ฿
    7 m/44'/0'/0'/0/1          1Jks8TjVKJ3KYYViybnfmjKmacdVxdCNHi        address index 1                    0.00000000 ฿
    8 m/44'/0'/0'/0/2          1GTSDyP58fU5yTBJGK4AD85xLo31zA6Q8V        address index 2                    0.00000000 ฿
    9 m/44'/0'/0'/0/3          14K5qxAR2phNQKp8WhuNe22172VKDiorNR        address index 3                    0.00000000 ฿
   10 m/44'/0'/0'/0/4          1H5HgfNMChmH838ZNYrEPnbKMr5VZF4n9N        address index 4                    0.00000000 ฿
   12 m/44'/0'/0'/1/0          1Ly2w7wUXivgZp9LmY8S2Auk6PSXKDEwqn        address index 0                    0.00000000 ฿
   13 m/44'/0'/0'/1/1          1JHuxg8jrDAdZbUSpawKcPkSRPZFPZckV6        address index 1                    0.00000000 ฿
   14 m/44'/0'/0'/1/2          1PFBQC1En2iQFbEYcCnQff6P2KTEcArcW6        address index 2                    0.00000000 ฿
   15 m/44'/0'/0'/1/3          1HLYDgjXgBpKkX1F1Si6bQfPTx1sW1M585        address index 3                    0.00000000 ฿
   16 m/44'/0'/0'/1/4          1G4Ai7jkz4zvzq7x3LFxaJ6wtSTDRczNZY        address index 4                    0.00000000 ฿

- - Transactions Account 0 (0)

= Balance Totals (includes unconfirmed) =


None
```