INDEX

Sr. No	Topic		Date	Remark
1	**Write the following programs for Blockchain in Python:** i. A Simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it. ii. A Simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it. iii. A transaction class to send and receive money and test it.			
2	**Write the following programs for Blockchain in Python:** i. Create multiple transactions and display them. ii. Create a blockchain, a genesis block and execute it.			
3	**Write the following programs for Blockchain in Python:** i. Create a mining function and test it. ii. Add blocks to the miner and dump the blockchain.			
4	**Implement and demonstrate the user of the following in Solidity:** i. Variable ii. Operations iii. Loops iv. Decision Making v. Strings			
5	**Implement and demonstrate the user of the following in Solidity:** i. Arrays ii. Enums iii. Structs iv. Mappings v. Conversations vi. Ether Units vii. Special Variables			

6	**Implement and demonstrate the user of the following in Solidity:** i. Functions ii. View Functions iii. Pure Functions iv. Fallback Functions v. Function Overloading vi. Mathematical Functions vii. Cryptographic Functions			
7	**Implement and demonstrate the user of the following in Solidity:** i. Contracts ii. Inheritance iii. Constructors iv. Abstract Class v. Interfaces			
8	**Implement and demonstrate the user of the following in Solidity:** i. Libraries ii. Assembly iii. Events iv. Error Handling			
9	**Mist Browser Installation.**			

PRACTICAL No: 1 (A(I))

AIM: Create a simple client class that generates the private and public keys by using the built-in python RSA algorithm and test it.

CODE:

```
!pip install crypto
!pip install pycryptodome
from Crypto.PublicKey import RSA
key=RSA.generate(1024)
p_key=key.public_key().export_key("PEM")
priv_key=key.export_key("PEM")
print("Kinjal Jaiswal \n")
print(p_key)
print(priv_key)
```

OUTPUT:

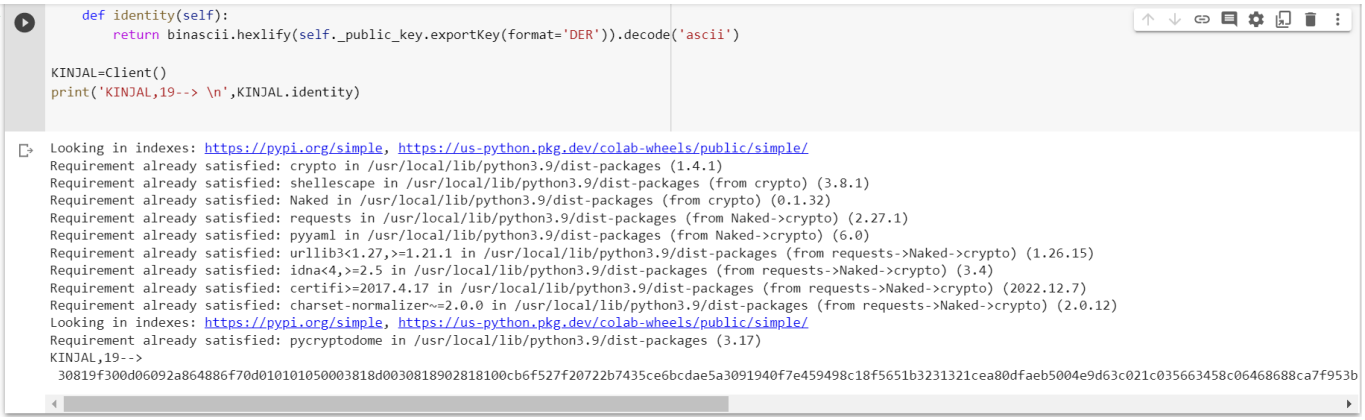
PRACTICAL No: 1 (A(II))

AIM: Create a simple client class that generates the private and public keys by using the built-in python RSA algorithm and test it.

CODE:

```
!pip install crypto
!pip install pycryptodome
#import random
from Crypto.PublicKey import RSA
from Crypto import Random
import binascii
from Crypto.Cipher import PKCS1_v1_5

class Client:
    def __init__(self):
        random=Random.new().read
        self._private_key=RSA.generate(1024,random) #1024->key size
        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')

KINJAL=Client()
print('KINJAL,19--> \n',KINJAL.identity)
```

OUTPUT:



PRACTICAL No: 1 (B)

AIM: A transaction class to send and receive money and test it.

Genesis → 1st block in blockchain

CODE:

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

class Client:
    def __init__(self):
        random=Random.new().read
        self._private_key=RSA.generate(1024,random) #1024->key size
        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,receiver,value):
        self.sender=sender
        self.receiver=receiver
        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,
```

```
            "receiver":self.receiver,
            "value":self.value,
            "time":self.time
        })
    def sign_tran(self):
        private_key=self.sender._private_key
        signer=PKCS1_v1_5.new(private_key)
        h=SHA.new(str(self.to_dict).encode('utf-8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')

def display_tran(transaction):
    dict=transaction.to_dict()
    print('\nsender,Kinjal--> \n'+dict['sender'])
    print('\nreceiver,Ravi--> \n'+dict['receiver'])
    print('\nvalue--> \n'+str(dict['value']))
    print('\ntime--> \n'+str(dict['time']))

transactions=[]

Kinjal=Client()
Ravi= Client()

t1=Transaction(
Kinjal,
Ravi.identity,
15)

t1.sign_tran()
display_tran(t1)
```

OUTPUT:

```
Requirement already satisfied: pyyaml in /usr/local/lib/python3.9/dist-packages (from Naked->crypto) (6.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (1.26.15)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.9/dist-packages (3.17)

sender,Kinjal-->
30819f300d06092a864886f70d01010105000381 8d0030818902818100e01f5bf1ba082950122bf40972be1b97479fe18a8a02a048427907ba5387be3c8f57f2dc2200e66aba222b2bca6503c0900b667e5dfcc0

receiver,Ravi-->
30819f300d06092a864886f70d01010105000381 8d0030818902818100ce189be1ce60ad1057edccd101895486b400171ecdf9d1fe6e731148646a6bd6dceb5328c121143d323ae62899fb46169205d017f79f63

value-->
15

time-->
2023-04-19 07:27:38.860306
```

PRACTICAL No: 2 (A)

AIM: Create multiple transactions and display them.

CODE:

```
!pip install crypto
!pip install pycryptodome
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
import datetime
import binascii
from collections import OrderedDict
import collections
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
class Client:
    def __init__(self):
        random = 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, recipent, value):
        self.sender = sender
        self.recipent = recipent
        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,
            'recipent': self.recipent,
            'value': self.value,
            'time': self.time
        })
    def sign_tran(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')
def display_transaction(transaction):
    # for transaction in transactions:
    dict = transaction.to_dict()
    print("sender:" + dict['sender'])
    print('-----')
    print("recipent:" + dict['recipent'])
    print('-----')
    print("value:" + str(dict['value']))
    print('-----')
    print("time:" + str(dict['time']))
    print('-----')
transactions = []
Kinjal = Client()
Nainu = Client()
Ravi = Client()
t1 = Transaction(
    Kinjal,
    Nainu.identity,
    15.0
)
t1.sign_tran()
transactions.append(t1)
t2 = Transaction(
    Nainu,
    Ravi.identity,
    17.0
)
t2.sign_tran()
transactions.append(t2)
t3 = Transaction(
    Ravi,
    Nainu.identity,
    10.0
)
t3.sign_tran()
transactions.append(t3)
tn = 1
for t in transactions:
    print("Transaction: ", tn)
    display_transaction(t)
    tn = tn + 1
    print('--------------------')
```

OUTPUT:

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: crypto in /usr/local/lib/python3.9/dist-packages (1.4.1)
Requirement already satisfied: Naked in /usr/local/lib/python3.9/dist-packages (from crypto) (0.1.32)
Requirement already satisfied: shellescape in /usr/local/lib/python3.9/dist-packages (from crypto) (3.8.1)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from Naked->crypto) (2.27.1)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.9/dist-packages (from Naked->crypto) (6.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (3.4)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.9/dist-packages (3.17)
Transaction:  1
sender:30819f300d06092a864886f70d01010105000381 8d00308189028181 0089ea3c29502c75bae4c5267712a0e415391623b1f1e8f76331196a893a0f16fde7d448c7a5ef294976304f3a2e4b8f9ff077f56c4d4a99288cc135278d74f7
-----
recipient:30819f300d06092a864886f70d01010105000381 8d00308189028181 009ed63ad818c09121243f0fc72762d7e9e2982cef3af9252429ca7c6fa647c55922d4dae8009b49ed1ef425d0605c46de716ee89e0b572612a3a7e466e786
-----
value:15.0
-----
time:2023-04-19 07:35:12.259064
-----



Transaction:  2
sender:30819f300d06092a864886f70d01010105000381 8d00308189028181 009ed63ad818c09121243f0fc72762d7e9e2982cef3af9252429ca7c6fa647c55922d4dae8009b49ed1ef425d0605c46de716ee89e0b572612a3a7e466e78653
-----
recipient:30819f300d06092a864886f70d01010105000381 8d00308189028181 00d589b2fdebb41e9985b13c3274efee1c28705491ab53899ca9545b85eedb4830dde24712e26adde8eca7523aed3492c6853183ab18fe780087b6de4cf89a
-----
value:17.0
-----
time:2023-04-19 07:35:12.262257



--------------------
Transaction:  3
sender:30819f300d06092a864886f70d01010105000381 8d00308189028181 00d589b2fdebb41e9985b13c3274efee1c28705491ab53899ca9545b85eedb4830dde24712e26adde8eca7523aed3492c6853183ab18fe780087b6de4cf89aed1
-----
recipient:30819f300d06092a864886f70d01010105000381 8d00308189028181 009ed63ad818c09121243f0fc72762d7e9e2982cef3af9252429ca7c6fa647c55922d4dae8009b49ed1ef425d0605c46de716ee89e0b572612a3a7e466e7865
-----
value:10.0
-----
time:2023-04-19 07:35:12.269547
-----
```

PRACTICAL No: 2 (B)

AIM: Create a block chain of Genesis block and execute it.

Noance: a randomly generated number (unique) used once in cryptography transaction.

CODE:

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


class Client:
    def __init__(self):
        random = 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, recipent, value):
        self.sender = sender
        self.recipent = recipent
        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,
            'recipent': self.recipent,
            'value': self.value,
            'time': self.time
        })

    def sign_tran(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')


def display_transaction(transaction):
    # for transaction in transactions:
    dict = transaction.to_dict()
    print("sender:" + dict['sender'])
    print('-----')
    print("recipent:" + dict['recipent'])
    print('-----')
    print("value:" + str(dict['value']))
    print('-----')
    print("time:" + str(dict['time']))
    print('-----')

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_transaction:
            display_transaction(transaction)
            print(".............")
            print("====================")

class Block:
    def __init__(self):
        self.verified_transaction=[]
        self.previous_block_hash=""
        self.Nonce=""
Kinjal = Client()
t0=Transaction(
    "Genesis",
    Kinjal.identity,
    500.0
)
block0=Block()
block0.previous_block_hash=None
Nonce=None
```

```
block0.verified_transaction.append(t0)
digest=hash(block0)
last_block_hash = digest
TPCoins=[]
TPCoins.append(block0)
dump_blockchain(TPCoins)
```

OUTPUT:

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: crypto in /usr/local/lib/python3.9/dist-packages (1.4.1)
Requirement already satisfied: shellescape in /usr/local/lib/python3.9/dist-packages (from crypto) (3.8.1)
Requirement already satisfied: Naked in /usr/local/lib/python3.9/dist-packages (from crypto) (0.1.32)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.9/dist-packages (from Naked->crypto) (6.0)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from Naked->crypto) (2.27.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (1.26.15)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (3.4)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->Naked->crypto) (2.0.12)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.9/dist-packages (3.17)
Kinjal jaiswal, Roll no: 19
Number of blocks in the chain:1
block#0
sender:Genesis
-----
recipent:30819f300d06092a864886f70d01010105000381d003081890281810 0bf6792f711fe0893cf7d1dbe4b0701a49a45e99772ad5f1c48dda2a9173be2a31ba3754cad605edcb1
-----
value:500.0
-----
time:2023-04-19 09:32:13.932887
-----
.............
====================
```

PRACTICAL No: 3 (A)

AIM: Create a mining function and test it.

Miners: verifies the transactions in block chain

CODE:

```
!pip install crypto
!pip install pycryptodome
import hashlib
def sha256(message):
    return hashlib.sha256(message.encode('ascii')).hexdigest()
def mine(message,difficulty=1):
    assert difficulty>=1 #debugging
    prefix= '1'* difficulty #verify diffficulty
    print ("prefix",prefix)
    for i in range(1000):
        digest = sha256(str(hash(message)) + str(i))
        print("Testing --> " + digest)
        if digest.startswith(prefix):
            print("After" + str(i) + "iterations found nounce" + digest)
            return i
mine("Kinjal", 3)
```

OUTPUT:

```
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.9/dist-packages (3.17)
Kinjal jaiswal, Roll no: 19
prefix 111
Testing --> 5d3acf3bfd2ab1cdf3948d7dbd8a5729fa25e04a75f9ca11cdbcb30e87dbedfd
Testing --> a81a77e29dad624d11fb08173d6a050f7ce86fec2b9eb07d71b61fbe0bb16692
Testing --> 7257d94d271e1efe0b9475a2e40a6292aa3bb0eb8cec14c4ab4cebc3cd53489e
Testing --> 27add7b3a5826eae668cd31621c4c6526b0874f7f3146b8e9ce6bb46d5f9fdfb
Testing --> 10a8c3f32dc94d13dcce09d0b77a66c11d3ba3c533078e9001f03e58318e0175
Testing --> f00d3138fae2ce3ef625ac6922719141ff011f48c1fa4b9402dd5df6d86bc472
Testing --> fa99d24bec3527259a15f1da4d844707b33a2617b982b190eb251c0e2269a920
Testing --> 17c028cabcacb3a33b0212598cad40d7ceb574b3fd77ba205d5e6b261981859f
Testing --> 676f62c095162ec7b902e8aba0fbcb6c3fa85570ba1c0d82c7a3796a0e4f8e86
Testing --> 72bef226ce7021da694b4b7accb80e529f505aa80015597caf938e73998819d6
Testing --> bdce3a772035196bc3e1524e36e7d0e952e97c04b9de99d5b824e2164da05d29
Testing --> 6b95cef80849cbbdc75e50b39da83dca97c30e37bfe8470ed15fb967768a94ec
Testing --> f2fe014660f88683e9c7f2292e1d168d886e1fbc3c1f947907eae390e747229f
Testing --> 82a7ab0febf4f9d724333926ac38dbaa2dbc7023f182acef82b4bbcbca07c7c3
Testing --> 3a71dcb5400d45d99cbf26ed53c7a6083bcb1ff51ff02f8222671dba558ca9310
Testing --> d0df19ccf65de53065cc2bf52bf6525be9cc94a999f8152ca47aa3742e56def0
Testing --> 3f543dbe4fd42a66540b4a85b2fb03caff5ccb916528069c9e706b885e4c6d3e
Testing --> 60edefe8b43ab8cf87f3953091e799747024fa8c55b8d92b43f7710b92bfe490
Testing --> e6058dc68ae0ea5f2c8e297c7f169a860a36b397ac3ef6243e3bcc86cc6f0bea
Testing --> 00833f7dbc8d11fa4de01ec839e420005525da3dddefedb73632134251c26ae4
Testing --> 9ff6ad042c2ca10bfe5dcd7611096f9f7860e23f783023d54860a0130ba32688
Testing --> 81dfde1d085fff38d815e0d8753eb6b2e6f66f553c4aea6d531385097d4d3161
Testing --> 1e233619cea51a6ec38534f784d77147b24194874356b13b84d1a7d6328aab65
Testing --> 24495a294c7db88ecfb1097013aaab56410d80d805f99bc81fe96daa0d8fd464
Testing --> 31ef3c63d6ca83403e7cb09e2f36c7fdb5370d9864f6f468ffec9d7eebb16242
Testing --> 2208cc616116e35fc896af8689b33a3cd2af667cdd4f146c47aa71110506326e
Testing --> dc1843ff2c0ac114fce16dceec062611f4cd5d5df56bdd8b08d73b4c407ceb3a
Testing --> 0f7509dbde0d0c5d4236e46b075db92174392998cf494ff7b4120201bb3a39ec
Testing --> e96ce1ed1845a8be87433ad69e8b035b921d10ec51d7d5cf2700d8e48e5d8195
Testing --> 658073a8bf5a5c01786e53ed6412c5fbdd7e3e0f03e9981cbf62a04ced457a4a
Testing --> 60cefdd7d8220b57b031f2eacfca9d4bd653b10a9b6120527c054cd93dc80675
Testing --> 111ada1bbcd7a012577cafb959c754bdeaab9916e6ef4a073a9dc352af90ee53
After864iterations found nounce111ada1bbcd7a012577cafb959c754bdeaab9916e6ef4a073a9dc352af90ee53
864
```

PRACTICAL No: 3 (B)

AIM: Add block to miner and dump the block chain.

Miners: verifies the transactions in block chain

CODE:

```
!pip install crypto
!pip install pycryptodome

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
import datetime
import binascii
from collections import OrderedDict
import collections
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
import hashlib

print("Kinjal Jaiswal, Roll no: 19")
class Client:
    def __init__(self):
        random = 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, recipent, value):
        self.sender = sender
        self.recipent = recipent
        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,
            'recipent': self.recipent,
            'value': self.value,
            'time': self.time
        })

    def sign_tran(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')


def display_transaction(transaction):
    # for transaction in transactions:
    dict = transaction.to_dict()
    print("sender:" + dict['sender'])
    print('-----')
    print("recipent:" + dict['recipent'])
    print('-----')
    print("value:" + str(dict['value']))
    print('-----')
    print("time:" + str(dict['time']))
    print('-----')

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_transaction:
            display_transaction(transaction)
            print(".............")
            print("====================")

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

def sha256(message):
    return hashlib.sha256(message.encode('ascii')).hexdigest()

def mine(message,difficulty=1):
    assert difficulty>=1 #debugging
    prefix= '1'* difficulty #verify diffficulty
```

```
    print ("prefix",prefix)
    for i in range(1000):
        digest = sha256(str(hash(message)) + str(i))
        print("Testing --> " + digest)
        if digest.startswith(prefix):
            print("After " + str(i) + "iterations found nounce " + digest)
            return i
mine("Kinjal", 3)

transactions = []

Kinjal = Client()
Nainu = Client()
Ravi = Client()

t0=Transaction(
    "Genesis",
    Kinjal.identity,
    500.0
)

t1 = Transaction(
    Kinjal,
    Nainu.identity,
    15.0
)

t1.sign_tran()
transactions.append(t1)

t2 = Transaction(
    Nainu,
    Ravi.identity,
    17.0
)

t2.sign_tran()
transactions.append(t2)

t3 = Transaction(
    Ravi,
    Nainu.identity,
    10.0
)

#blockchain
TPCoins=[]
```

```
block0=Block()
block0.previous_block_hash=None
Nonce=None
block0.verified_transaction.append(t0)
digest=hash(block0)
last_block_hash = digest
last_block_hash=digest
TPCoins.append(block0)


block1=Block()
block1.previous_block_hash=last_block_hash
block1.verified_transaction.append(t1)
block1.verified_transaction.append(t2)
block1.Nonce=mine(block1,2)
digest=hash(block1)
last_block_hash=digest
TPCoins.append(block1)

block2=Block()
block2.previous_block_hash=last_block_hash
block2.verified_transaction.append(t3)
Nonce=mine(block2,2)
block2.Nonce=mine(block2,2)
digest=hash(block2)
last_block_hash=digest
TPCoins.append(block2)
dump_blockchain(TPCoins)
```

OUTPUT:

```
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.9/dist-packages (3.17)
Kinjal Jaiswal, Roll no: 19
prefix 111
Testing --> 5d3acf3bfd2ab1cdf3948d7dbd8a5729fa25e04a75f9ca11cdbcb30e87dbedfd
Testing --> a81a77e29dad624d11fb08173d6a050f7ce86fec2b9eb07d71b61fbe0bb16692
Testing --> 7257d94d271e1efe0b9475a2e40a6292aa3bb0eb8cec14c4ab4cebc3cd53489e
Testing --> 27add7b3a5826eae668cd31621c4c6526b0874f7f3146b8e9ce6bb46d5f9fdfb
Testing --> 10a8c3f32dc94d13dcce09d0b77a66c11d3ba3c533078e9001f03e58318e0175
Testing --> f00d3138fae2ce3ef625ac6922719141ff011f48c1fa4b9402dd5df6d86bc472
Testing --> fa99d24bec3527259a15f1da4d844707b33a2617b982b190eb251c0e2269a920
Testing --> 17c028cabcacb3a33b0212598cad40d7ceb574b3fd77ba205d5e6b261981859f
Testing --> 676f62c095162ec7b902e8aba0fbcb6c3fa85570ba1c0d82c7a3796a0e4f8e86
Testing --> 72bef226ce7021da694b4b7accb80e529f505aa80015597caf938e73998819d6
Testing --> bdce3a772035196bc3e1524e36e7d0e952e97c04b9de99d5b824e2164da05d29
Testing --> 6b95cef80849cbbdc75e50b39da83dca97c30e37bfe8470ed15fb967768a94ec
Testing --> f2fe014660f88683e9c7f2292e1d168d886e1fbc3c1f947907eae390e747229f
Testing --> 82a7ab0febf4f9d724333926ac38dbaa2dbc7023f182acef82b4bbcbca07c7c3
Testing --> 3a71dcb5400d45d99cbf26ed53c7a6083bcb1f51ff02f8222671dba558ca9310
Testing --> ac5b8aba74f4514de05b59dda5649f33c7f2da83516a789a7d5d38ee19a059c7
Testing --> b067f3deaeaeca69aa3d67f3eb338649d4048ede50eebbc4513d570621e22059
Testing --> c213f1a88acd4c37faf15728894c4c16e925ccd8256f6e7425fc640bc8cdc371
Testing --> 6182c57b2ef4c024c9bb87bdfcd3af8fdd4da03876d56cf7812ea641fb489499
Testing --> 95d8c82c7ae54b4e79278fbf0631caa956c518e3d566d202761a5d7bf1349d3f
Testing --> 12bdbcd3d3b7d375939cd928b6cda26731d065d50eb1eb929c16f8a75488c503
Testing --> 5b635c6100b5c7c690ec6a39beb2692ccc03b4aa571a4fc78f3cfa2197aa6abe
Testing --> a1740350ed13a67772b8337f1faf2be16f4534890736cf9d4b9b47908aef49ef
```

```
After 864iterations found nounce 111ada1bbcd7a012577cafb959c754bdeaab9916e6ef4a073a9dc352af90ee53
prefix 11
Testing --> ed630faa10f0e73153c12b4ed7e6aa3d2e7e8a6f8c38abe34b0b7d162ee0df01
Testing --> 528ccc17a3b66febe92fcb5a5f7ed32ac1cbb73fe2835ddd1dad23ee324ecc79
Testing --> 7cbcff5a4db83d771e639963d6139451f3108afdc518f9ff5da8eafeef311ef6
Testing --> 327cb65195b170be38c9caec7fc6b36ba09debbacce7621bfca6164d6ee8123a
Testing --> 0e6d73f4d94f8f56c8f3f0e6f5411c880ae8b93dffdf78b87e4a9cc070f01d09
Testing --> f40b90922ba3e975f09dbe9983fb85804dad740046e25b0f0269d31f6d3c830c
Testing --> dcf0ef861d92059a7cb45e0a1be40c34074a490c901198453d89d36323521151
Testing --> ff730b63e0c7db5e9b8b212c6336b6b502075a1596dcfd4ecc76e7881524ef73
Testing --> 837ec82823f36332ee40f76a08fe70af657dfb66789fafe985a0c8fd628addc1
Testing --> 3392faa55e8e43d3836e72b075f1a790bad5aba3fa6da9abc8d55b070c4d5b0b
Testing --> 934b1f57863bba7f0031e39d32e2ba003333851eae2435c40c670f7483dce6fa
Testing --> 4cc9dbcf531fc89a0b35a736b65db80cfbf9bf5b2099a0975802cfe7632f4ba0
```

```
After 739iterations found nounce 11fe7877ebcd64dc7760543e550410378f5b7ab54ecc46e72b8829a13110b067
Number of blocks in the chain:3
block#0
sender:Genesis
-----
recipent:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
value:500.0
-----
time:2023-04-19 09:47:25.027621
-----
.............
====================
block#1
sender:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
recipent:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
value:15.0
-----
time:2023-04-19 09:47:25.027929
-----
.............
====================
sender:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
```

```
====================
sender:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
recipent:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
value:17.0
-----
time:2023-04-19 09:47:25.029829
-----
.............
====================
block#2
sender:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
recipent:30819f300d06092a864886f70d010101050003818d003081890281810...
-----
value:10.0
-----
time:2023-04-19 09:47:25.031515
-----
.............
====================
```

PRACTICAL No: 4

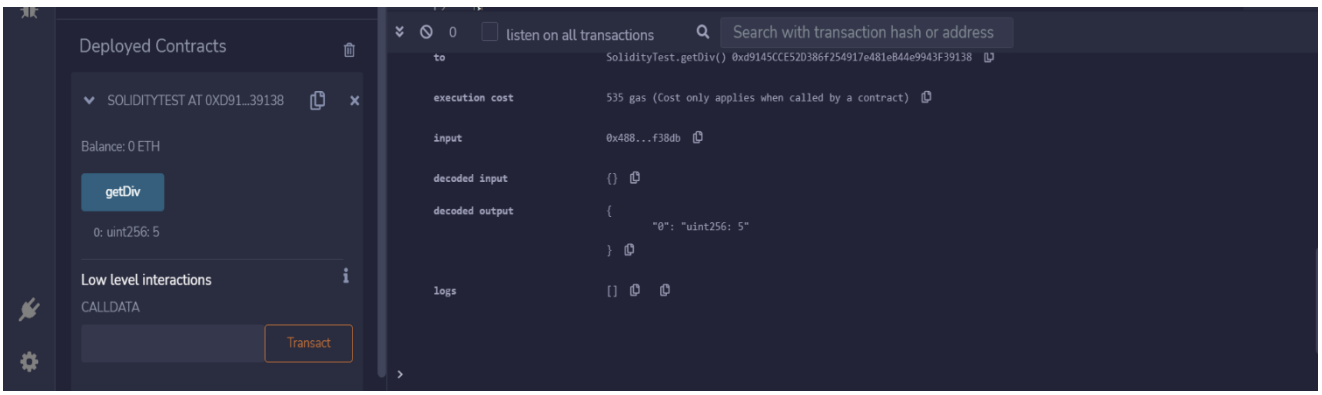
1. **Variable**
2. **Operations**
3. **Loops**
4. **Decision Making**
5. **Strings**

AIM: 1. Variable

CODE:

```
pragma solidity ^0.8.0;
contract SolidityTest {
   uint storedData; // State variable
   constructor() public{
      storedData=10;
   }
   function getDiv() public view returns(uint){
      uint a=10; // local variable
      uint b=2;
      uint result = a / b;
   return result; // accesss the state variable
}
}
```

OUTPUT:



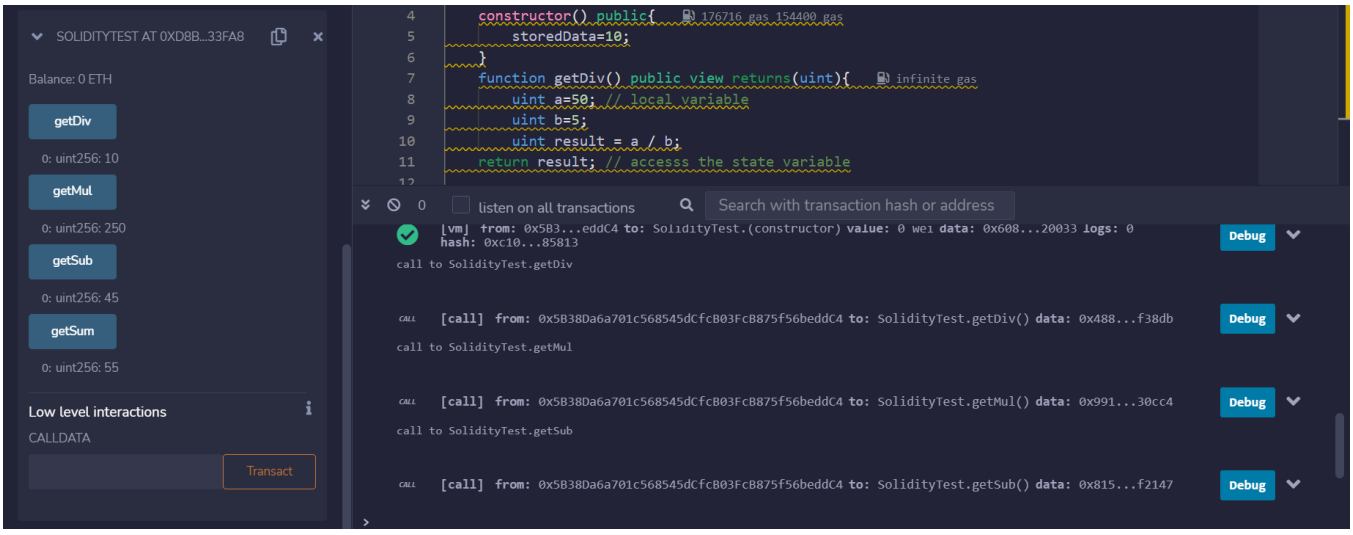
AIM: 2. Operations

CODE:

```
pragma solidity ^0.8.0;
contract SolidityTest {
   uint storedData; // State variable
   constructor() public{
      storedData=10;
   }
   function getDiv() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
      uint result = a / b;
   return result; // accesss the state variable

   }
   function getMul() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
      uint result = a * b;
   return result; // accesss the state variable
   }
   function getSum() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
      uint result = a + b;
   return result; // accesss the state variable
   }
   function getSub() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
```

```
        uint result = a - b;
    return result; // accesss the state variable
    }
}
```

OUTPUT:



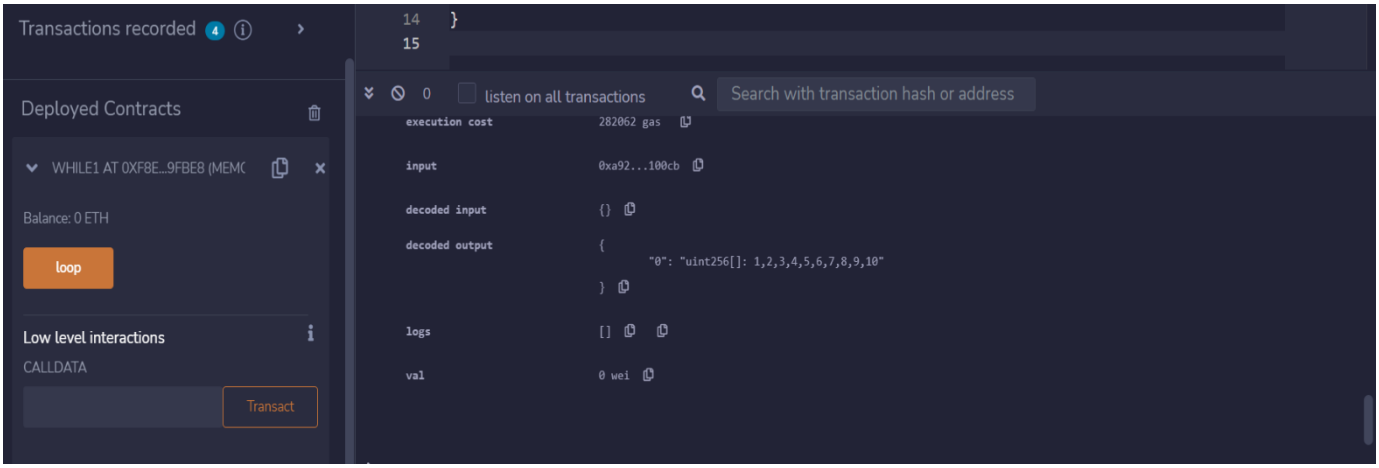
AIM: 3. Loops

a) While Loop

CODE:

```
pragma solidity ^0.8.0;
contract while1{
uint[] data;
uint8 j=0;
function loop() public returns(uint[] memory)
{
   while (j<10)
   {
      j++;
      data.push(j);
   }
   return data;
}
}
```

OUTPUT:

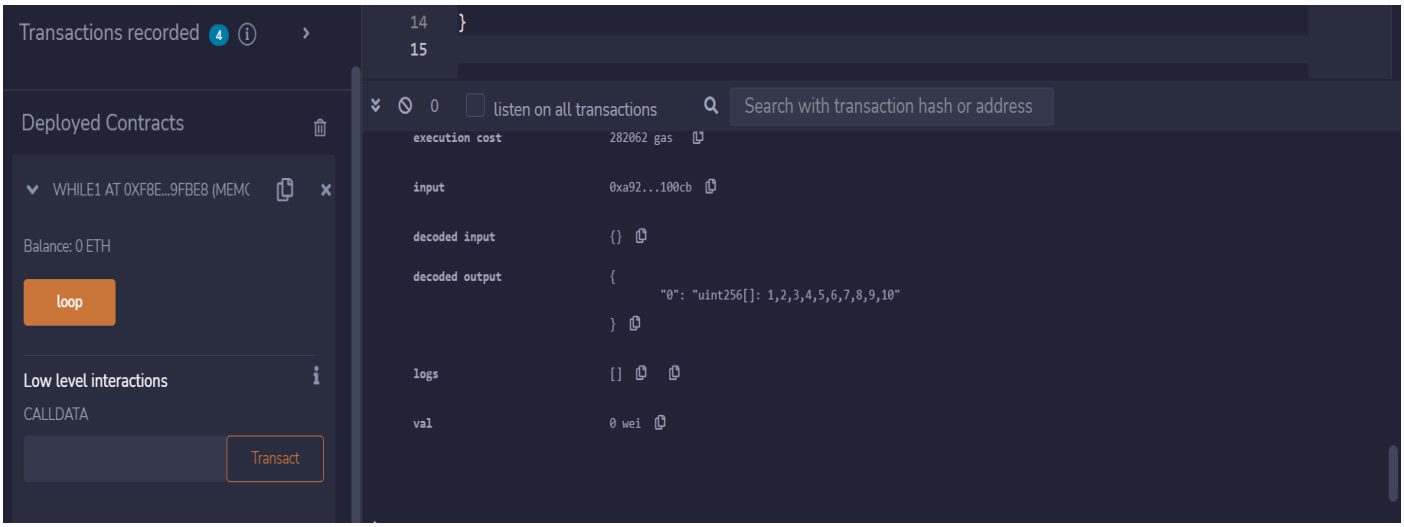


b). Do While Loop.

CODE:

```
pragma solidity ^0.8.0;
contract doWhile1{
uint[] data;
uint8 j=0;
function loop() public returns(uint[] memory)
{
   do
   {
      j++;
      data.push(j);
   }
   while (j<10);
   return data;
}
}
```

OUTPUT:

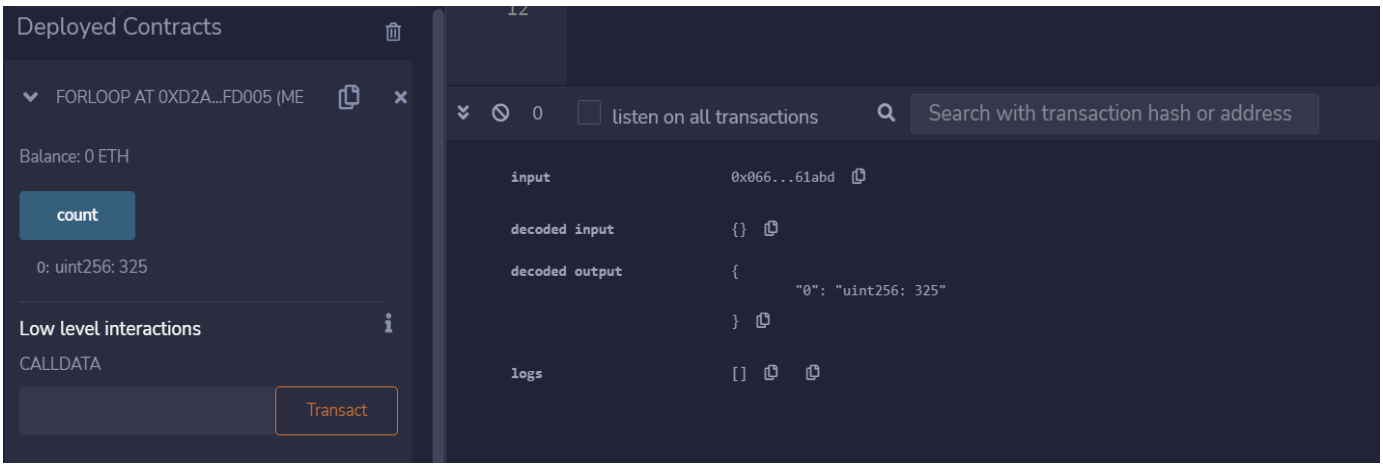


b). For Loop.

CODE:

```
pragma solidity ^0.8.0;
contract ForLoop{
    function count() public pure returns(uint256){
        uint256 sum=0;
        for(uint256 i=0;i<=25;i++){
            sum+=i;
        }
        return sum;
    }
}
```

OUTPUT:



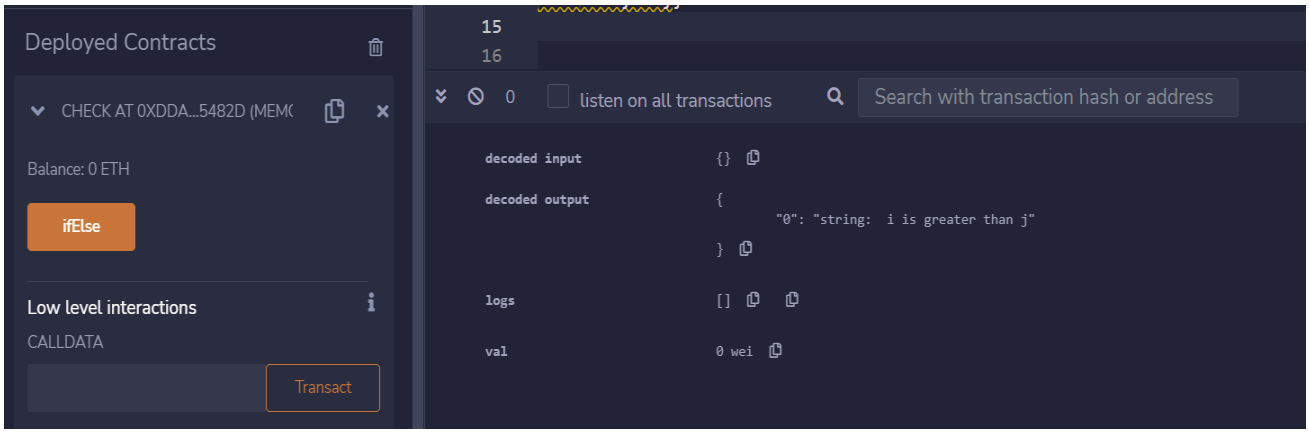
AIM: 4. Decision Making

a) If Else

CODE:

```
pragma solidity ^0.8.0;
contract Check{
   uint i=100;
   uint j=80;
   function ifElse() public returns(string memory)
   {
      if(i<j)
      {
         return "i is smaller than j";
      }
      else
      {
         return " i is greater than j";
      }  }}
```

OUTPUT:

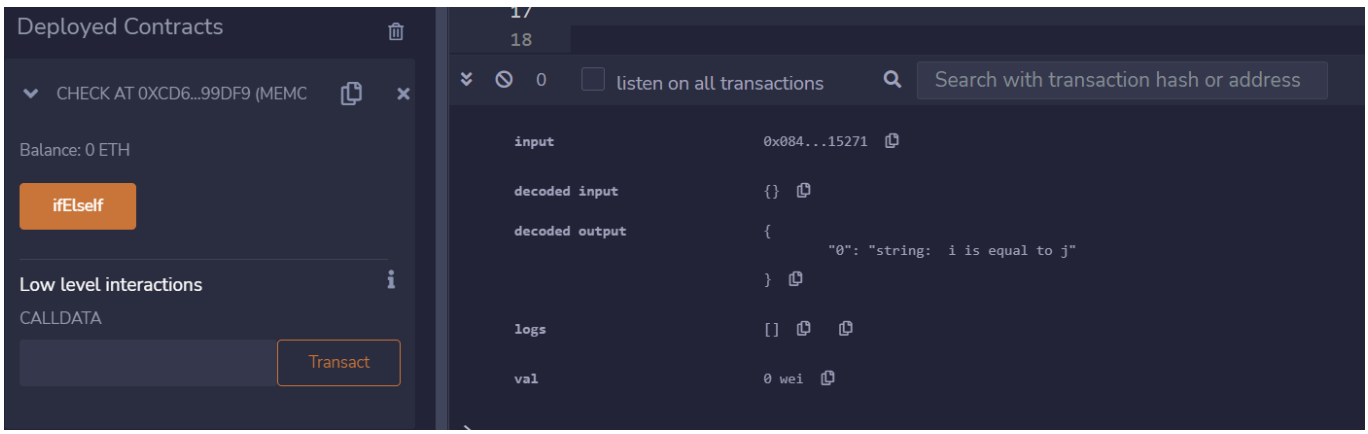


a) If Else - If

CODE:

```
pragma solidity ^0.8.0;
contract Check{
   uint i=100;
   uint j=100;
   function ifElseIf() public returns(string memory)
   {
      if(i<j)
      {
         return "i is smaller than j";}
      else if(i>j)
      {
         return " i is greater than j"; }
      else
      {
         return " i is equal to j";
      }}}
```

OUTPUT:



AIM: 4. Strings

a) Regular Strings

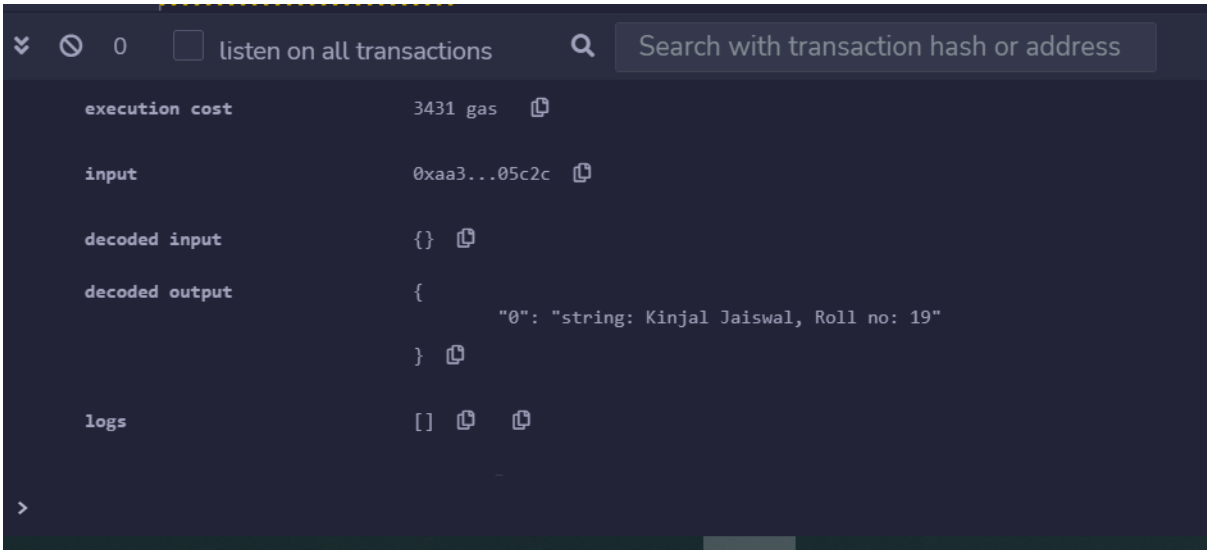
CODE:

```
pragma solidity ^0.8.0;
contract SS{
   string str1="M.SC I.T PART 2";
   string str2='Kinjal Jaiswal, Roll no: 19';
   string str3=new string(20);
   function getstr1() public returns(string memory)
   {
      return str1;   }
   function getstr2() public returns(string memory)
   {
      return str2;  }
   function getstr3() public returns(string memory)
   {
      return str3;
   }
}
```

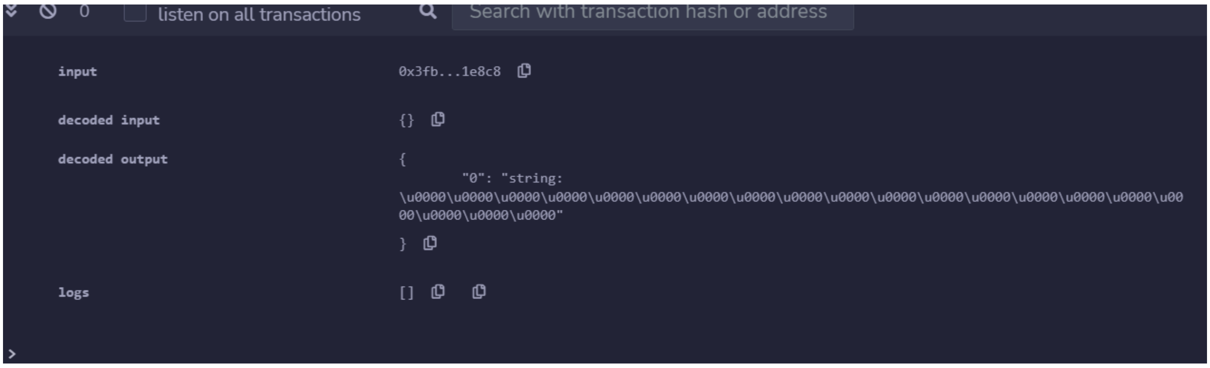
OUTPUT:

String 1

String 2



String 3

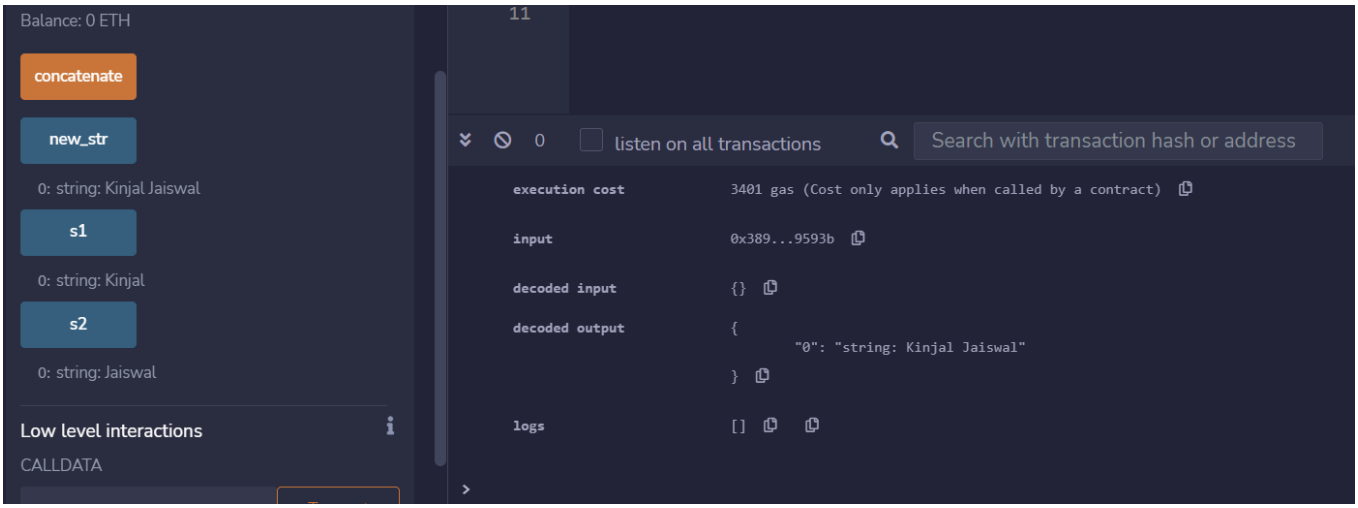


b) Concatenate

CODE:

```
pragma solidity >=0.5.0 <0.9.0;
  contract Demo{
    string public s1 = "Kinjal ";
    string public s2 = "Jaiswal";
    string public new_str;

    function concatenate() public {
      new_str = string(abi.encodePacked(s1, s2));
     }
}
```

OUTPUT:

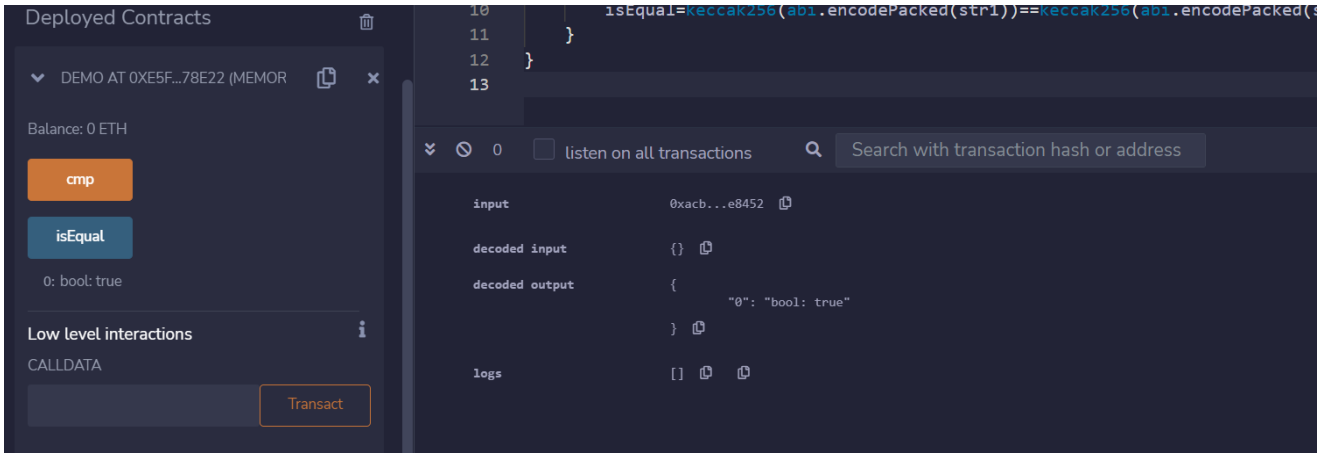


a) Compare

CODE:

```
pragma solidity ^0.8.0;
contract Demo{
    string str1="Kinjal";
    string str2='Kinjal';

    bool public isEqual;

    function cmp() public
    {
        isEqual=keccak256(abi.encodePacked(str1))==keccak256(abi.encodePacked(str2));
    }
}
```

OUTPUT:



PRACTICAL No: 5

1. **Arrays**
2. **Enums**
3. **Structs**
4. **Mappings**
5. **Conversations**
6. **Ether Units**
7. **Special Variables**

AIM: 1. Arrays

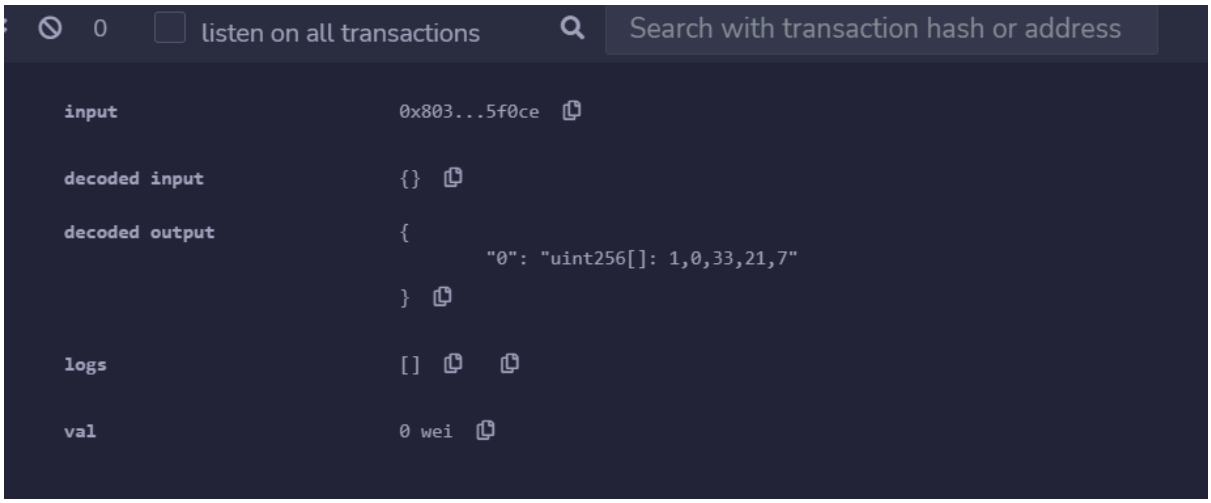
CODE:

```
pragma solidity ^0.5.0;
contract Arrray{
   uint[] nums=[1,2,33,21];

   function getlength() public returns(uint){
      return nums.length;
   }

        function pop() public{
                delete nums[1];
        }
        function push() public returns (uint[] memory){
                nums.push(7);
                return nums;
        }
        function push1(uint i) public{
                 nums.push(i);
        }
}
```

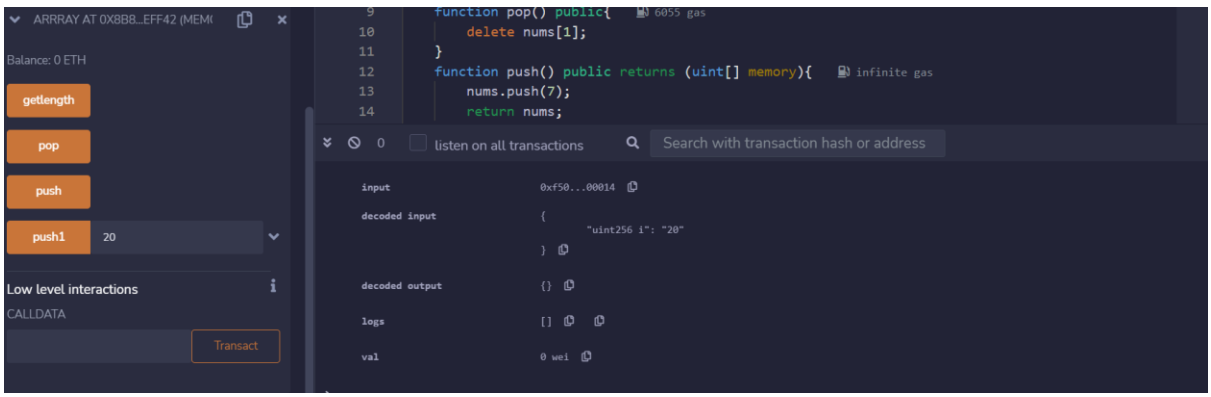
OUTPUT:

Get Length

Pop



Push

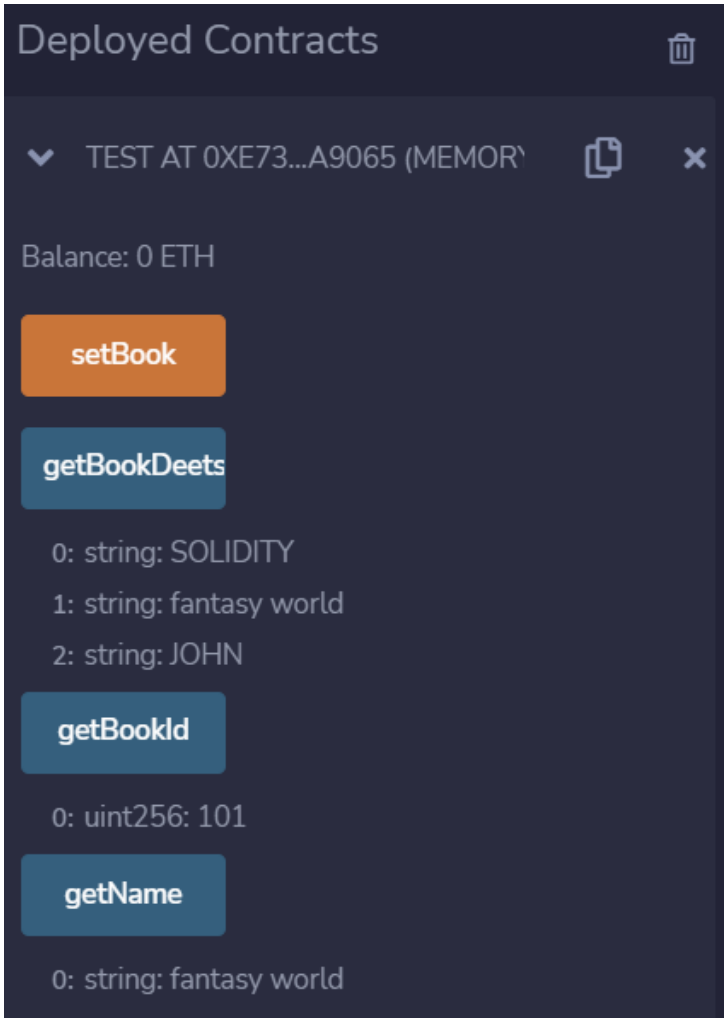


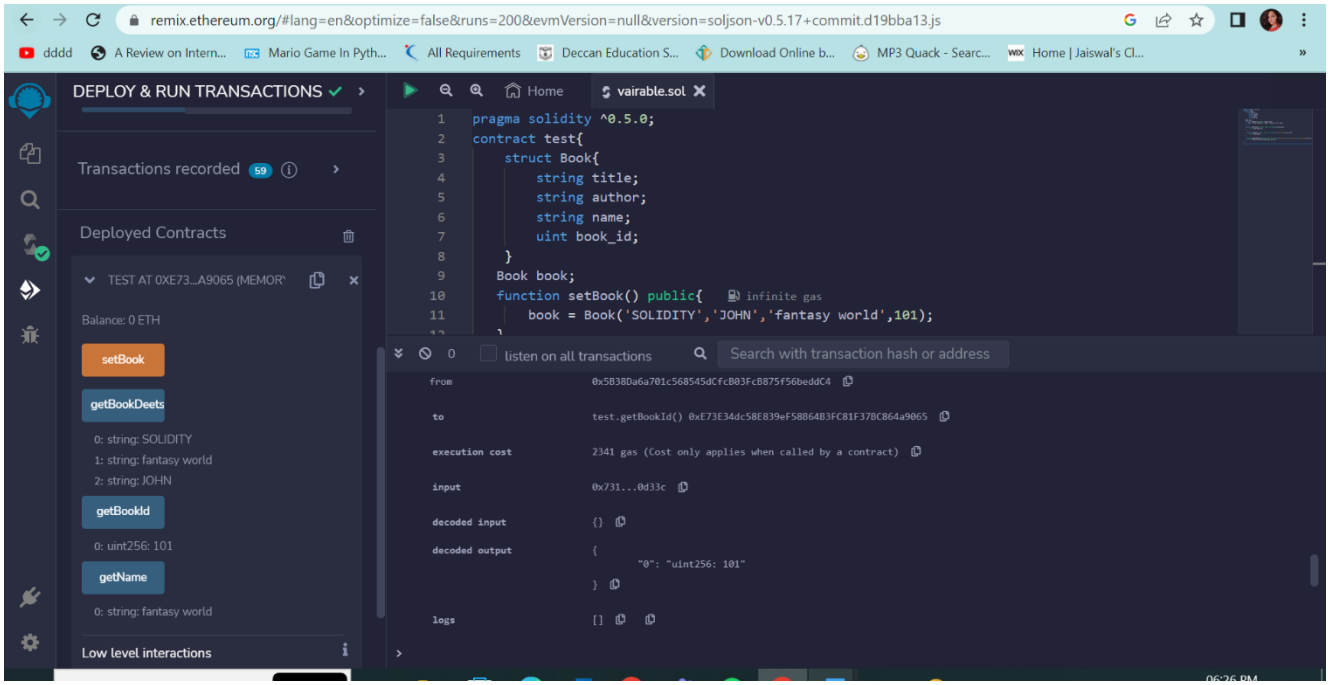
AIM: 2. Struct

CODE:

```
pragma solidity ^0.5.0;
contract test{
   struct Book{
       string title;
       string author;
       string name;
       uint book_id;
   }
  Book book;
  function setBook() public{
      book = Book('SOLIDITY','JOHN','fantasy world',101);
  }

  function getBookId() public view returns(uint){
       return book.book_id;
  }

   function getName() public view returns(string memory){
       return book.name;
  }

  function getBookDeets() public view returns(string memory,string memory,string
memory){
      return(book.title,book.name,book.author);
  }
}
```

OUTPUT:


Deployed Contracts
TEST AT 0XE73...A9065 (MEMOR
Balance: 0 ETH
setBook
getBookDeets
0: string: SOLIDITY
1: string: fantasy world
2: string: JOHN
getBookId
0: uint256: 101
getName
0: string: fantasy world


remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
DEPLOY & RUN TRANSACTIONS
Transactions recorded 59
Deployed Contracts
TEST AT 0XE73...A9065 (MEMOR
Balance: 0 ETH
setBook
getBookDeets
0: string: SOLIDITY
1: string: fantasy world
2: string: JOHN
getBookId
0: uint256: 101
getName
0: string: fantasy world
Low level interactions
Home
vairable.sol
pragma solidity ^0.5.0;
contract test{
struct Book{
string title;
string author;
string name;
uint book_id;
}
Book book;
function setBook() public{
book = Book('SOLIDITY','JOHN','fantasy world',101);
listen on all transactions
Search with transaction hash or address
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to test.getBookId() 0xE73E34dc58E839eF58864B3FC81F37BC864a9065
execution cost 2341 gas (Cost only applies when called by a contract)
input 0x731...0d33c
decoded input {}
decoded output { "0": "uint256: 101" }
logs []

AIM: 3. Enum

CODE:

```
pragma solidity ^0.5.0;
contract Types{
    enum week_days{
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday,
        Saturday,
        Sunday
    }
    week_days week;
    week_days choice;
    week_days constant default_value = week_days.Sunday;
    function set_value() public{
        choice = week_days.Wednesday;
    }
    function get_choice() public view returns(week_days){
            return choice;
    }
}
```

OUTPUT:

AIM: 4. Mapping

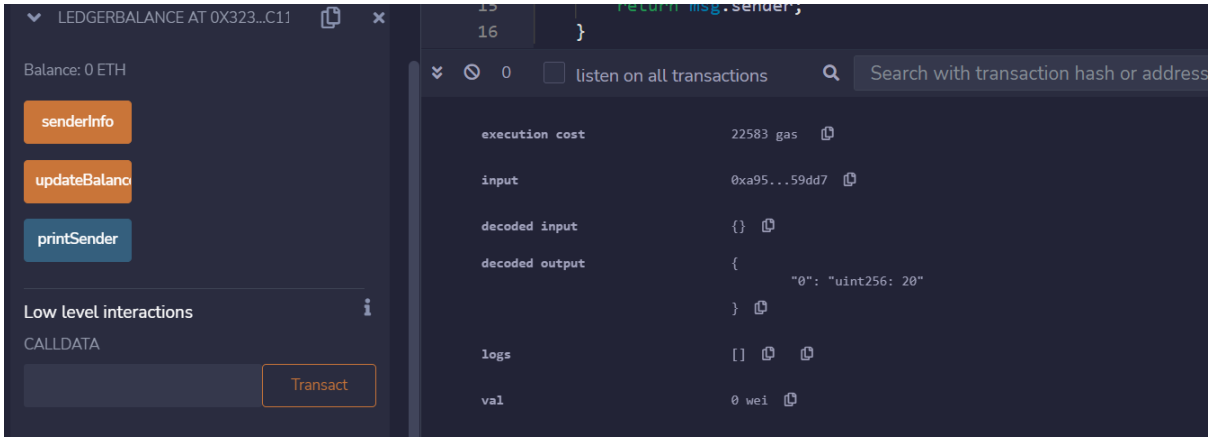
CODE:

```
pragma solidity ^0.5.0;
contract LedgerBalance{
   mapping(address => uint) balance;
   mapping(address => string) name;
   function updateBalance() public returns(uint){
         balance[msg.sender]=20;
         return balance[msg.sender];
   }

   function senderInfo() public returns(string memory){
      name[msg.sender] = "tanya";
      return name[msg.sender];
   }
   function printSender() public view returns(address){
      return msg.sender;
   }
}
```

OUTPUT:

Sender Info

Update Balance



Print Sender

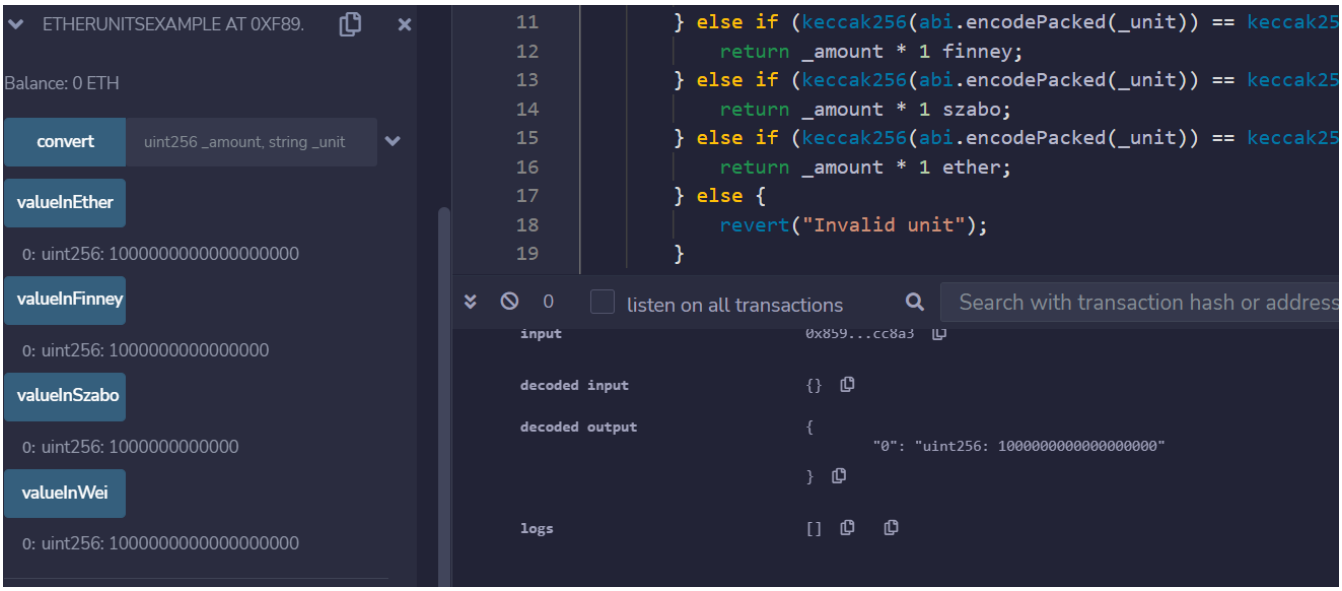
AIM: 5. Conversions

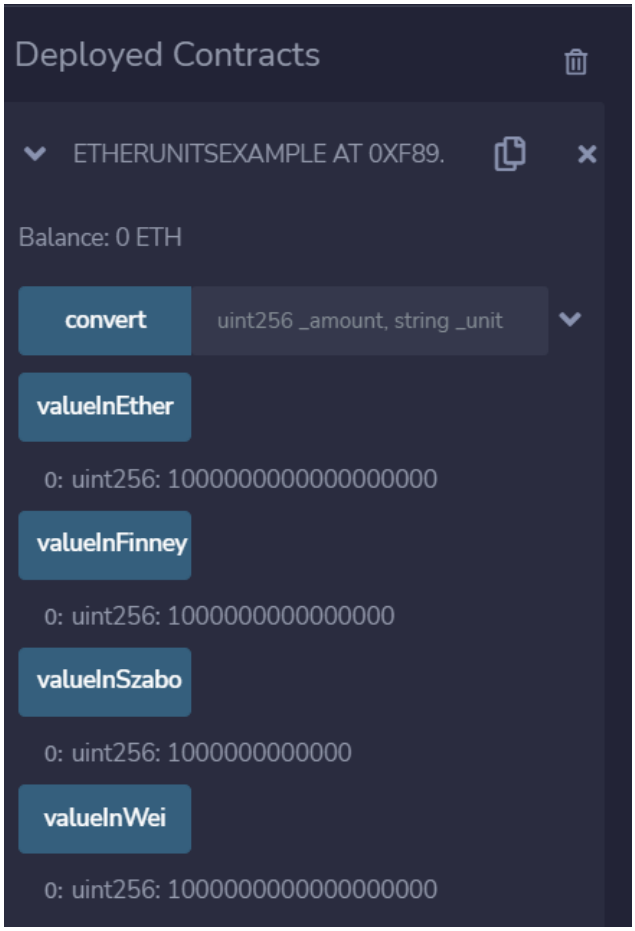
6. Ether Units

CODE:

```
pragma solidity >=0.4.0 <0.7.0;

contract EtherUnitsExample {

   uint256 public valueInWei = 1 ether; // 1 ether in Wei

   uint256 public valueInFinney = 1 finney; // 1 finney in Wei

   uint256 public valueInSzabo = 1 szabo; // 1 szabo in Wei

   uint256 public valueInEther = 1 ether; // 1 ether in Wei


   function convert(uint256 _amount, string memory _unit) public pure returns (uint256) {

      if (keccak256(abi.encodePacked(_unit)) == keccak256(abi.encodePacked("wei"))) {

         return _amount;

      } else if (keccak256(abi.encodePacked(_unit)) ==
keccak256(abi.encodePacked("finney"))) {

         return _amount * 1 finney;

      } else if (keccak256(abi.encodePacked(_unit)) ==
keccak256(abi.encodePacked("szabo"))) {

         return _amount * 1 szabo;

      } else if (keccak256(abi.encodePacked(_unit)) ==
keccak256(abi.encodePacked("ether")) || keccak256(abi.encodePacked(_unit)) ==
keccak256(abi.encodePacked("eth"))) {

         return _amount * 1 ether;

      } else {

         revert("Invalid unit");

      }

   }}
```

OUTPUT:


ETHERUNITSEXAMPLE AT 0XF89.
Balance: 0 ETH
convert
uint256 _amount, string _unit
valueInEther
0: uint256: 1000000000000000000
valueInFinney
0: uint256: 1000000000000000
valueInSzabo
0: uint256: 1000000000000
valueInWei
0: uint256: 1000000000000000000
} else if (keccak256(abi.encodePacked(_unit)) == keccak25
return _amount * 1 finney;
} else if (keccak256(abi.encodePacked(_unit)) == keccak25
return _amount * 1 szabo;
} else if (keccak256(abi.encodePacked(_unit)) == keccak25
return _amount * 1 ether;
} else {
revert("Invalid unit");
}
0
listen on all transactions
Search with transaction hash or address
input
decoded input
{}
decoded output
{
"0": "uint256: 1000000000000000000"
}
logs
[]


Deployed Contracts
ETHERUNITSEXAMPLE AT 0XF89.
Balance: 0 ETH
convert
uint256 _amount, string _unit
valueInEther
0: uint256: 1000000000000000000
valueInFinney
0: uint256: 1000000000000000
valueInSzabo
0: uint256: 1000000000000
valueInWei
0: uint256: 1000000000000000000

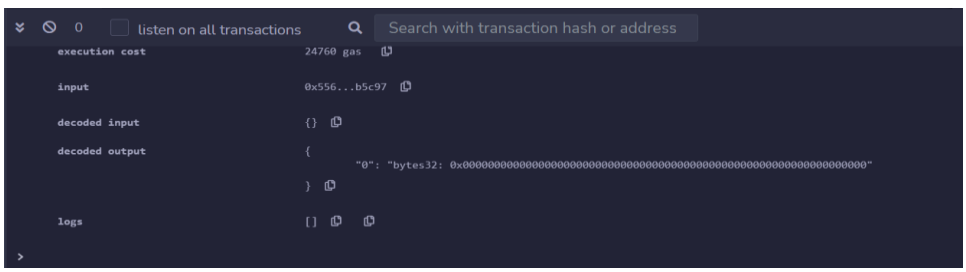
AIM: 7. Special Variables.

a). Solidity contract to demonstrate the special variables block. Number and block hash.

CODE:

```
pragma solidity ^0.5.0;

contract prac

{

        uint BNumber;

        bytes32 BHashPresent;

        bytes32 BHashPrevious;

        function PresentHash()

                        public returns(bytes32)

        {

                BNumber = block.number;

                BHashPresent =blockhash(BNumber);

                return BHashPresent;

        }

        function PreviousHash()

                        public returns(bytes32)

        {

                BNumber = block.number;

                BHashPrevious = blockhash(BNumber - 1);

                return BHashPrevious;

        }

}
```

OUTPUT:

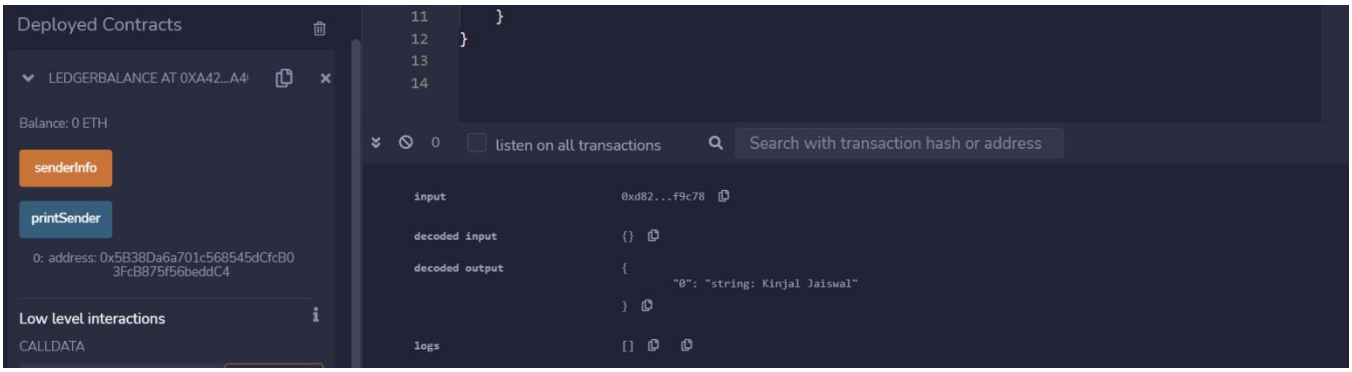


b). Solidity contract to demonstrate msg.sender

CODE:

```
pragma solidity ^0.5.0;
contract LedgerBalance{
    mapping(address => string) name;
    function senderInfo() public returns(string memory){
        name[msg.sender] = "Kinjal Jaiswal";
        return name[msg.sender];
    }
    function printSender() public view returns(address){
        return msg.sender;
    }
}
```

OUTPUT:



PRACTICAL No: 6

AIM: Implement and demonstrate the use of the following in Solidity.

1. **Functions**
2. **View Functions**
3. **Pure Functions**
4. **Fallback Functions**
5. **Function Overloading**
6. **Mathematical Functions**
7. **Cryptographic Functions**

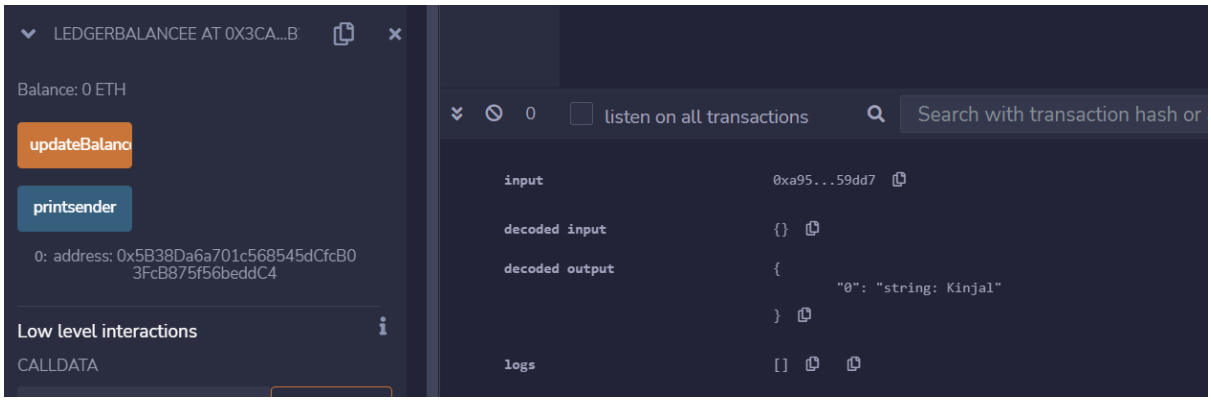
AIM: 1. Functions

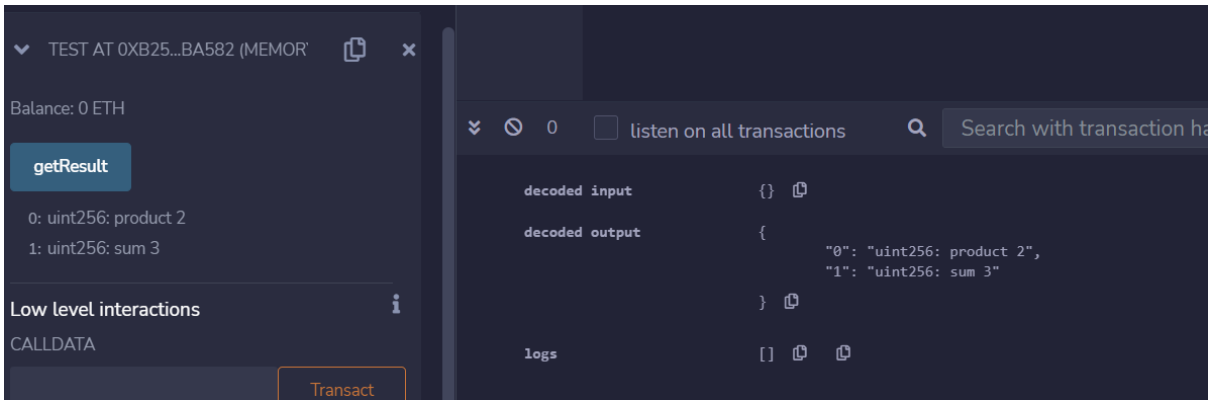
CODE:

```
pragma solidity ^0.8.0;
contract LedgerBalancee {
    mapping(address => string) name;

    function updateBalance() public returns(string memory) {
        name[msg.sender]="tan";
        return name [msg.sender];
    }
    function printsender() public view returns(address){
        return msg.sender;
    }

}
contract Test {
   function getResult() public view returns(uint product, uint sum){
      uint a = 1; // local variable
      uint b = 2;
```

```
    product = a * b;
    sum = a + b;
    //return(a*b, a+b);
  }}
```

OUTPUT:



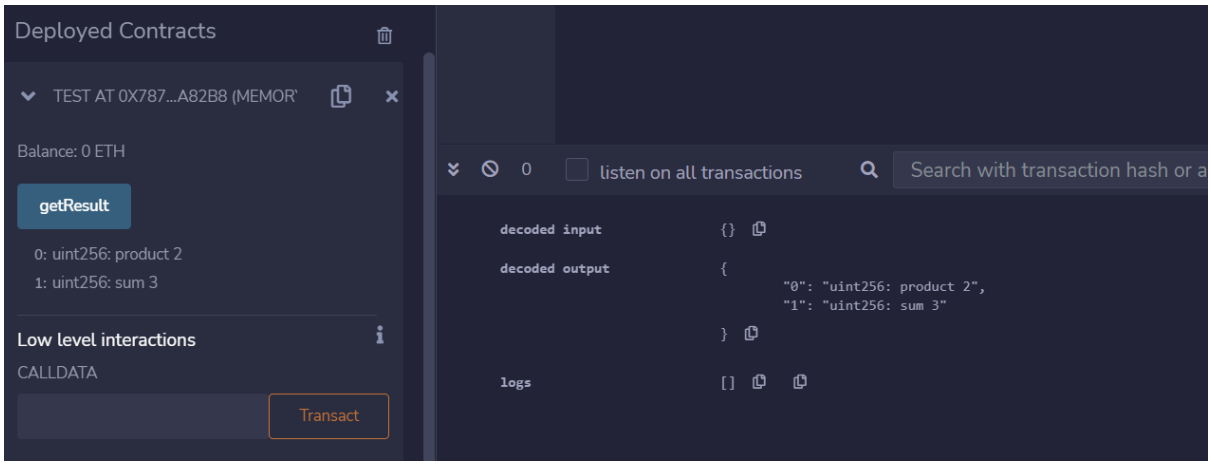


AIM: 2. View Functions

CODE:

```
pragma solidity ^0.8.0;
contract Test{
    function getResult() public view returns(uint product,uint sum){
        uint a = 1; //local variable
        uint b = 2;
        product = a*b;
        sum = a+b;
    }
}
```

OUTPUT:

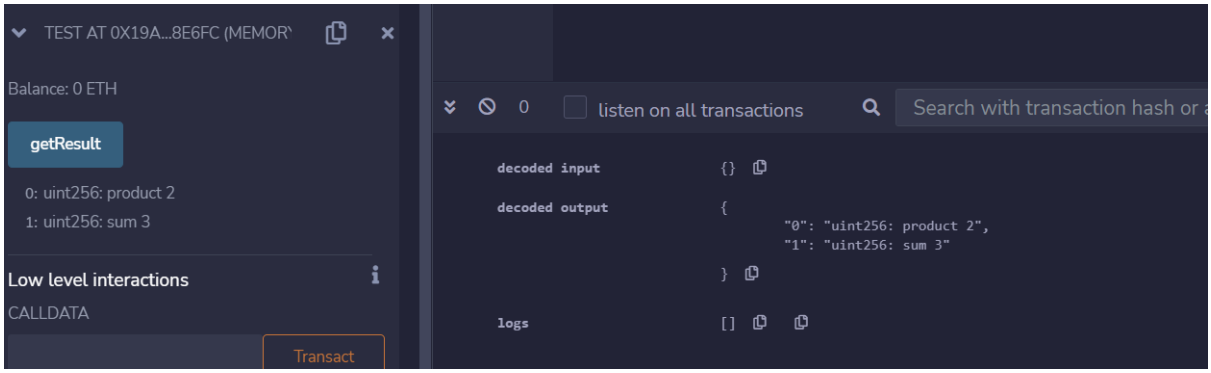


AIM: 3. Pure Functions

CODE:

```
pragma solidity ^0.5.0;
contract Test{

   function getResult() public pure returns(uint product,uint sum){
      uint a = 1; //local variable
      uint b = 2;
      product = a*b;
      sum = a+b;
   }
}
```

OUTPUT:

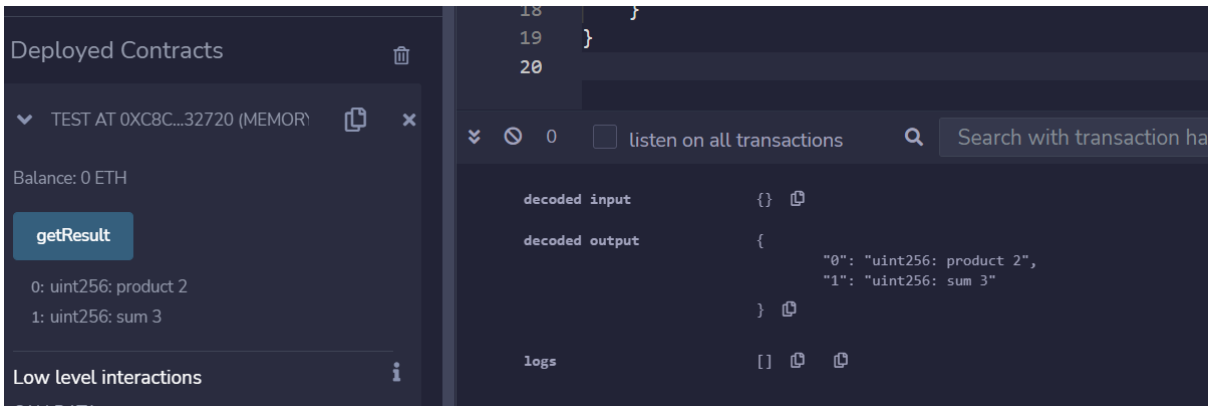


AIM: 4. Fallback Functions

CODE:

```
pragma solidity ^0.8.12;
contract A {
    uint n;
    function set(uint value) external {
        n=value;
    }
    //fallback function
    function() external payable{
        n=0;
    }
}
contract example{
    function callA(A a) public returns (bool){
        (bool success,) = address(a).call(abi.encodeWithSignature("setter()"));
        require(success);
        address payable payableA=address(uint160(address(a)));
        return(payableA.send(2 ether));
    }
}
```

OUTPUT:

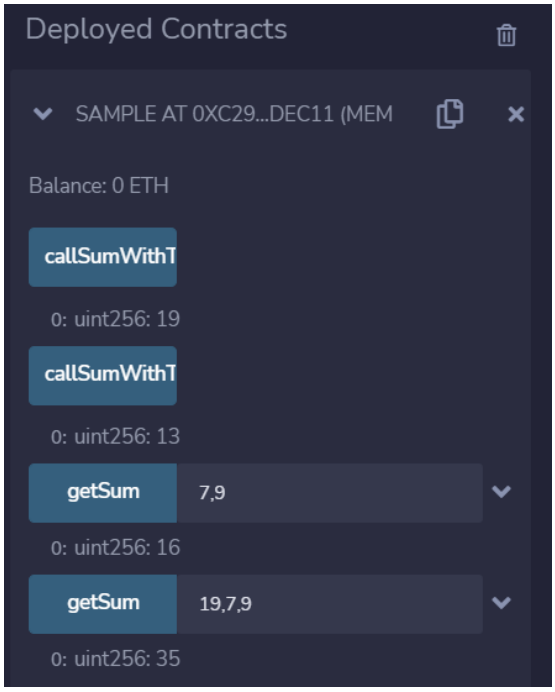


AIM: 5. Functions Overloading

CODE:

```
pragma solidity ^0.8.12;
contract Sample{
    function getSum(uint a, uint b) public pure returns (uint){
        return a+b;
    }
    function getSum(uint a, uint b, uint c) public pure returns (uint){
        return a+b+c;
    }
    function callSumWithTwoArguments() public pure returns (uint){
        return getSum(4,9);
    }
     function callSumWithThreeArguments() public pure returns (uint){
        return getSum(4,9,6);
    }
}
```

OUTPUT:

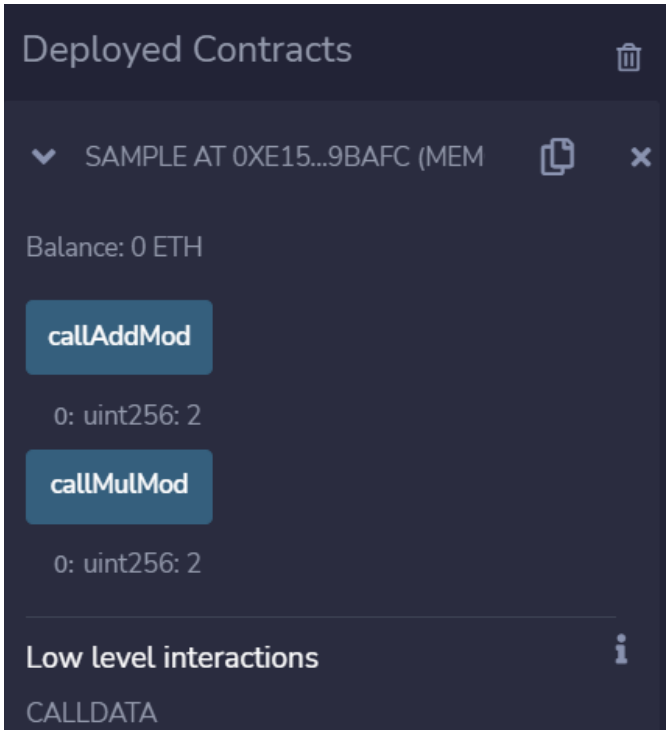


AIM: 6. Mathematical Functions

CODE:

```
pragma solidity ^0.8.0;
contract Sample{
    function callAddMod() public pure returns (uint){
        return addmod(3,4,5);
//3+4 % 5
    }
   function callMulMod() public pure returns (uint){
        return mulmod(3,4,5);
    }
//3*4 % 5
}
```

OUTPUT:

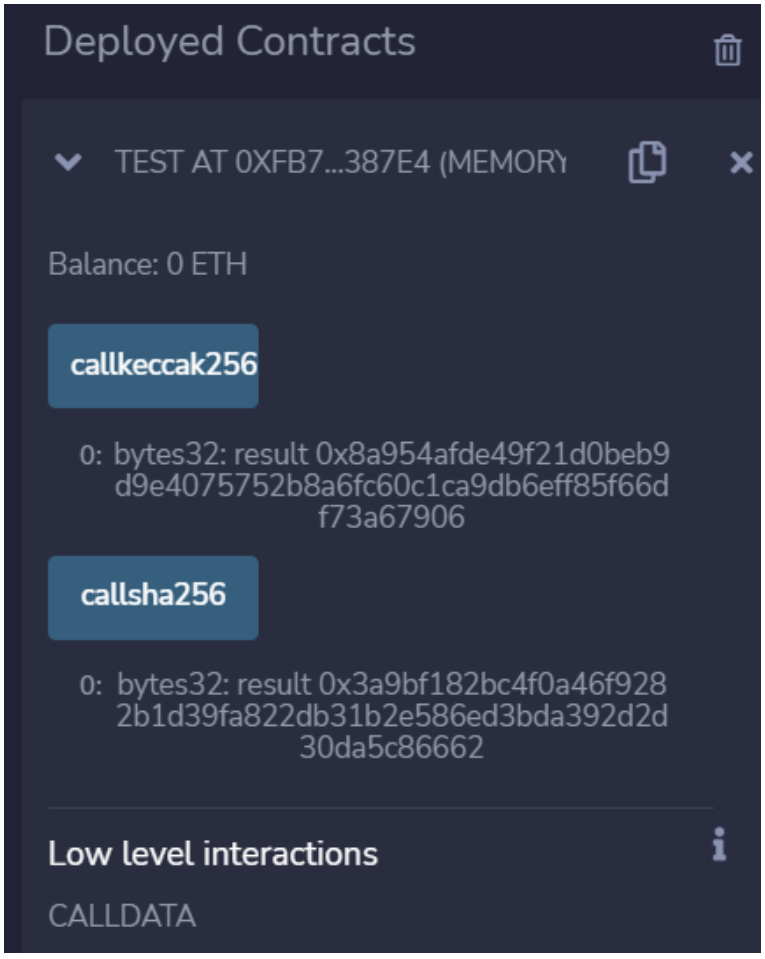


AIM: 7. Cryptographic Functions

CODE:

```
pragma solidity ^0.8.12;
contract Test{
    function callsha256() public pure returns(bytes32 result){
        return sha256("Kinjal");
    }
    function callkeccak256() public pure returns(bytes32 result){
        return keccak256("Kinjal");
    }
}
```

OUTPUT:



PRACTICAL No: 7

AIM: Implement and demonstrate the use of the following in Solidity

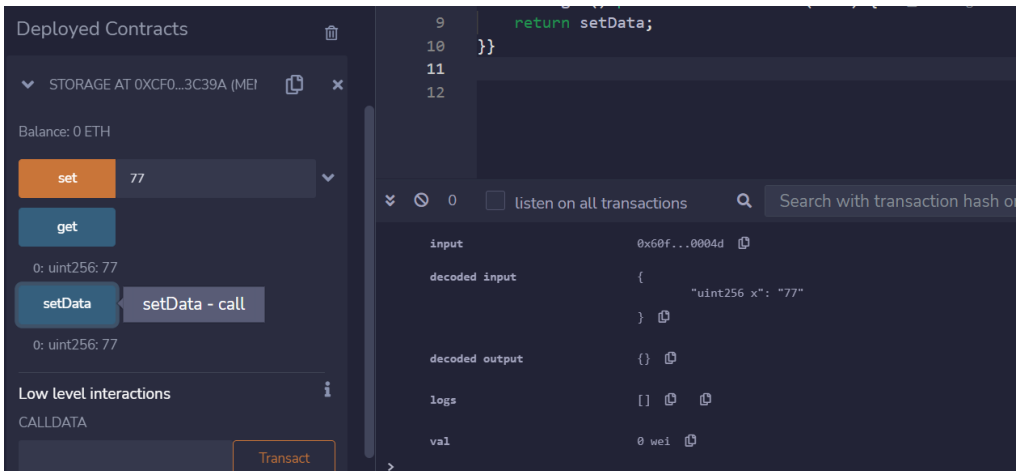
1. **Contracts**
2. **Inheritance**
3. **Constructors**
4. **Abstract class**
5. **Interfaces**

AIM: 1. Contracts

CODE:

```
pragma solidity ^0.8.0;
contract Storage
{
uint public setData;
function set(uint x) public{
setData = x;
}
function get() public view returns (uint) {
   return setData;
}}
```

OUTPUT:



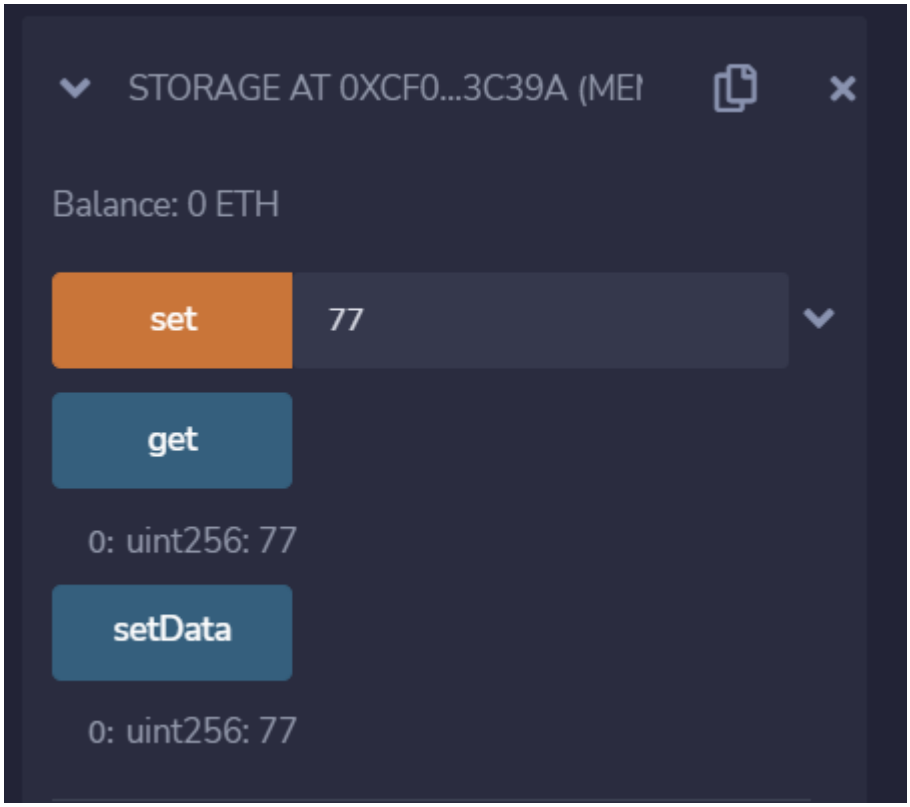
AIM: 2. Inheritance

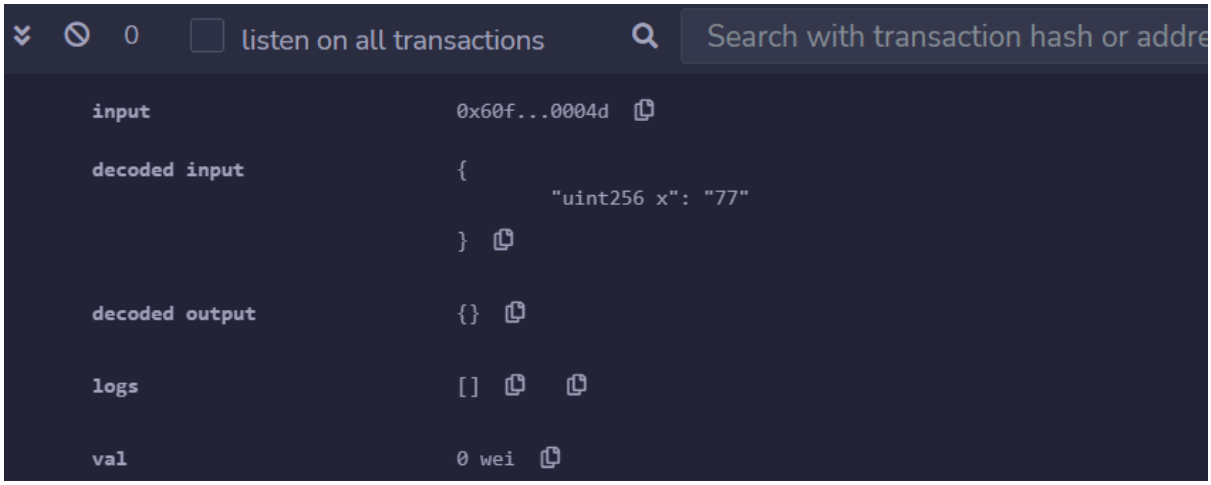
a). Single Inheritance

CODE:

```
pragma solidity 0.5.0;
contract parent{
	uint internal sum;
	function setValue() external {
		uint a = 10;
		uint b = 25;
		sum = a + b;
	}
}
contract child is parent{ //defining the child contract
	function getValue(
	) external view returns(uint) {
		return sum;
	}
}
contract caller {
	child cc = new child();

	function testInheritance(
	) public returns (uint) {
		cc.setValue();
		return cc.getValue();
	}
}
```

OUTPUT:


STORAGE AT 0XCF0...3C39A (MEI
Balance: 0 ETH
set
77
get
0: uint256: 77
setData
0: uint256: 77


0
listen on all transactions
Search with transaction hash or addre
input
0x60f...0004d
decoded input
{
"uint256 x": "77"
}
decoded output
{}
logs
[]
val
0 wei

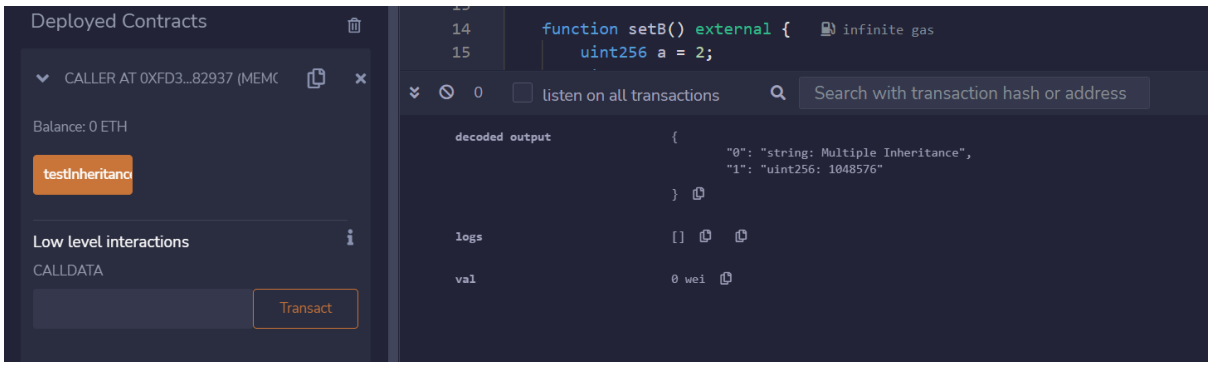
b). Multiple Inheritance

CODE:

```
pragma solidity ^0.8.0;

contract A {
    string internal x;

    function setA() external {
        x = "Multiple Inheritance";
    }
}

contract B {
    uint256 internal pow;

    function setB() external {
        uint256 a = 2;
        uint256 b = 20;
        pow = a**b;
    }
}

contract C is A, B {

    function getStr() external view returns (string memory) {
        return x;
    }

    function getPow() external view returns (uint256) {
```

```
        return pow;
    }

}

contract caller {

    C contractC = new C();

    function testInheritance() public returns (string memory, uint256) {
        contractC.setA();
        contractC.setB();
        return (contractC.getStr(), contractC.getPow());
    }
}
```

OUTPUT:



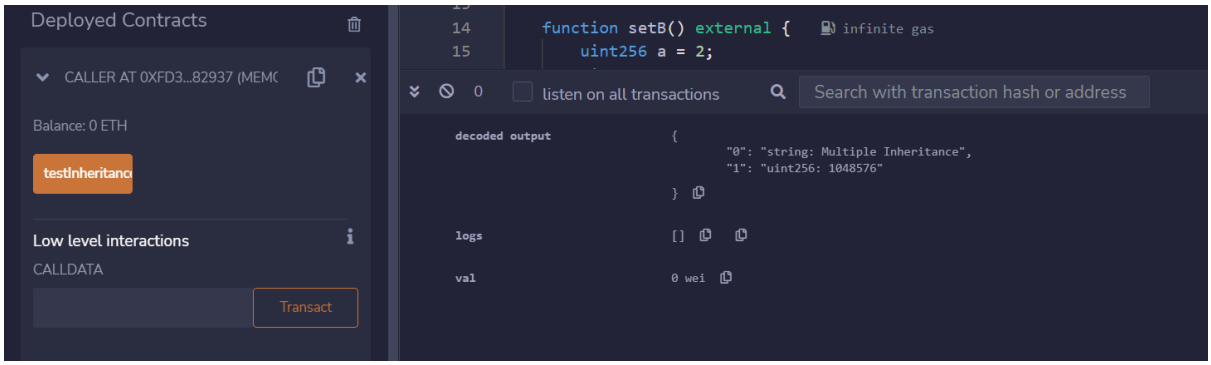
c). Multi-Level Inheritance

CODE:

```
pragma solidity ^0.5.0;

contract A {

    uint256 internal x;

  function setX() external {
     x=10;
  }
}
contract B is A {

    uint256 internal y;

  function setY() external {
     y=20-x;
  }
}
contract C is B{
  function getY() external view returns(
    uint){
      return y;
  }
}
contract caller {
  C cc = new C();
  function testInheritance(
```

```
    ) public returns (
      uint256) {
        cc.setX();
        cc.setY();
        return cc.getY();
    }
}
```

OUTPUT:

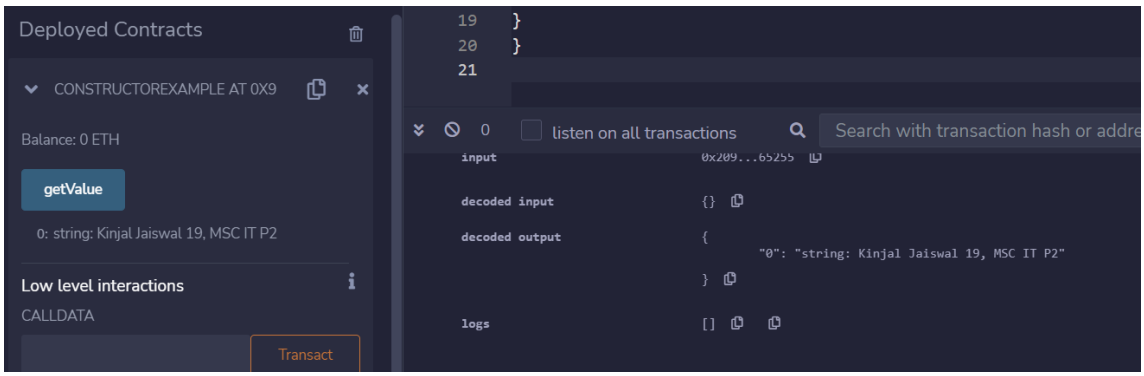


AIM: 3. Constructors

CODE:

```
// Solidity program to demonstrate
// creating a constructor
pragma solidity ^0.8.0;

// Creating a contract
contract constructorExample {

// Declaring state variable
string str;

constructor() public {
str = "Kinjal Jaiswal 19, MSC IT P2";
}
// Defining function to
// return the value of 'str'
function getValue(
) public view returns (
string memory) {
return str;
}
}
```

OUTPUT:



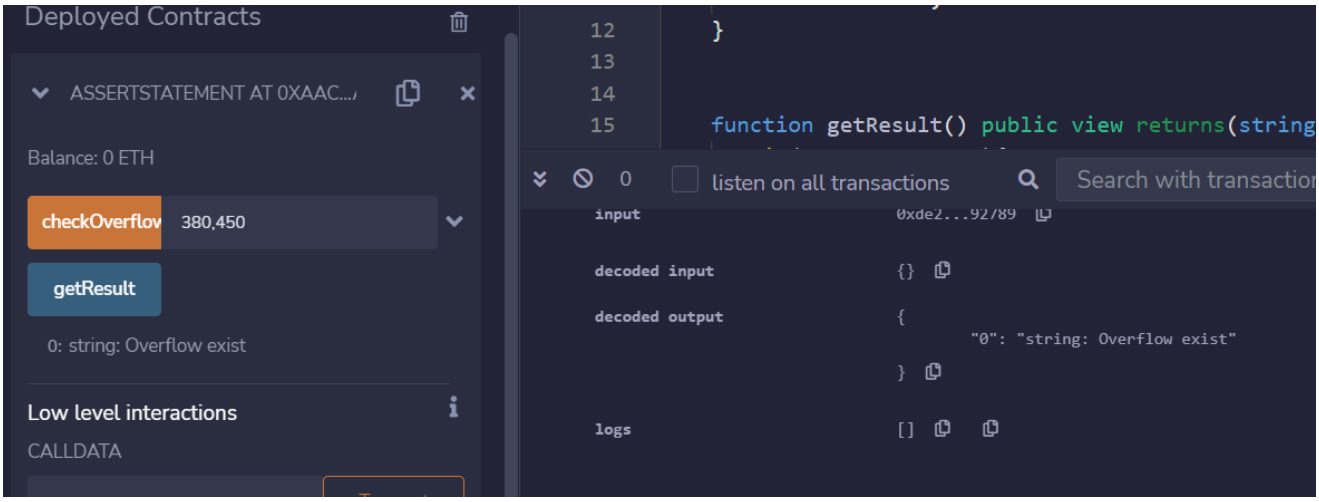
AIM: 4. Abstract Class

CODE:

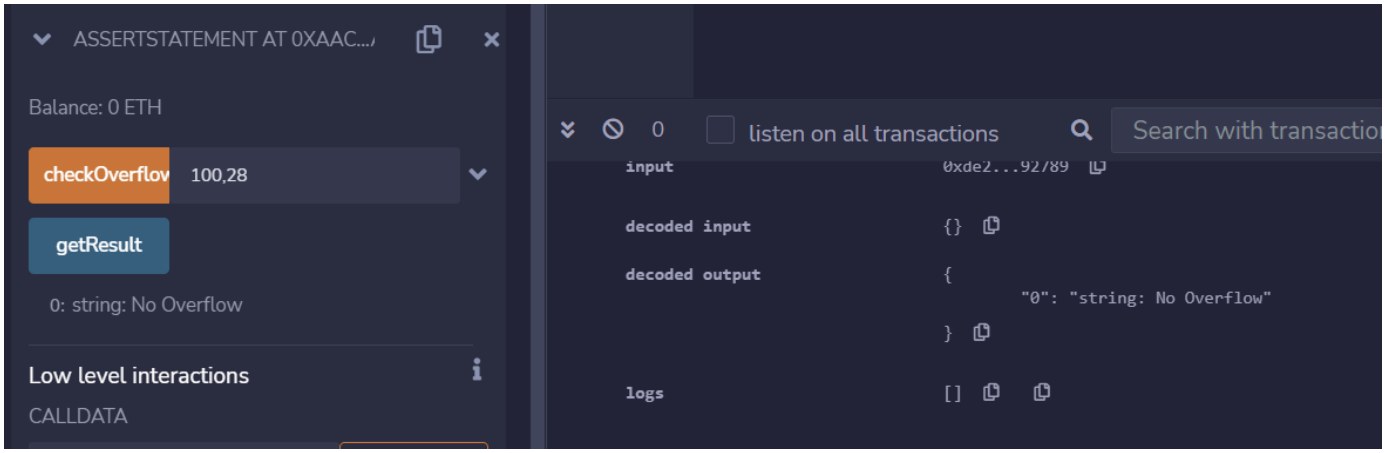
```
pragma solidity ^0.5.0;



contract assertStatement {

	bool result;

	function checkOverflow(uint _num1, uint _num2) public {
		uint sum = _num1 + _num2;
		assert(sum<=255);
		result = true;
	}


	function getResult() public view returns(string memory){
		if(result == true){
			return "No Overflow";
		}
		else{
			return "Overflow exist";
		}
	}
}
```

OUTPUT:

Overflow:



No Overflow:

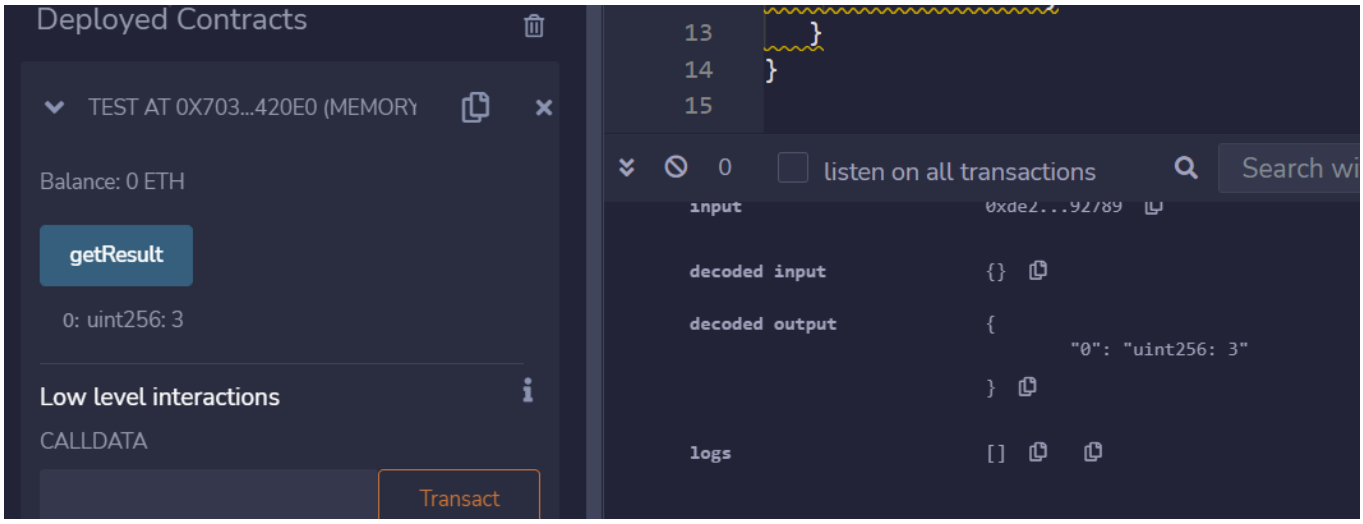


AIM: 5. Interfaces

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: 8

AIM: Implement and demonstrate the use of the following in Solidity

1. **Libraries**
2. **Assembly**
3. **Events**
4. **Error handling**

AIM: 1. Libraries

CODE:

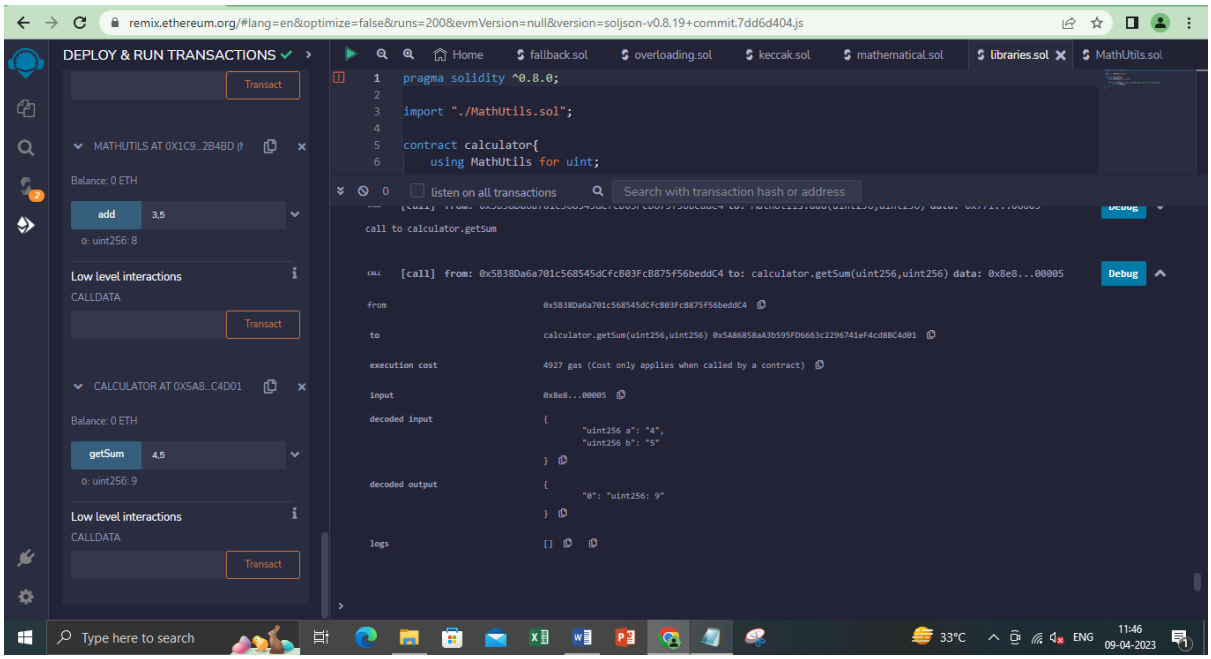
Libraries.sol:

```
pragma solidity ^0.8.0;
import "./MathUtils.sol";
contract calculator{
   using MathUtils for uint;

   function getSum(uint a, uint b) public pure returns(uint){
      return a.add(b);
   }
}
```

MathUtils.sol:

```
pragma solidity ^0.8.0;
library MathUtils{
   function add(uint x, uint y) public pure returns(uint){
      return x+y;
   }
}
```

OUTPUT:

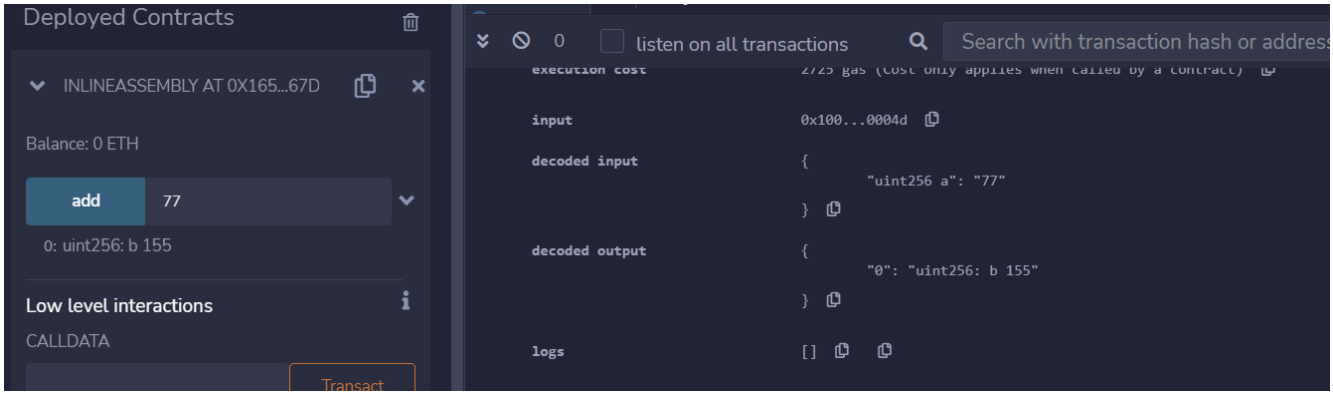

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.19+commit.7dd6d404.js
DEPLOY & RUN TRANSACTIONS
Transact
MATHUTILS AT 0X1C9...2B4BD
Balance: 0 ETH
add 3,5
0: uint256: 8
Low level interactions
CALLDATA
Transact
CALCULATOR AT 0X5A8...C4D01
Balance: 0 ETH
getSum 4,5
0: uint256: 9
Low level interactions
CALLDATA
Transact
Home
fallback.sol
overloading.sol
keccak.sol
mathematical.sol
libraries.sol
MathUtils.sol
pragma solidity ^0.8.0;
import "./MathUtils.sol";
contract calculator{
using MathUtils for uint;
listen on all transactions
Search with transaction hash or address
call to calculator.getSum
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: calculator.getSum(uint256,uint256) data: 0x8e8...00005
Debug
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to calculator.getSum(uint256,uint256) 0x5A86858aA3b595FD6663c2296741eF4cd8BC4d01
execution cost 4927 gas (Cost only applies when called by a contract)
input 0x8e8...00005
decoded input { "uint256 a": "4", "uint256 b": "5" }
decoded output { "0": "uint256: 9" }
logs []
Type here to search
33°C
ENG
11:46
09-04-2023

AIM: 2. Assembly

CODE:

```
pragma solidity ^0.8.0;
contract InlineAssembly {
    function add(uint a) public view returns (uint b) {
        assembly {
            let c := add(a, 56)
            mstore(0x80, c)
            {
                let d := add(sload(c), 22)
                b := d
            }
            b := add(b, c)
        }
    }
}
```

OUTPUT:

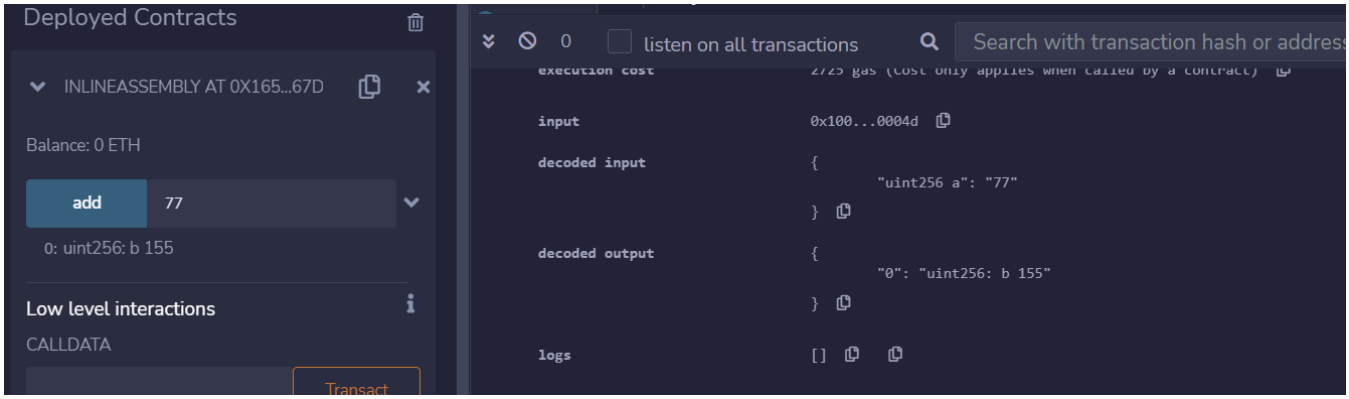


AIM: 3. Events

CODE:

```
pragma solidity ^0.5.0;
contract eventExample {
        uint256 public value = 0;
        event Increment(address owner);
        function getValue(uint _a, uint _b) public {
                emit Increment(msg.sender);
                value = _a + _b;
        }
}
```

OUTPUT:



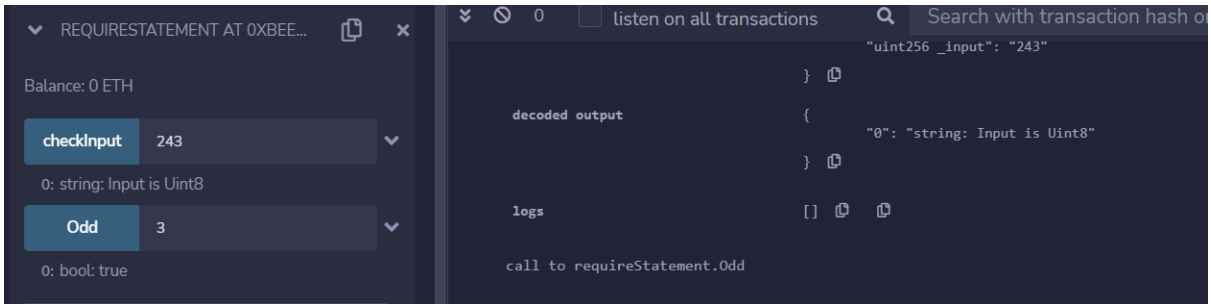
AIM: 3. Error Handling

a) Require

CODE:

```
pragma solidity ^0.5.0;
contract requireStatement {
    function checkInput(uint _input) public view returns(string memory){
        require(_input >= 0, "invalid uint8");
        require(_input <= 255, "invalid uint8");
        return "Input is Uint8";
    }
    function Odd(uint _input) public view returns(bool){
        require(_input % 2 != 0);
        return true;
    }
}
```

OUTPUT:



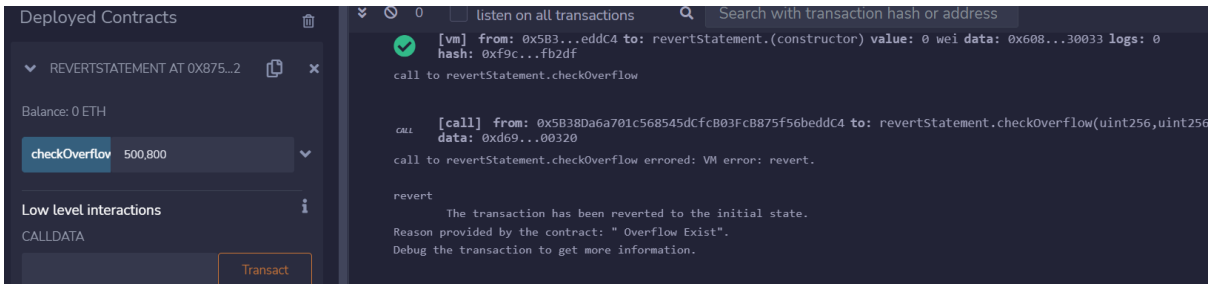
AIM: Revert

CODE:

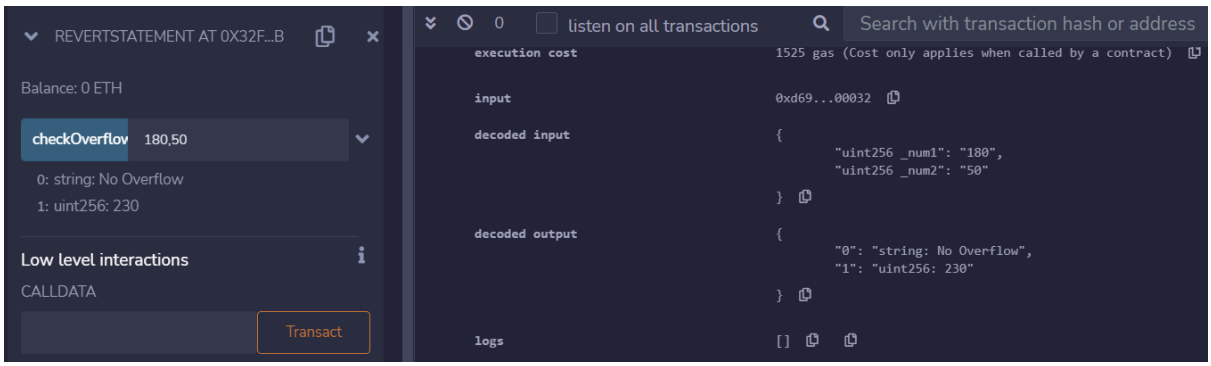
```
pragma solidity ^0.8.0;
contract revertStatement {
function checkOverflow(uint _num1, uint _num2) public view returns(string memory, uint){
        uint sum = _num1 + _num2;
        if(sum < 0 || sum > 255){
            revert(" Overflow Exist");
        }
        else{
            return ("No Overflow", sum);
        }
    }
}
```

OUTPUT:

Overflow



No Overflow



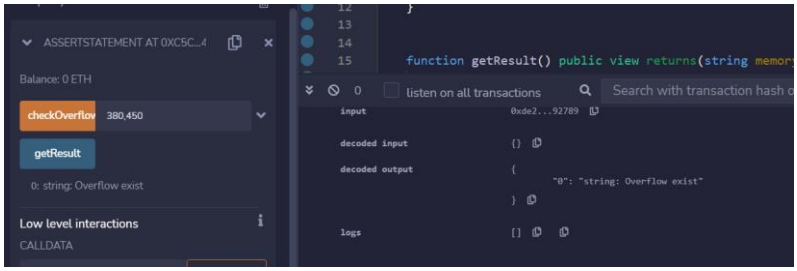
AIM: Assert

CODE:

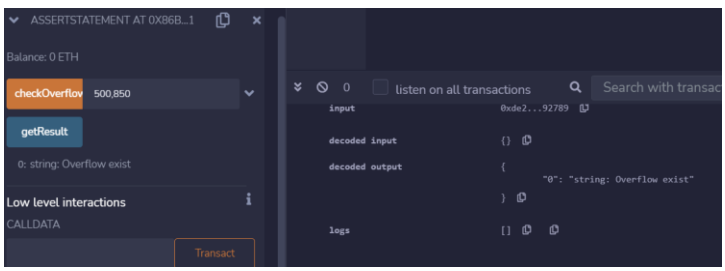
```
pragma solidity ^0.5.0;
contract assertStatement {
    bool result;
    function checkOverflow(uint _num1, uint _num2) public {
        uint sum = _num1 + _num2;
        assert(sum<=255);
        result = true;
    }
    function getResult() public view returns(string memory){
        if(result == true){
            return "No Overflow";
        }
        else{
            return "Overflow exist";
        }}}
```

OUTPUT:

Overflow



No Overflow



PRACTICAL No: 9

AIM: Mist Brower Installation.

CODE:

Install geth:

https://geth.ethereum.org/downloads

Install Mist:

https://github.com/ethereum/mist/releases

OUTPUT: