PRACTICAL JOURNAL IN

Blockchain

Advance IOT

Deep Learning

SUBMITTED BY

Yash Santosh Kadam
ROLL NO : 2214519

IN PARTIAL FULLFILMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY PART-II

SEMESTER IV

ACADEMIC YEAR 2022-2023

PARLE TILAK VIDYALAYA ASSOCIATION'S

MULUND COLLEGE OF COMMERCE (AUTONOMOUS)

(AFFILIATED TO UNIVERSITY OF MUMBAI)

NAAC RE-ACCREDITED A GRADE – III CYCLE

MULUND WEST, MUMBAI 400080 MAHARASHTRA, INDIA

2022-23

Blockchain

Practical No: 1

Aim: Write the following programs for Blockchain in Python:

1a) A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it.

Code:

```
import hashlib import random import string import json
import binascii import numpy as np import pandas as pd
import pylab as pl import logging import datetime import
collections import Crypto import Crypto.Random from
Crypto.PublicKey import RSA from Crypto.Signature
import PKCS1_v1_5 import binascii

class Client:    def __init__(self):
      random = Crypto.Random.new().read       self._private_key =
RSA.generate(1024, random)       self._public_key = self._private_key.publickey()
self._signer = PKCS1_v1_5.new(self._private_key)
   @property    def identity(self):       return binascii.hexlify(self._public_key.exportKey(format='DER')).
decode('ascii') Dinesh = Client()
print (Dinesh.identity)
```

Output:

1b) A transaction class to send and receive money and test it.

Code:

```
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':  se  lf.recipient,'value':  self.value,'time'  :
self.time})    def sign_transaction(self):
      private_key = self.sender._private_key       signer = PKCS1_v1_5.new(private_key)
h = SHA.new(str(self.to_dict()).encode('utf8'))       return
binascii.hexlify(signer.sign(h)).decode('ascii')
```

```
Dinesh = Client()
Ramesh = Client()
 t = Transaction(Dinesh,Ramesh.identity,5.0)
 signature = t.sign_transaction() print (signature)
```

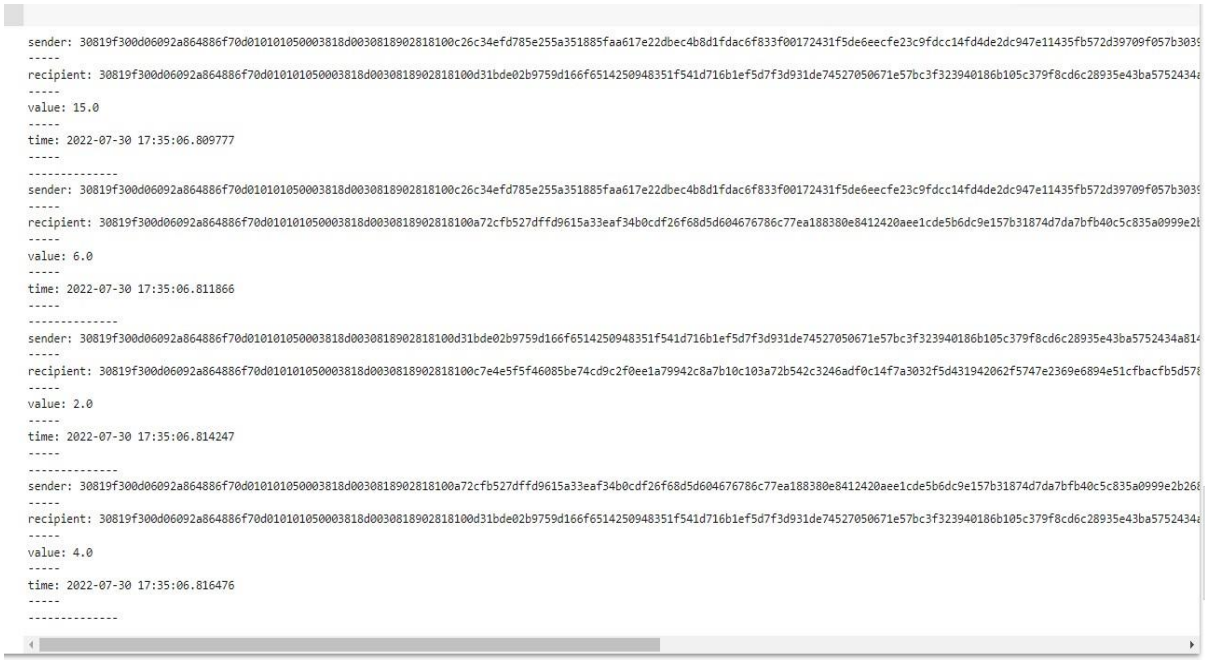
Output:

1c) Create Multiple Transaction and display them.

Code:

```
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 ('-----')
   transactions = []
Dinesh = Client()
Ramesh = Client()
Seema = Client()
Vijay = Client()
 t1 = Transaction(Dinesh, Ramesh.identity, 15.0) t1.sign_transaction()
transactions.append(t1)
 t2 = Transaction(Dinesh, Seema.identity, 6.0) t2.sign_transaction()
transactions.append(t2)
 t3 = Transaction(Ramesh, Vijay.identity, 2.0) t3.sign_transaction()
transactions.append(t3)
   t4 = Transaction(Seema, Ramesh.identity, 4.0)
t4.sign_transaction() transactions.append(t4)
 for transaction in transactions:   display_transaction
(transaction)   print ('--------------')
```

Output:



1d) Create a blockchain, a genesis block and execute it.

Code:

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

Dinesh = Client()
 t0 = Transaction (
"Genesis",
Dinesh.identity,
500.0
)  block0 = Block()

block0.previous_block_hash = None
Nonce = None
 block0.verified_transactions.append (t0)
 digest        =        hash        (block0)
last_block_hash = digest
```

```
TPCoins = []
 def dump_blockchain (self):
   print ("Number of blocks in the chain: " + str(len (self)))     for x in range (len(TPCoins)):
block_temp = TPCoins[x]        print ("block # " + str(x))     for transaction in
block_temp.verified_transactions:
       display_transaction (transaction)     print ('--------------')     print
('=====================================')

TPCoins.append (block0) dump_blockchain(TPCoins)
```

Output:

```
Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a4a79ac0645d8483c7c97fb6b67e1bf90f17d87d1f
-----
value: 500.0
-----
time: 2022-07-30 17:50:16.288802
-----
--------------
====================================
```

1e) Create a mining function and test it.

Code:

```
def sha256(message):
   return hashlib.sha256(message.encode('ascii')).hexdigest()
 def mine(message, difficulty=1):     assert
difficulty >= 1     prefix = '1' * difficulty     for i
in range(1000):
       digest = sha256(str(hash(message)) + str(i))        if digest.startswith(prefix):
          print ("after " + str(i) + " iterations found nonce: "+ digest
)    return digest

mine ("test message", 2)
```

Output:

```
after 238 iterations found nonce: 1124711db6185591392c6a06c24a3c2ebbaeca647fb8374824f939ada27c09e9
after 353 iterations found nonce: 11e0a4b57bb76496ecc6ab5a3c126165bc9dbbaf80f664e7e192a422360c3884
after 419 iterations found nonce: 11280dfd9ab05b3dbfa8699901537329414O8faa4a3b0832819b161f52321c08
after 511 iterations found nonce: 1150944bd7ea429acd052da390b148f27eb881686e0f8bfcdf77f833af878e2e
after 822 iterations found nonce: 110e61d41d94b48f6dfcb6f43f9ceafe2ab7168456dab088e05126d43d0366ef
after 924 iterations found nonce: 11147bf32bafeee4505b5cc40c3b227366d8a41ab3bd740e437c74400b7a8127
'6d80c4cf3cd4fb49aad25deced696c3a48e17974ecbfb1441e2128517f0152b2'
```

1f) Add blocks to the miner and dump the blockchain

Code:

```
last_transaction_index = 0
 block = Block() for i in range(3):     temp_transaction =
transactions[last_transaction_index]
#   validate   transaction   #   if   valid                block.verified_transactions.append
(temp_transaction)     last_transaction_index += 1
    block.previous_block_hash = last_block_hash     block.Nonce =
mine (block, 2)     digest = hash (block)     TPCoins.append (block)
   last_block_hash = digest


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

TPCoins.append (block) last_block_hash = digest

dump_blockchain(TPCoins)
```

Output:

```
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539cbb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539cbb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539cbb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
```

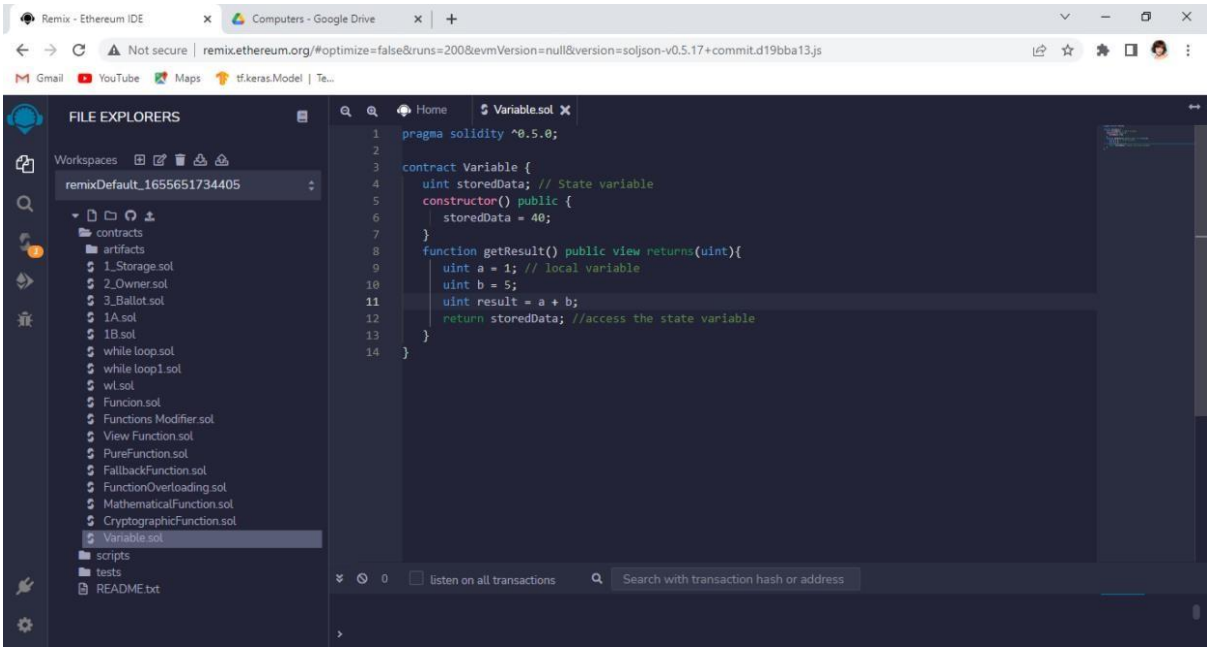
Practical No: 2

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

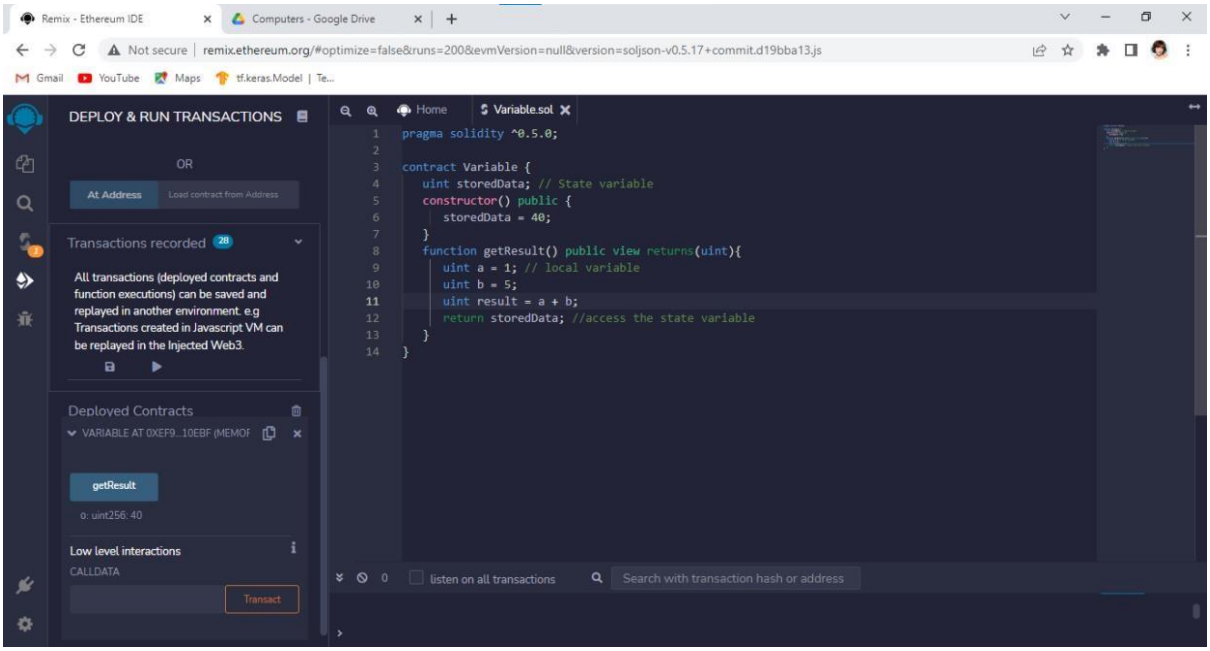
2a) Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. Variable

Variable

Code:



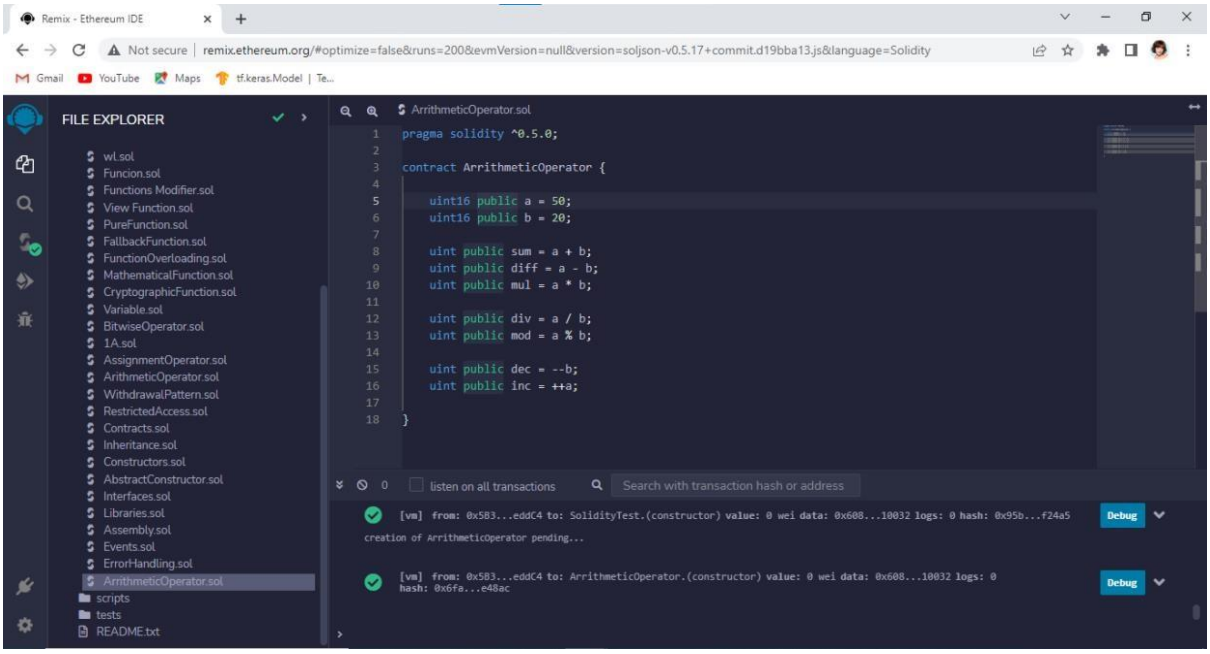
Output:



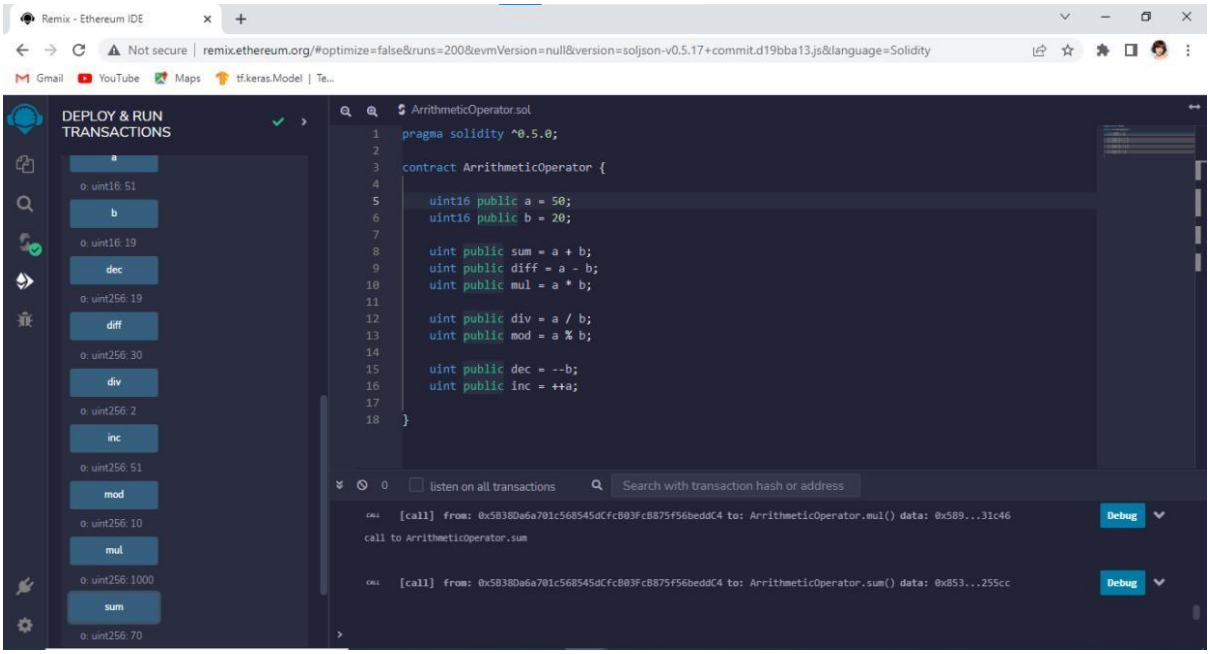
Operators:

Arithmetic Operator

Code:

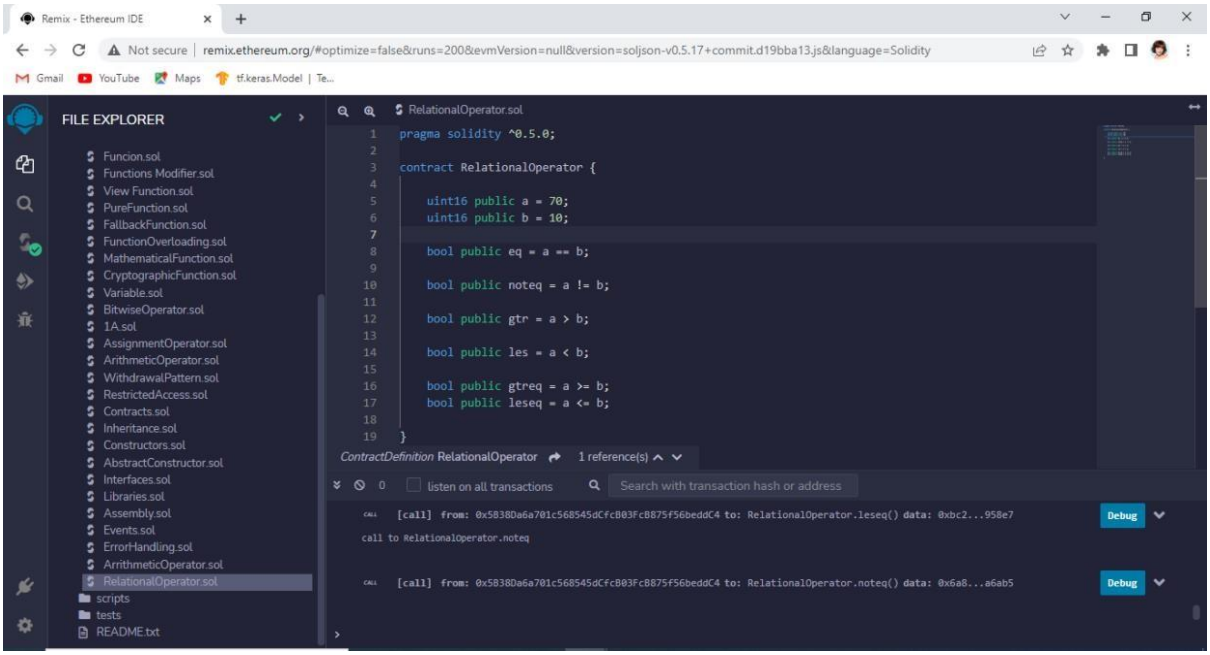

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity
Gmail YouTube Maps tf.keras.Model | Te...
FILE EXPLORER
wl.sol
Funcion.sol
Functions Modifier.sol
View Function.sol
PureFunction.sol
FallbackFunction.sol
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
Variable.sol
BitwiseOperator.sol
1A.sol
AssignmentOperator.sol
ArithmeticOperator.sol
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
Inheritance.sol
Constructors.sol
AbstractConstructor.sol
Interfaces.sol
Libraries.sol
Assembly.sol
Events.sol
ErrorHandling.sol
ArrithmeticOperator.sol
scripts
tests
README.txt
ArrithmeticOperator.sol
1 pragma solidity ^0.5.0;
2
3 contract ArrithmeticOperator {
4
5 uint16 public a = 50;
6 uint16 public b = 20;
7
8 uint public sum = a + b;
9 uint public diff = a - b;
10 uint public mul = a * b;
11
12 uint public div = a / b;
13 uint public mod = a % b;
14
15 uint public dec = --b;
16 uint public inc = ++a;
17
18 }
listen on all transactions
Search with transaction hash or address
[vm] from: 0x583...eddC4 to: SolidityTest.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x95b...f24a5
Debug
creation of ArrithmeticOperator pending...
[vm] from: 0x583...eddC4 to: ArrithmeticOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0x6fa...e48ac
Debug

Output:

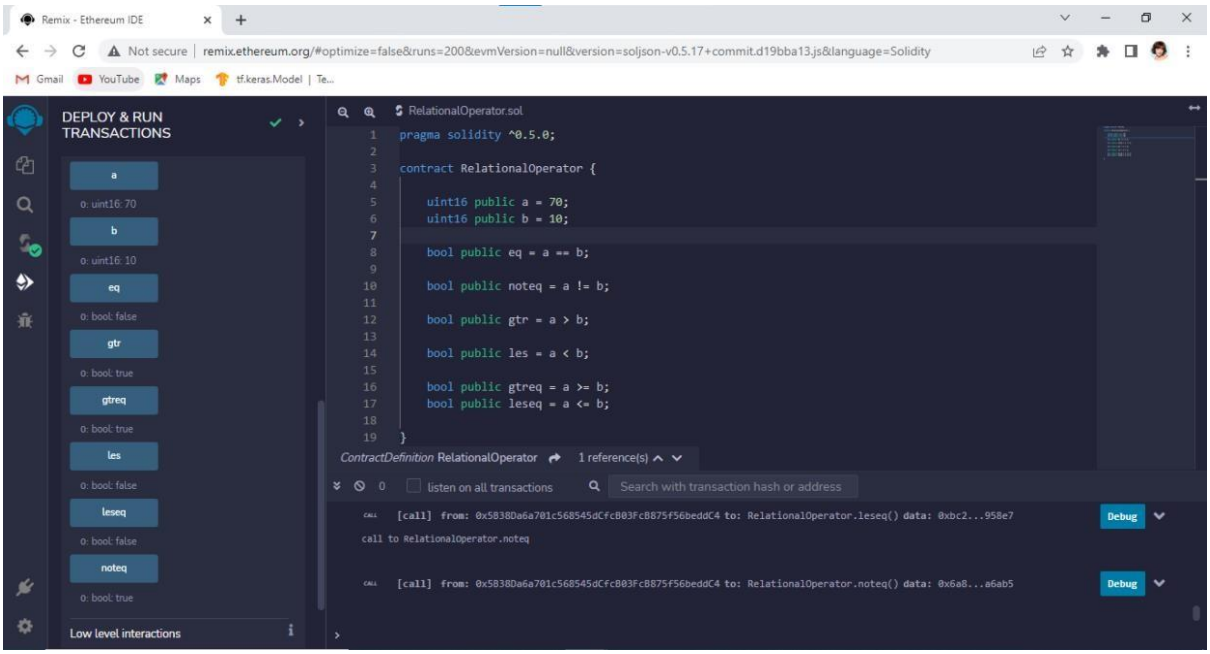

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity
Gmail YouTube Maps tf.keras.Model | Te...
DEPLOY & RUN TRANSACTIONS
a
0: uint16: 51
b
0: uint16: 19
dec
0: uint256: 19
diff
0: uint256: 30
div
0: uint256: 2
inc
0: uint256: 51
mod
0: uint256: 10
mul
0: uint256: 1000
sum
0: uint256: 70
ArrithmeticOperator.sol
1 pragma solidity ^0.5.0;
2
3 contract ArrithmeticOperator {
4
5 uint16 public a = 50;
6 uint16 public b = 20;
7
8 uint public sum = a + b;
9 uint public diff = a - b;
10 uint public mul = a * b;
11
12 uint public div = a / b;
13 uint public mod = a % b;
14
15 uint public dec = --b;
16 uint public inc = ++a;
17
18 }
listen on all transactions
Search with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCFcB03FcB875f56beddC4 to: ArrithmeticOperator.mul() data: 0x589...31c46
Debug
call to ArrithmeticOperator.sum
[call] from: 0x5B38Da6a701c568545dCFcB03FcB875f56beddC4 to: ArrithmeticOperator.sum() data: 0x853...255cc
Debug

Relational Operator

Code:

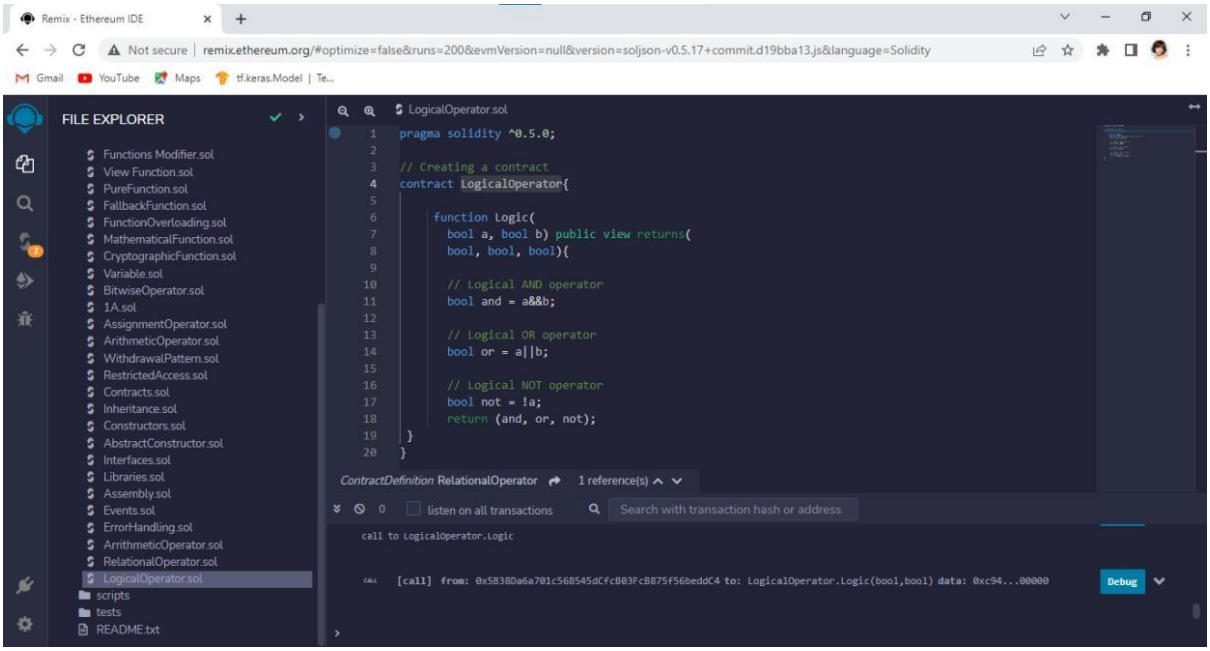


Output:

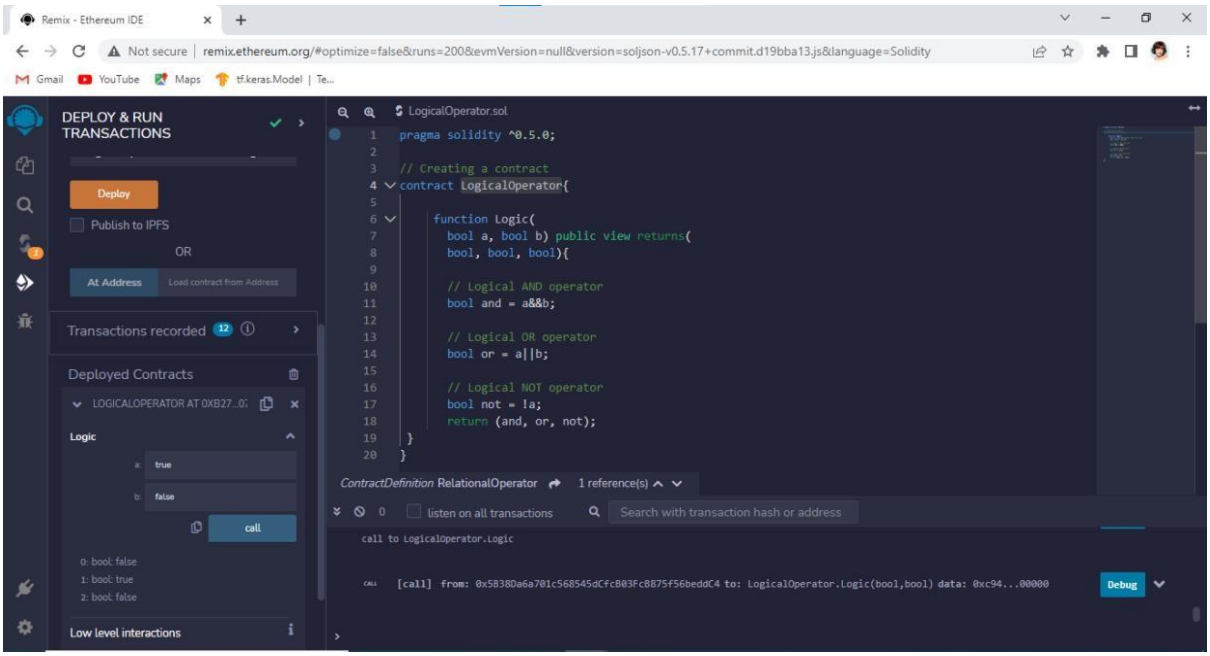


Logical Operator

Code:

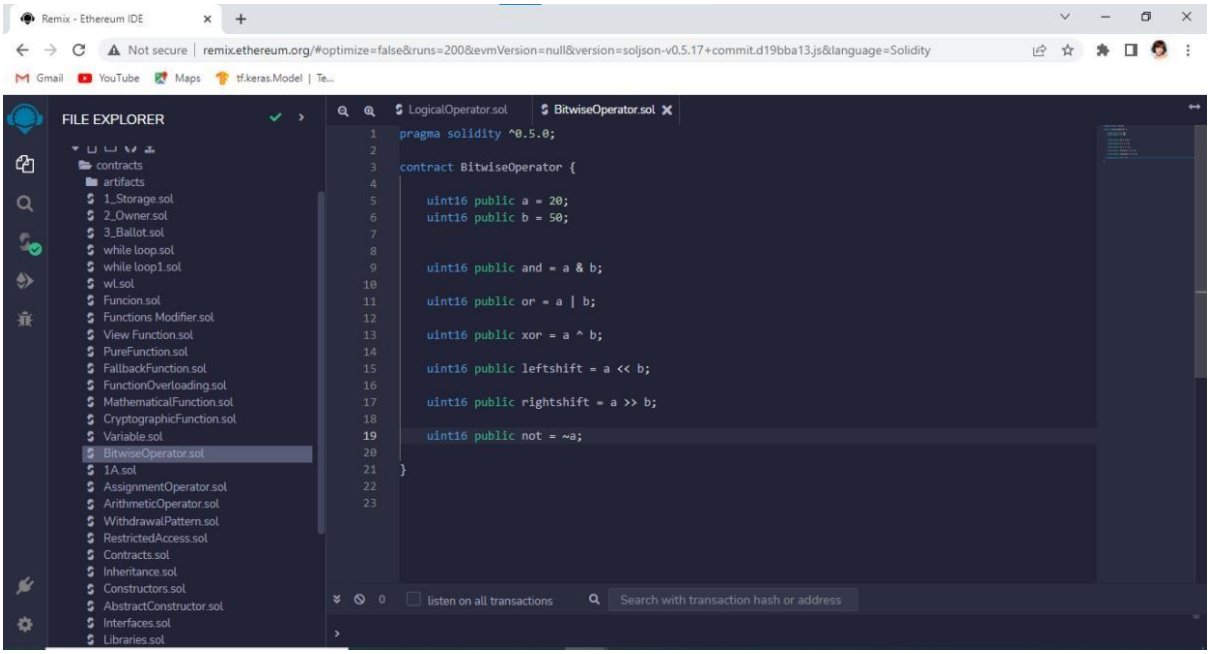


Output:

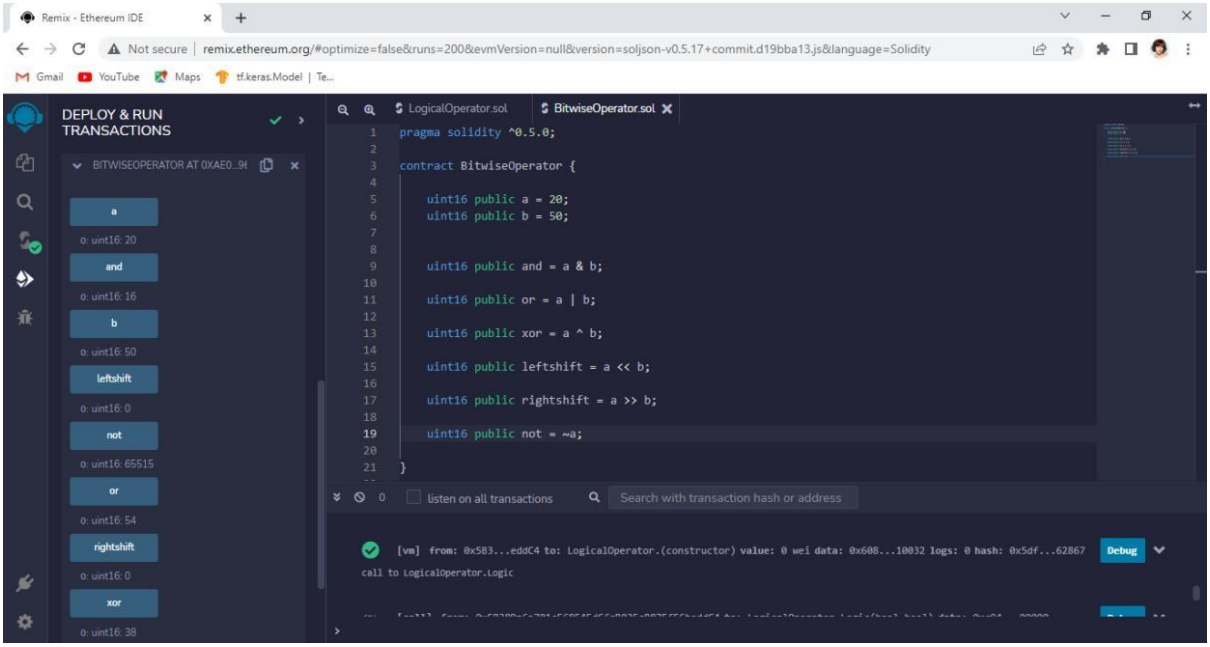


Bitwise Operator

Code:

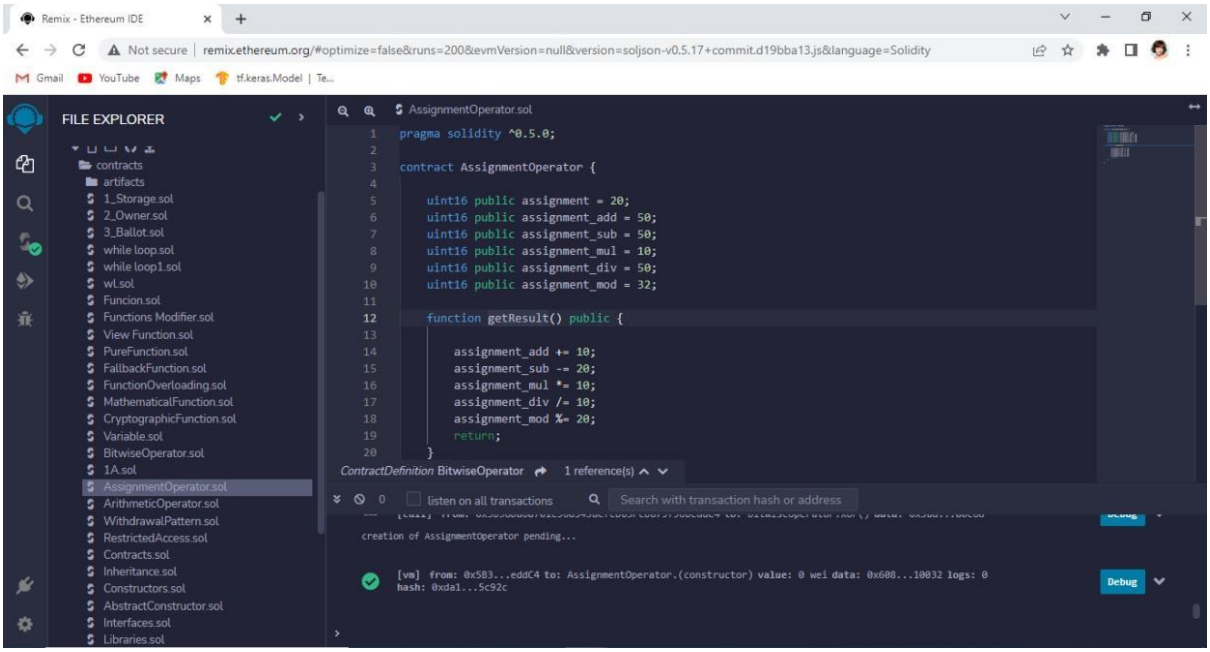


Output:

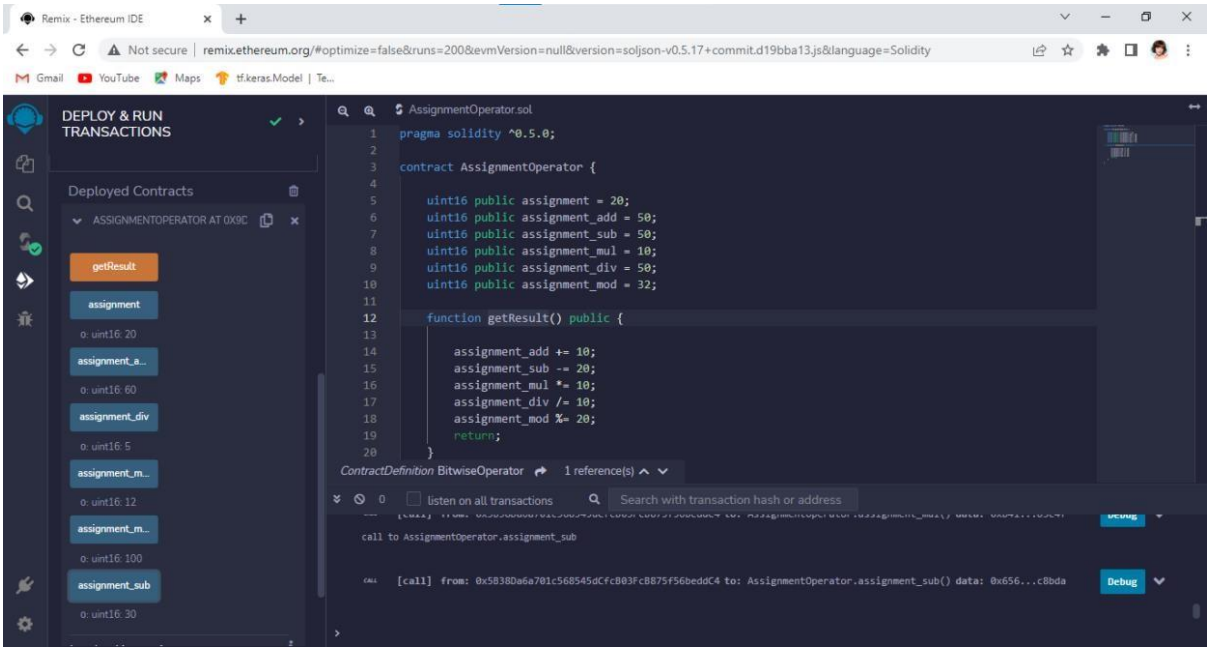


Assignment Operator

Code:



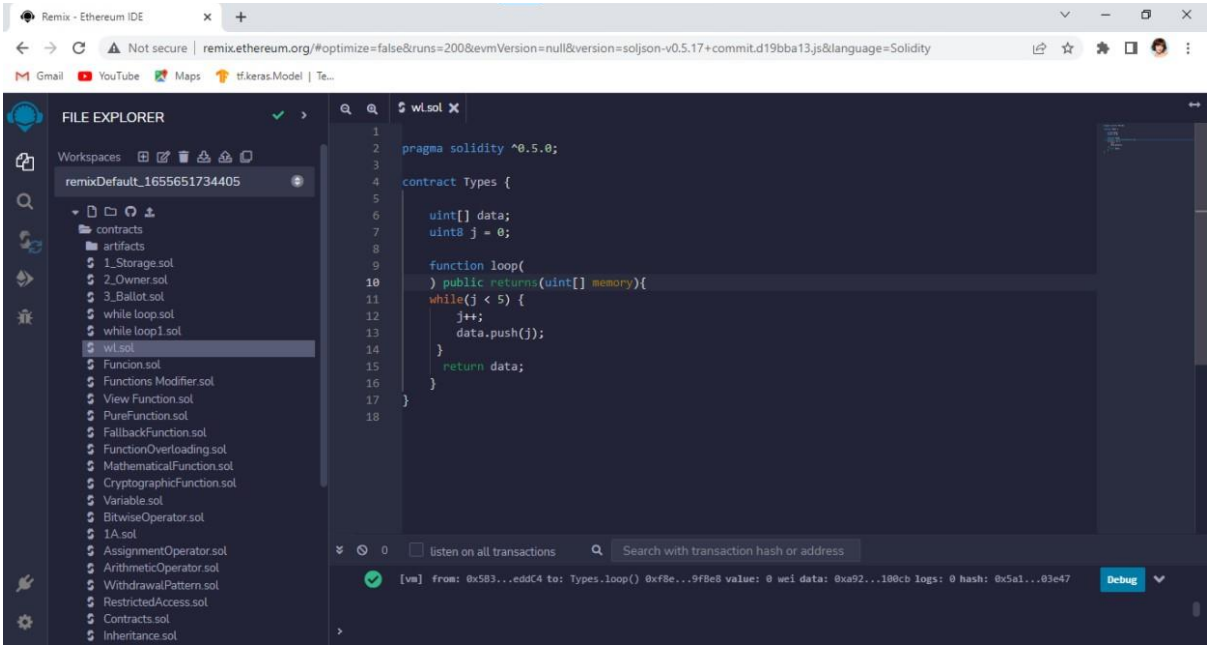
Output:



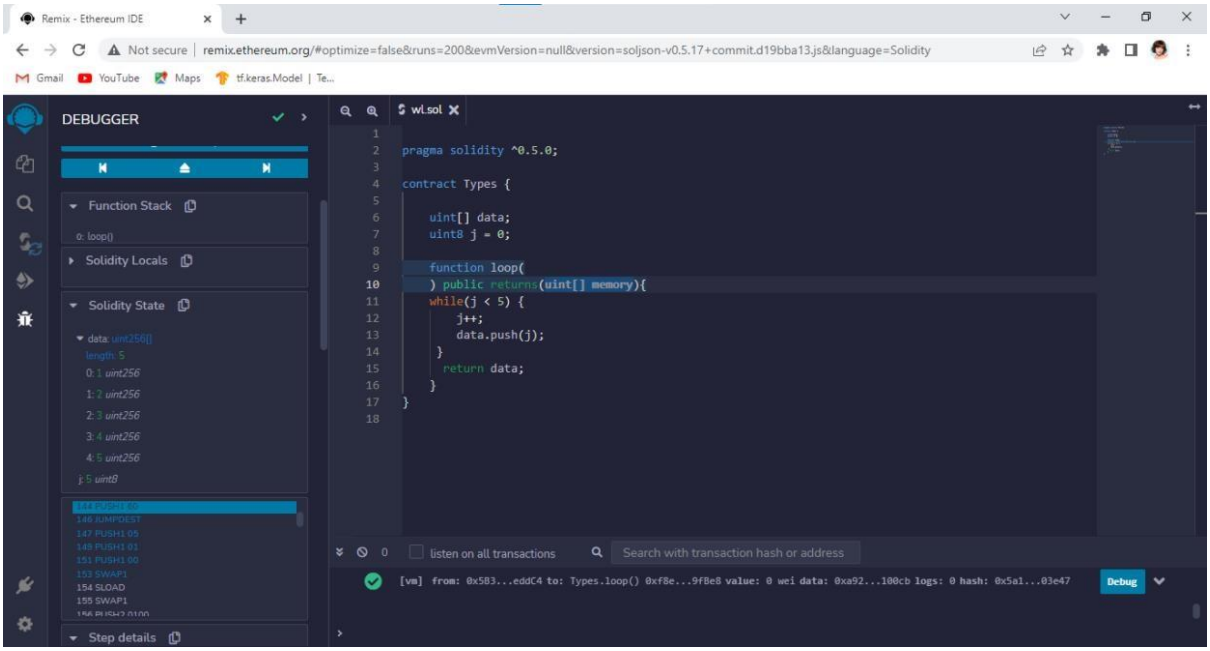
Loops

While Loop

Code:

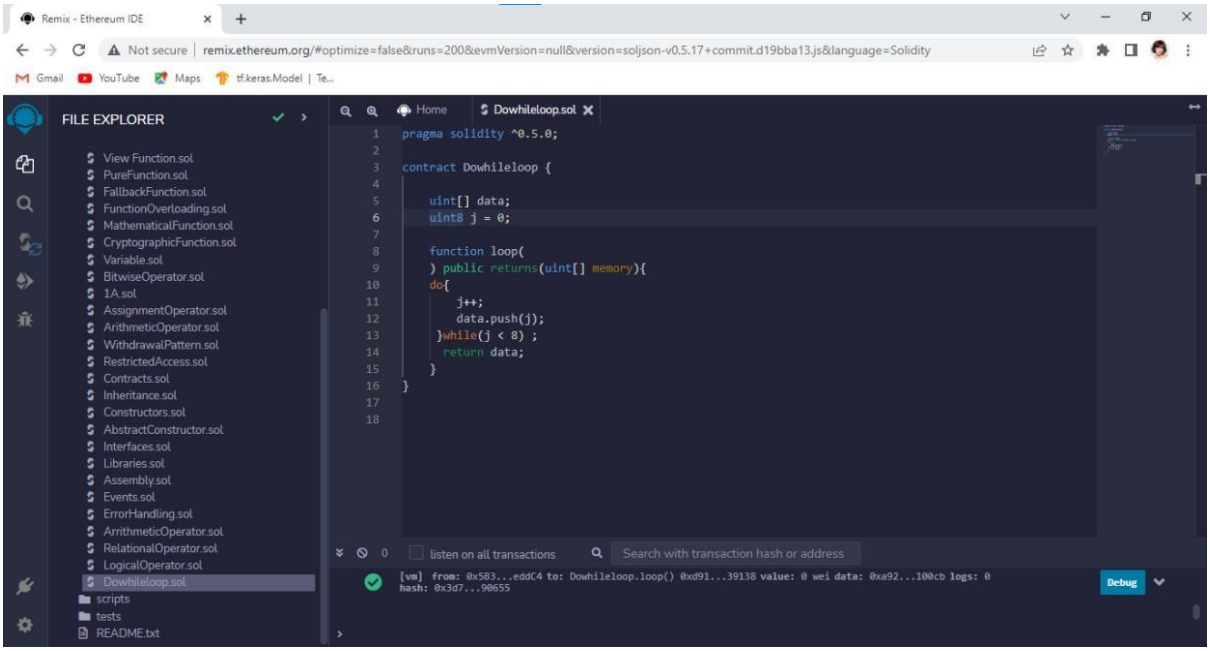


Output:

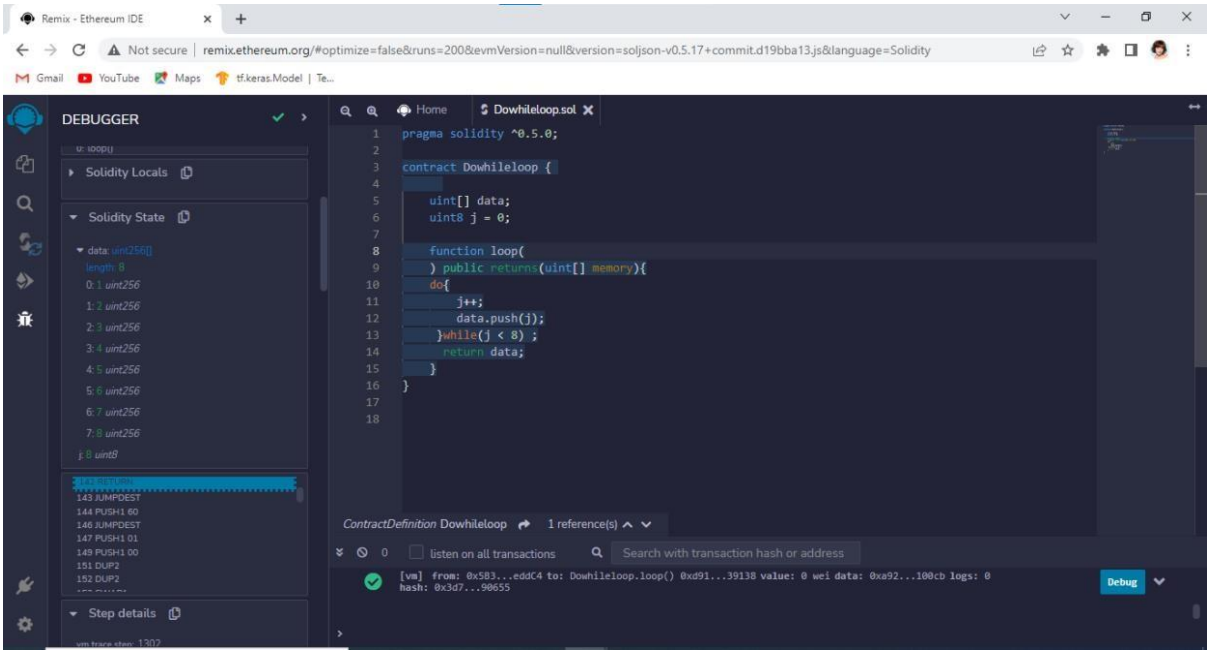


Dowhile loop

Code:

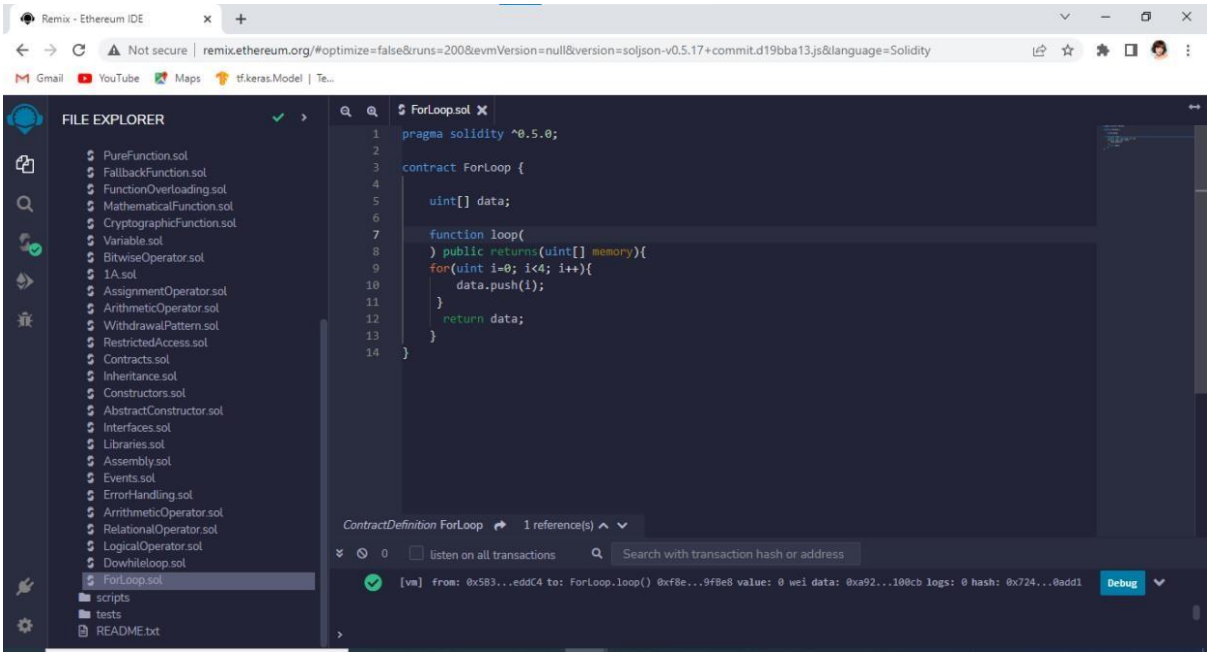


Output:

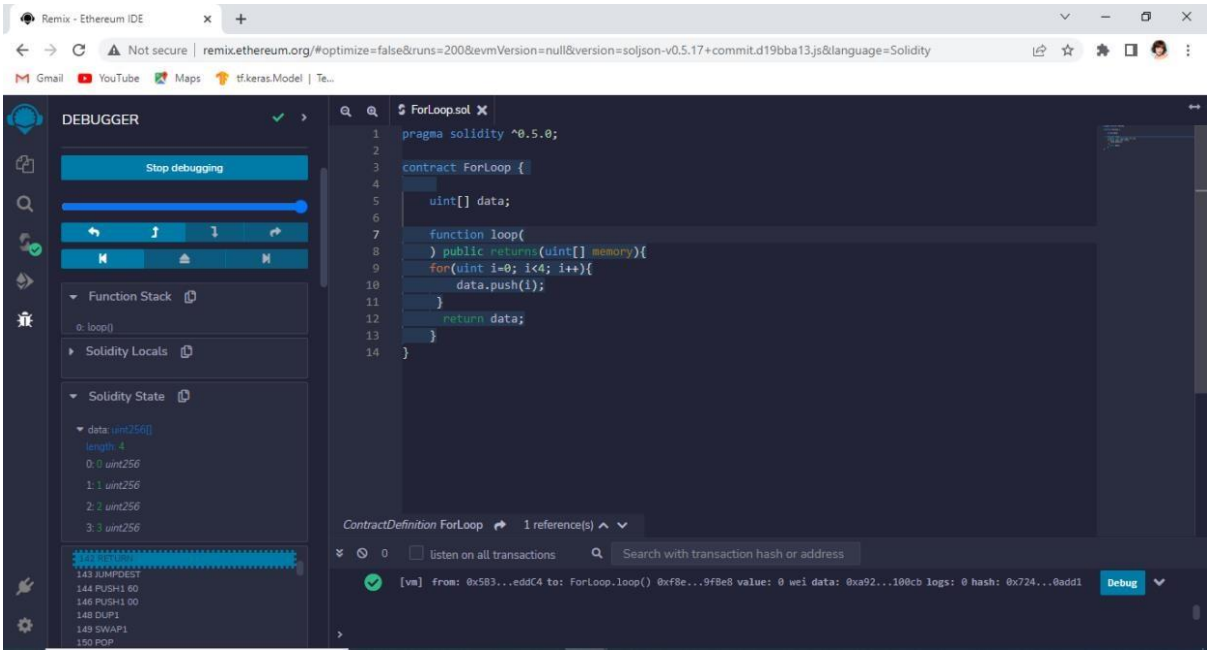


For loop

Code:



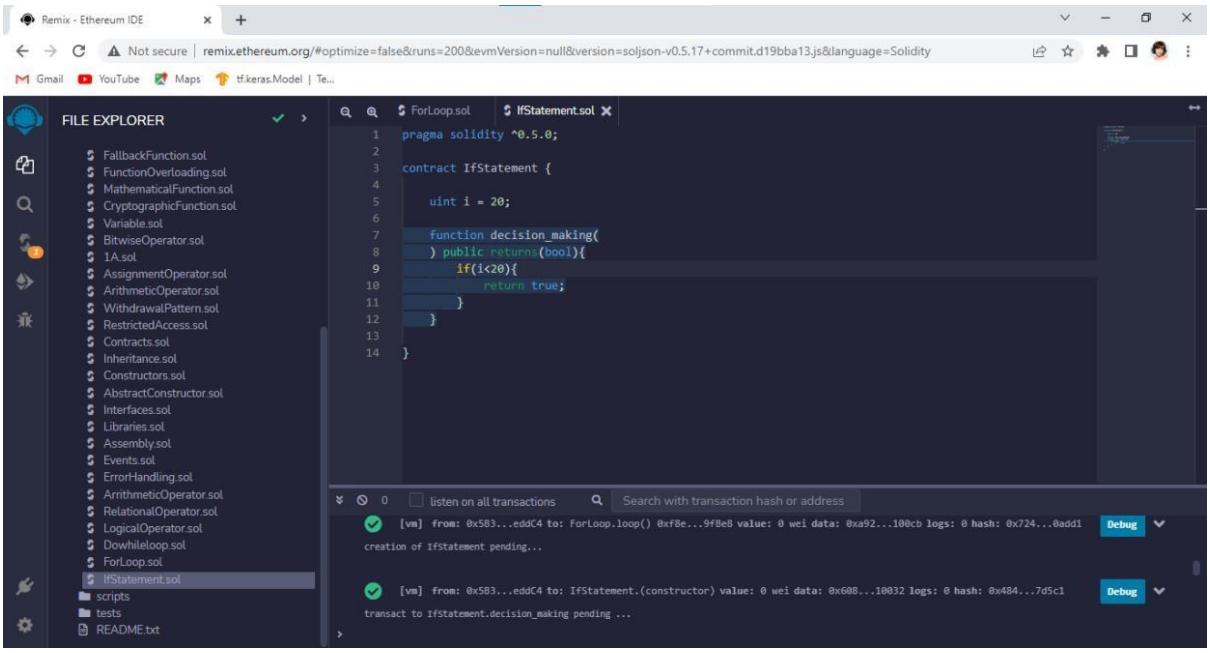
Output:



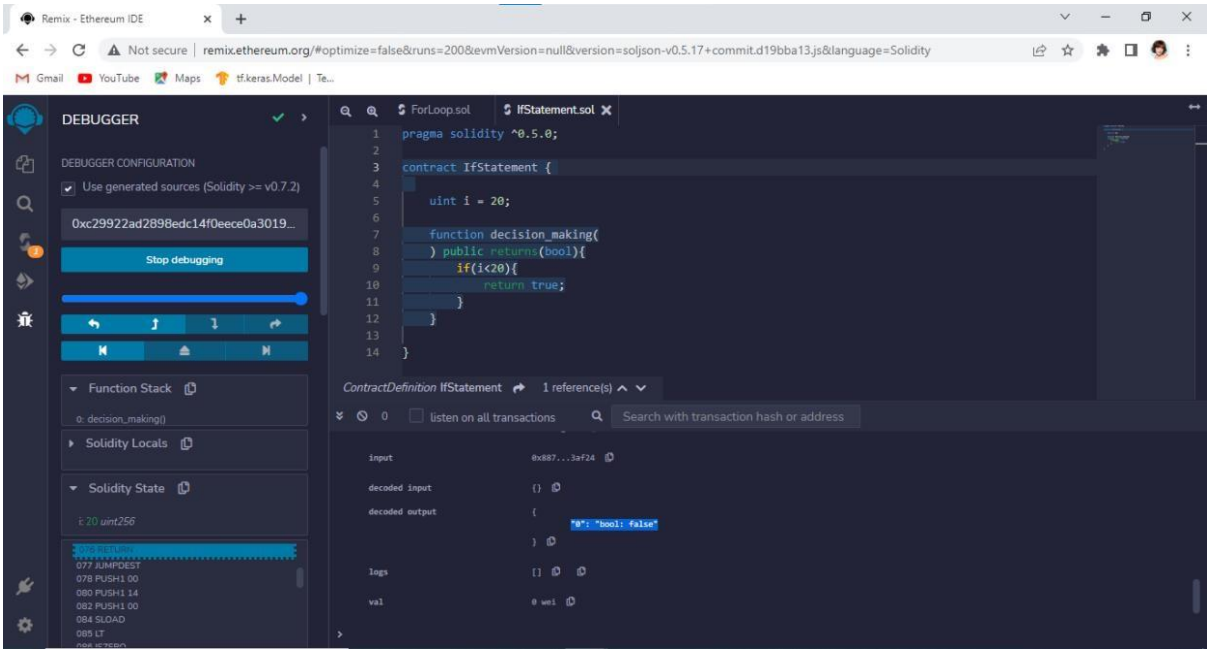
Decision making

If statement

Code:

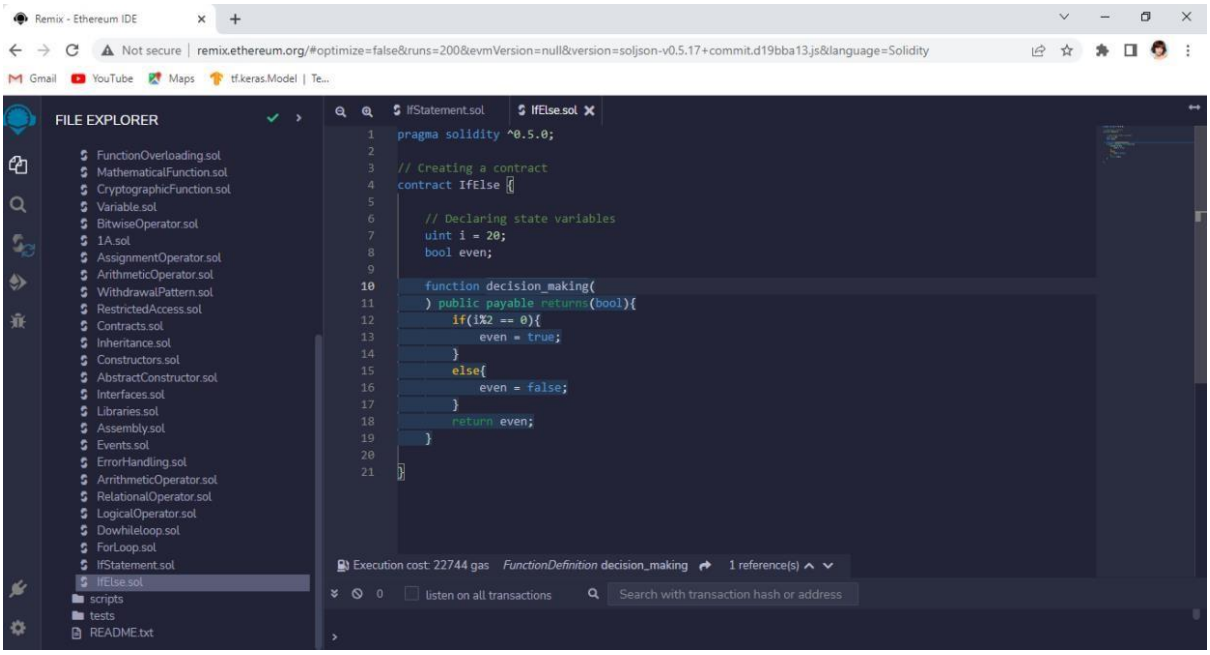


Output:

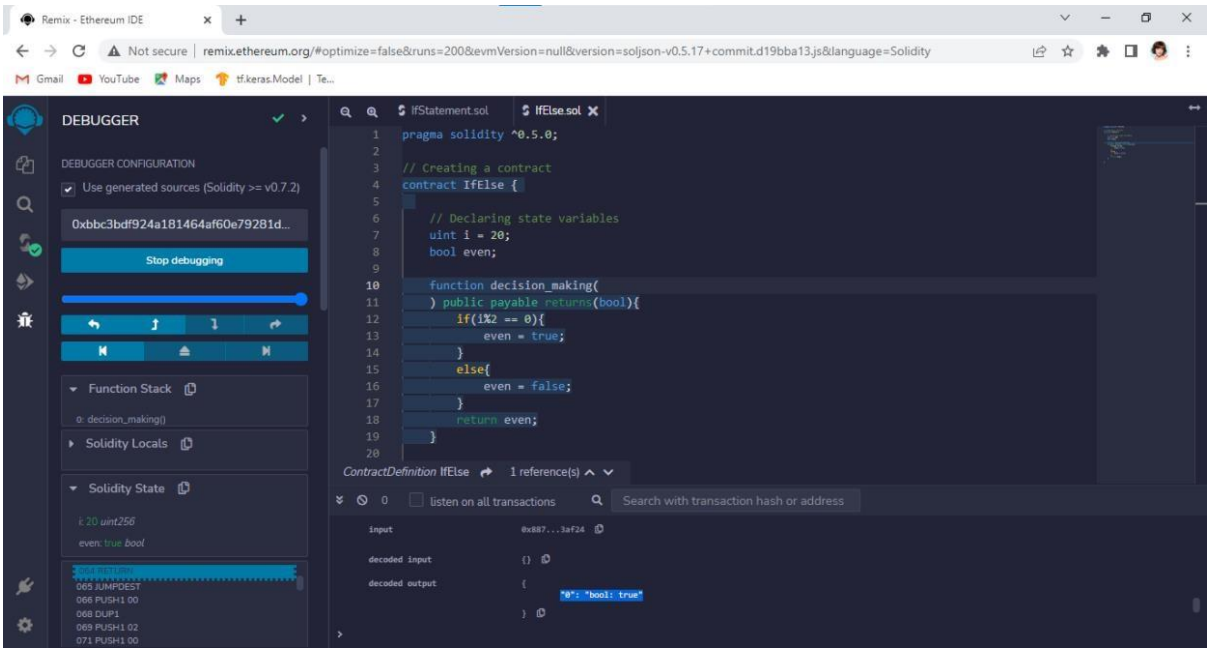


If…else statement

Code:

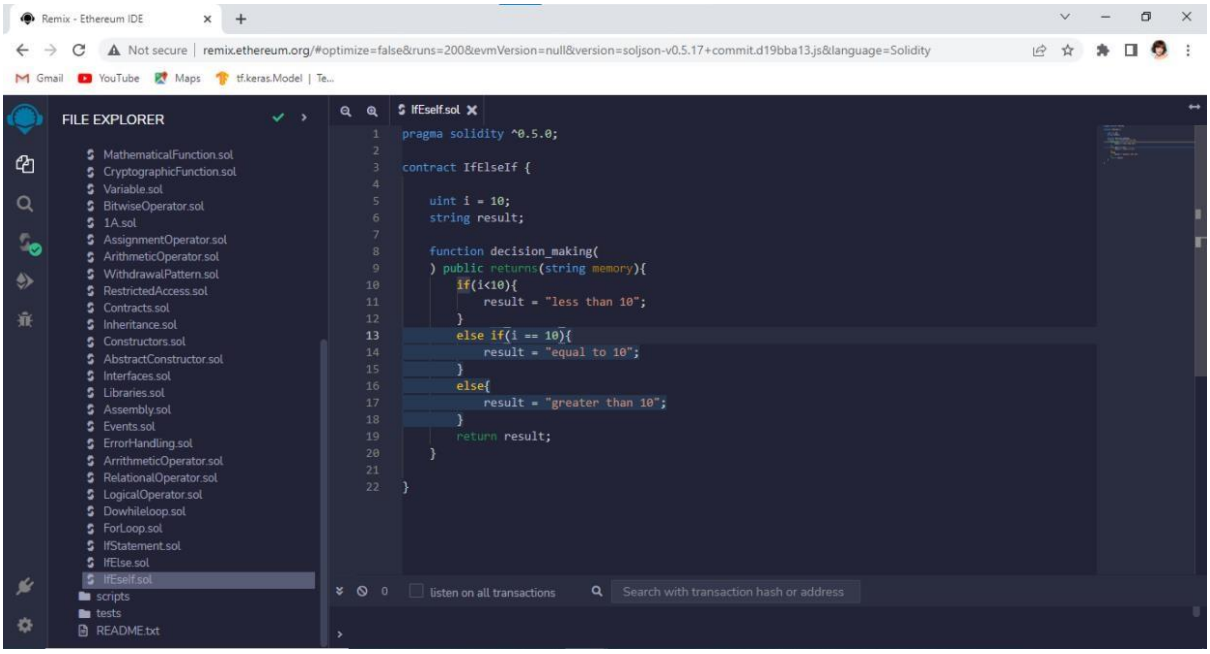


Output:

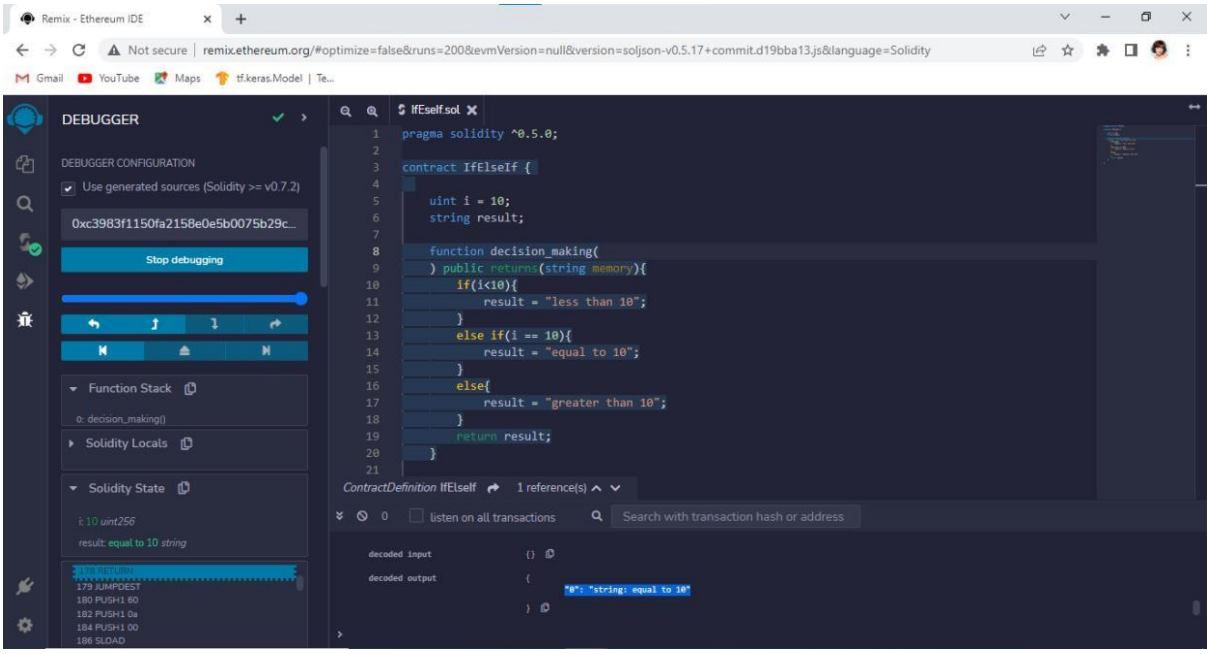


If…else if…else statement

Code:

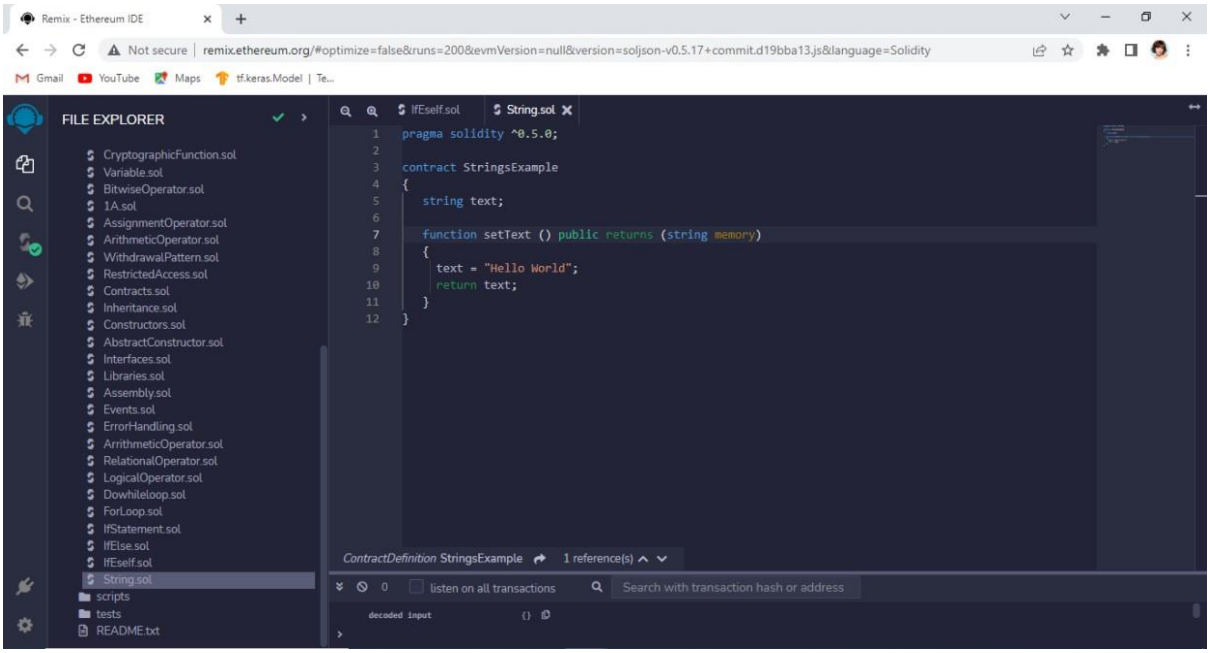


Output:

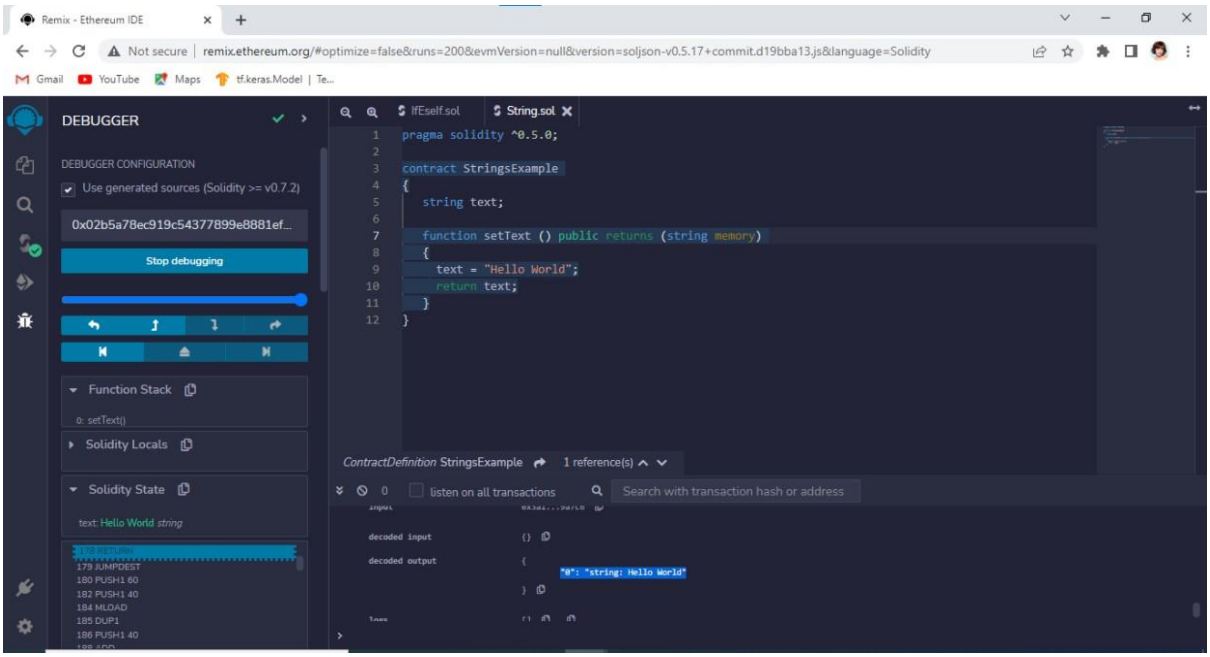


String

Code:

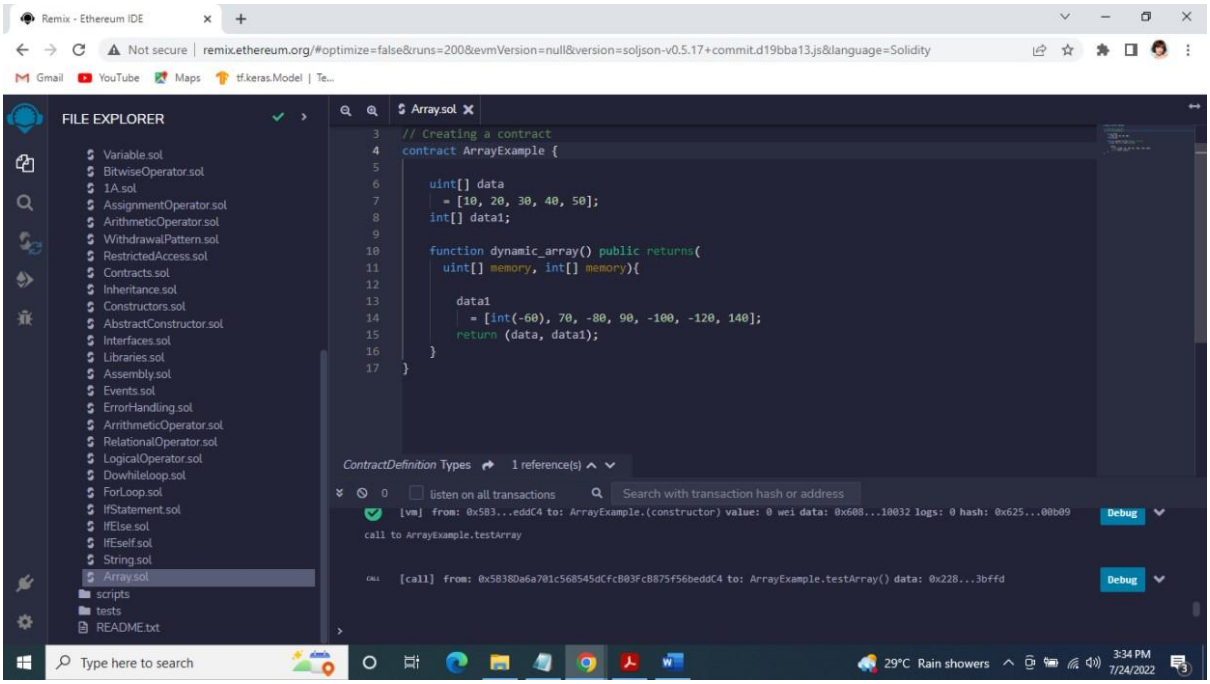


Output:

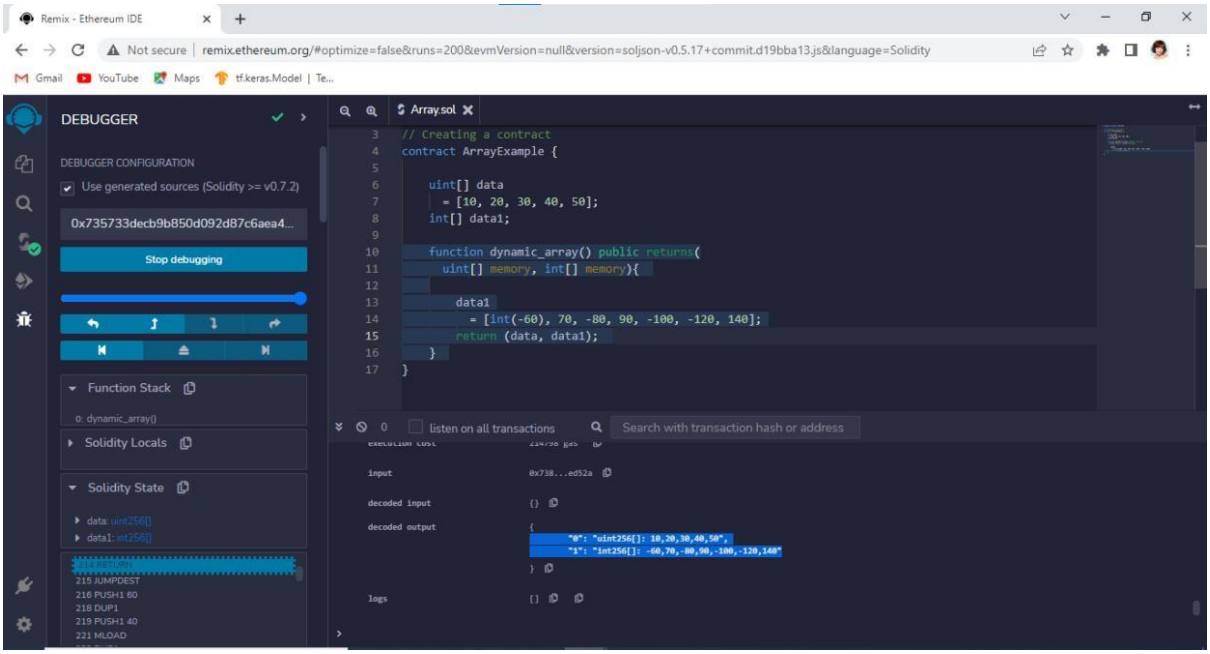


Array

Code:

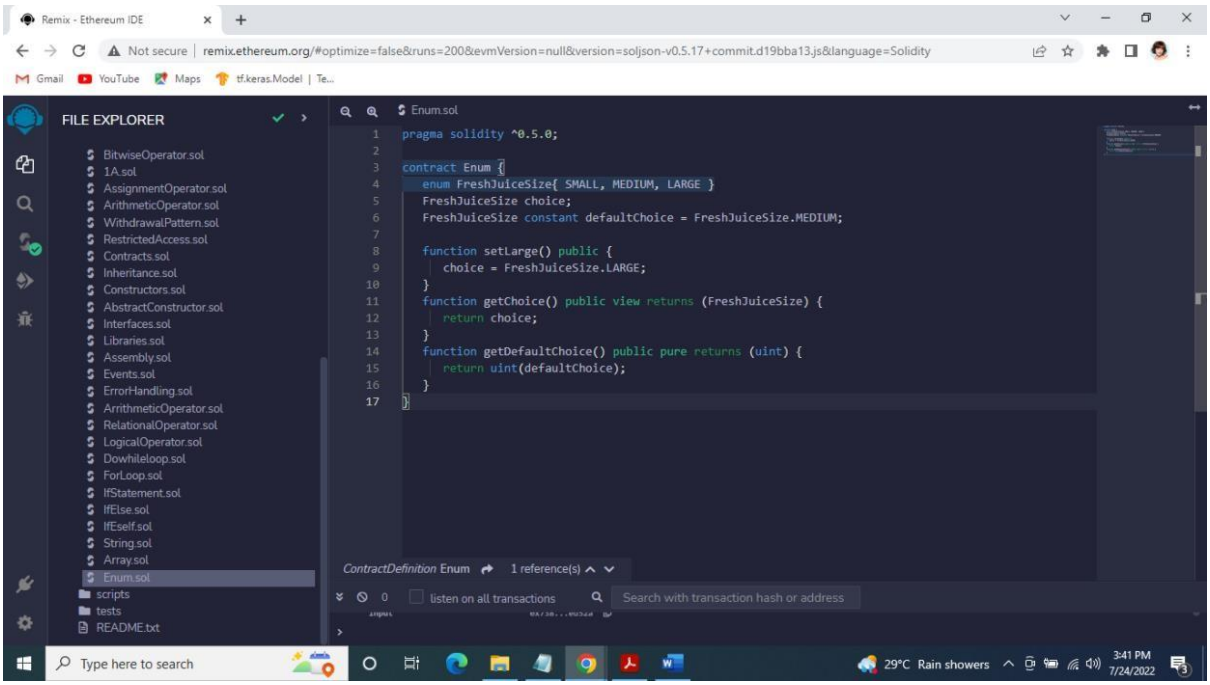


Output:

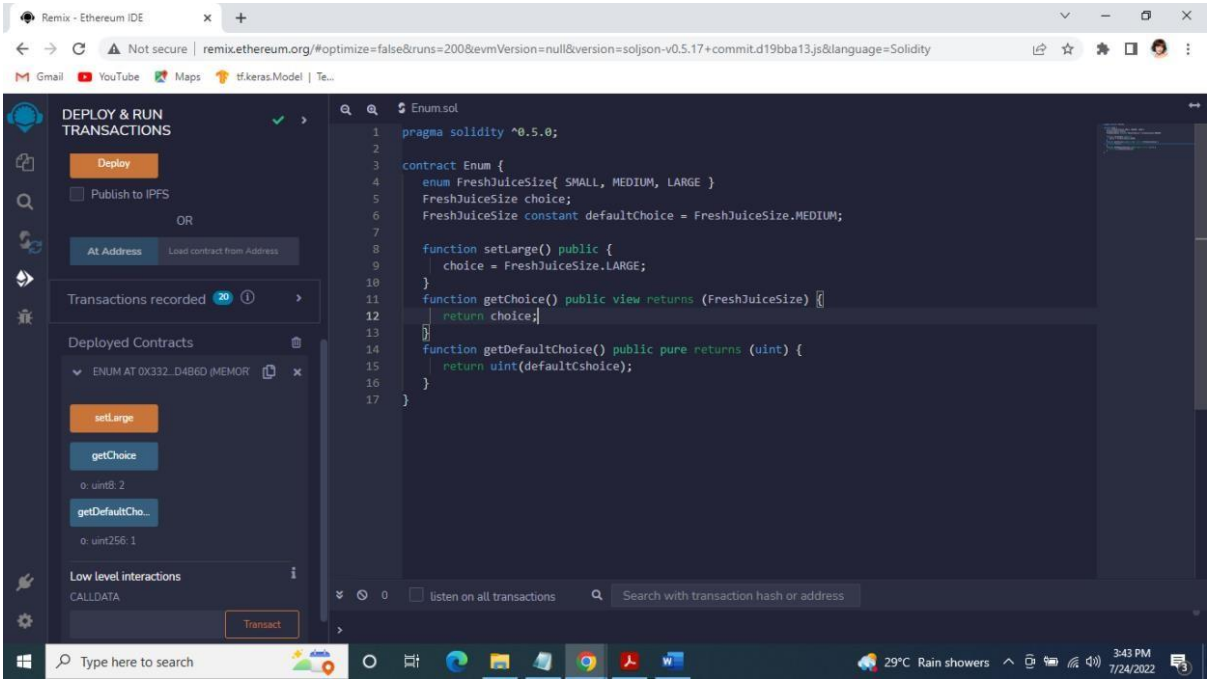


Enums

Code:

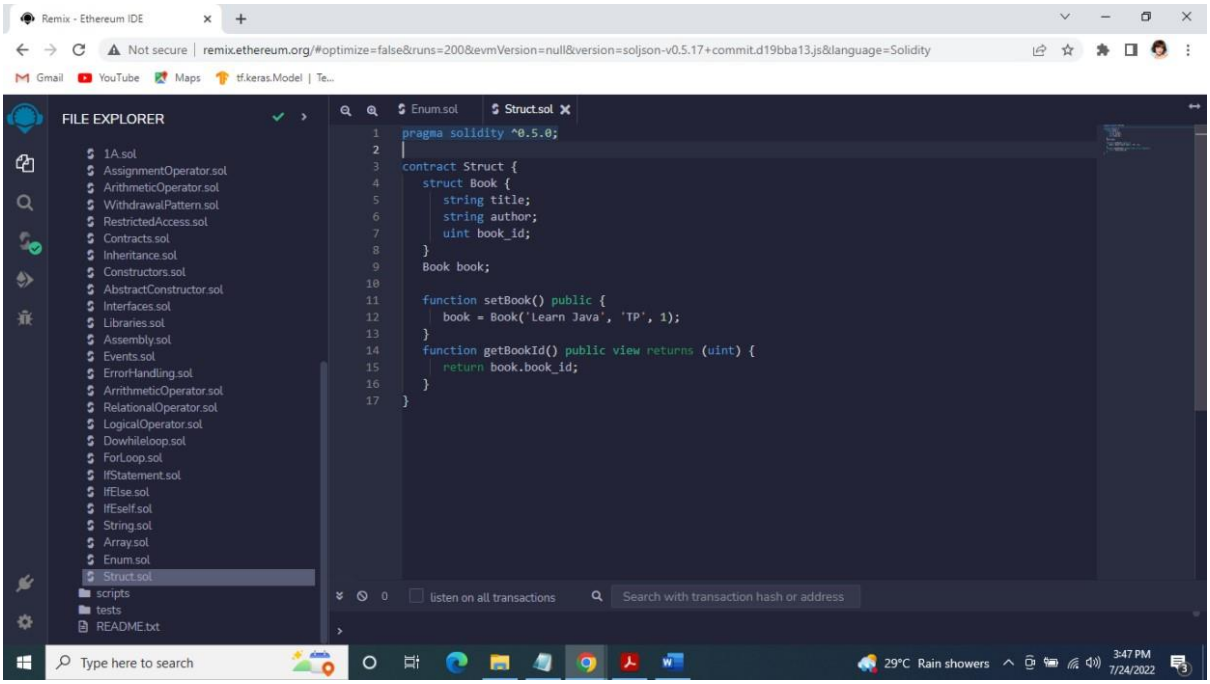


Output:

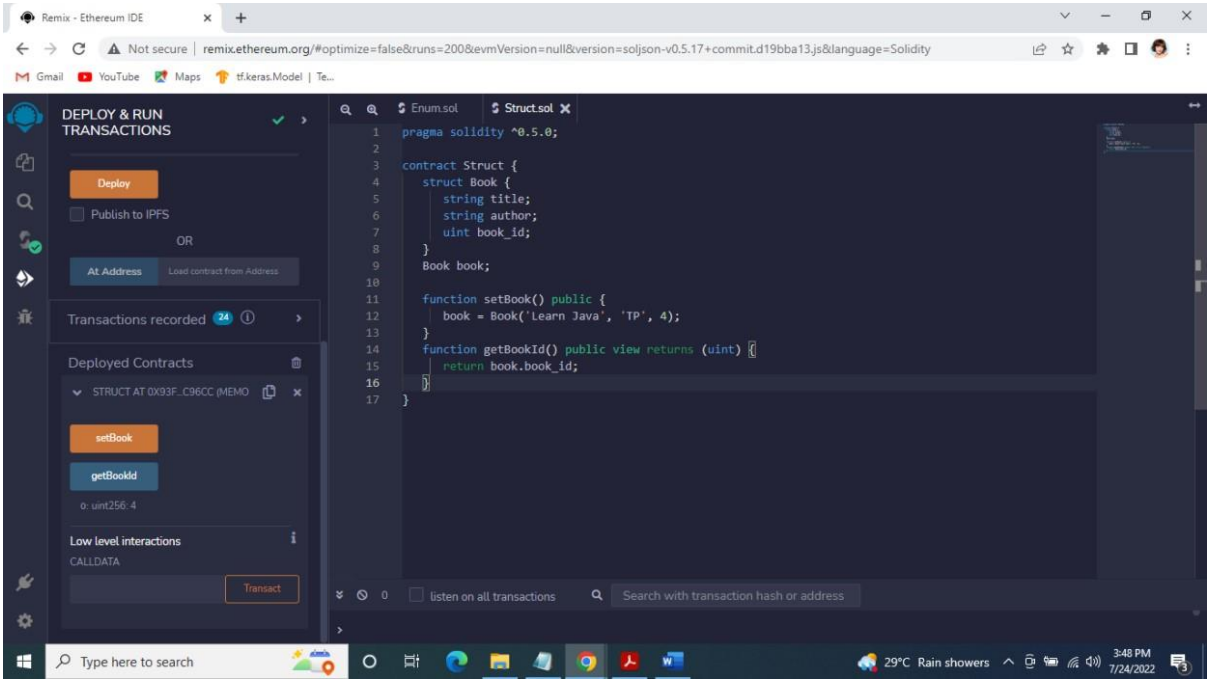


Struct

Code:

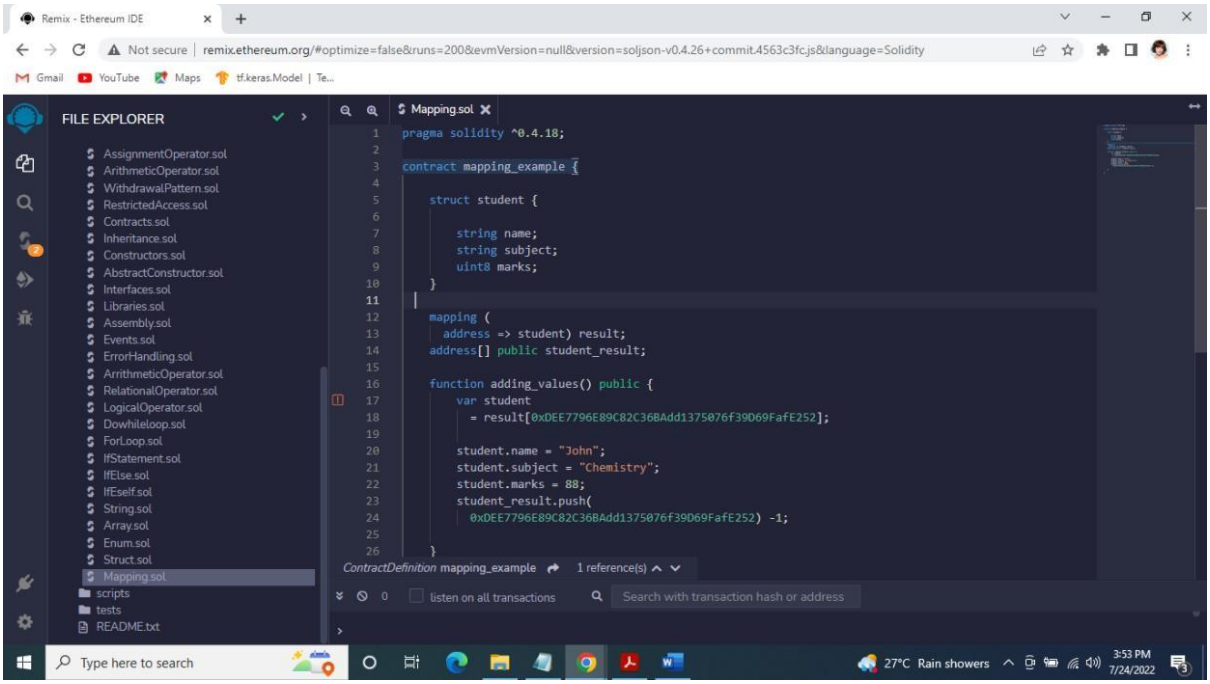


Output:

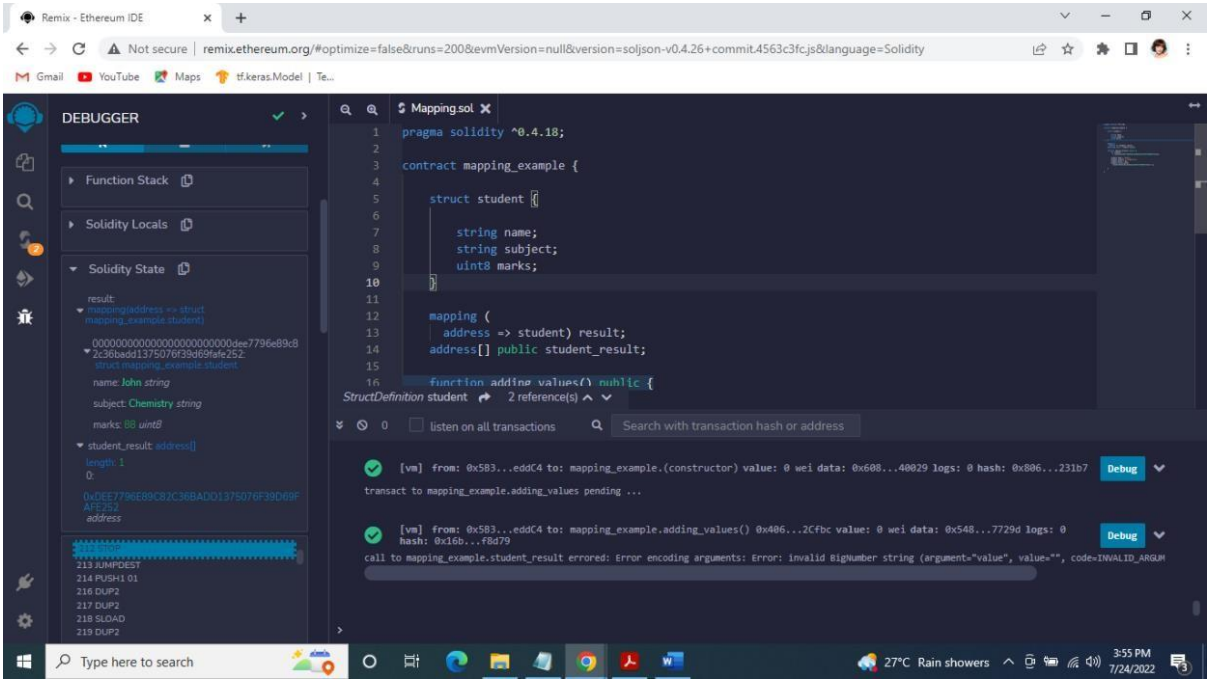


Mapping

Code:

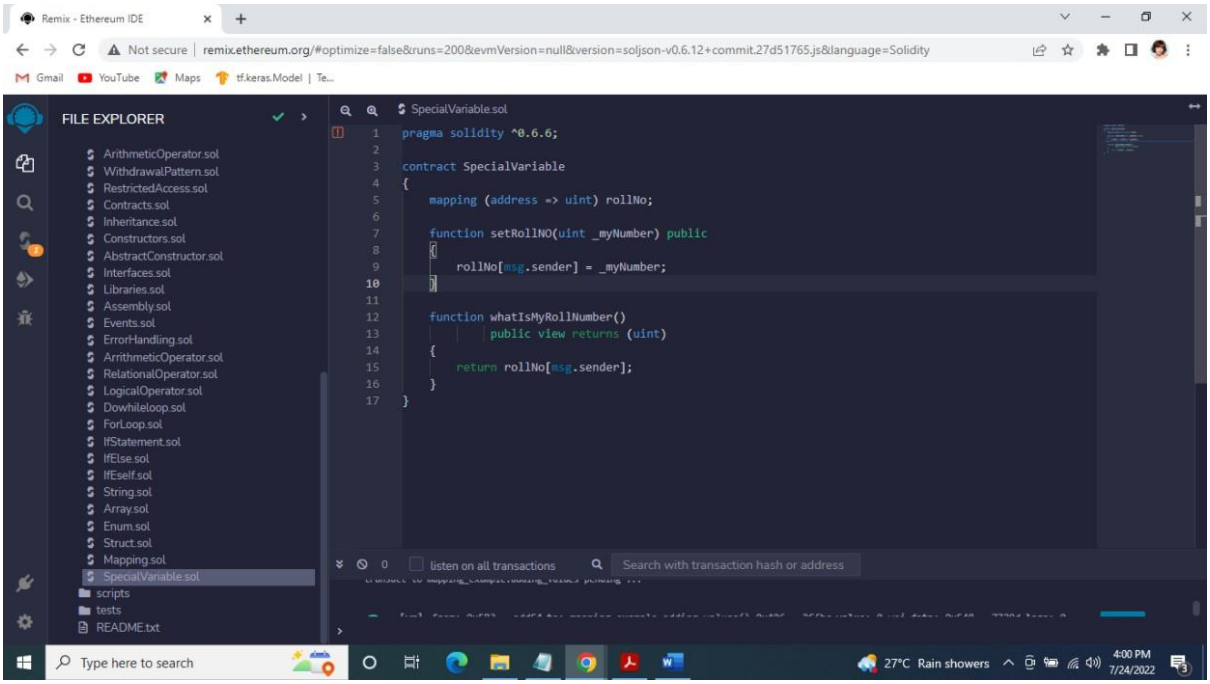


Output:

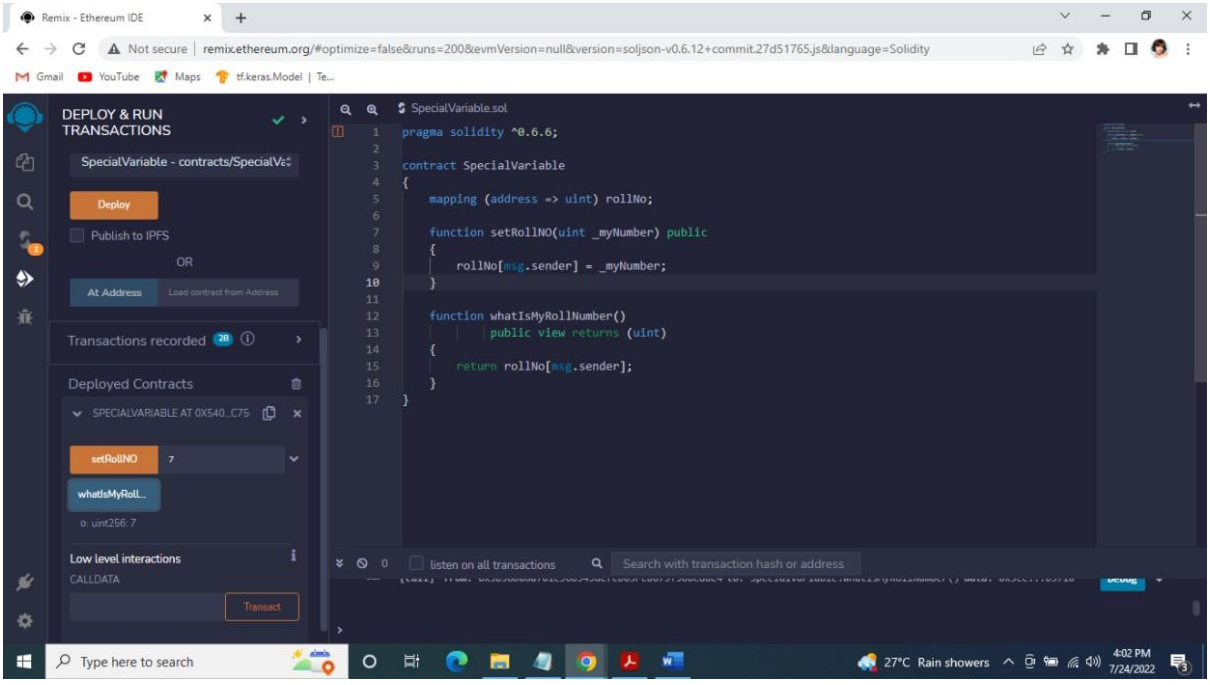


Special Variable

Code:



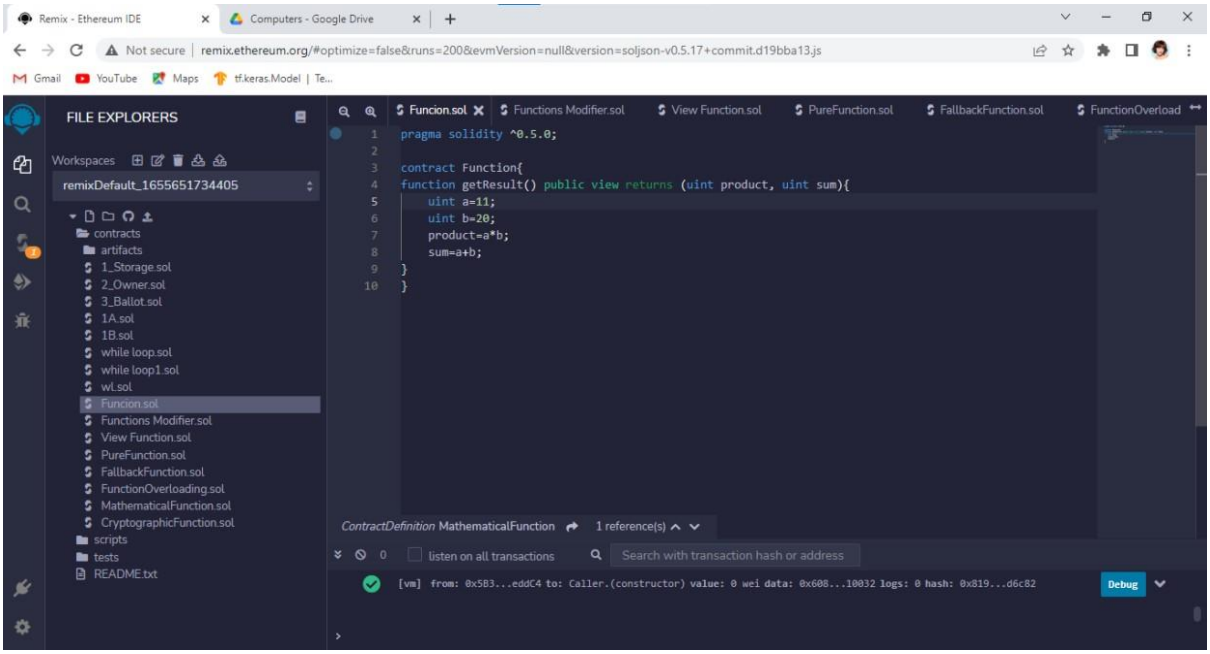
Output:



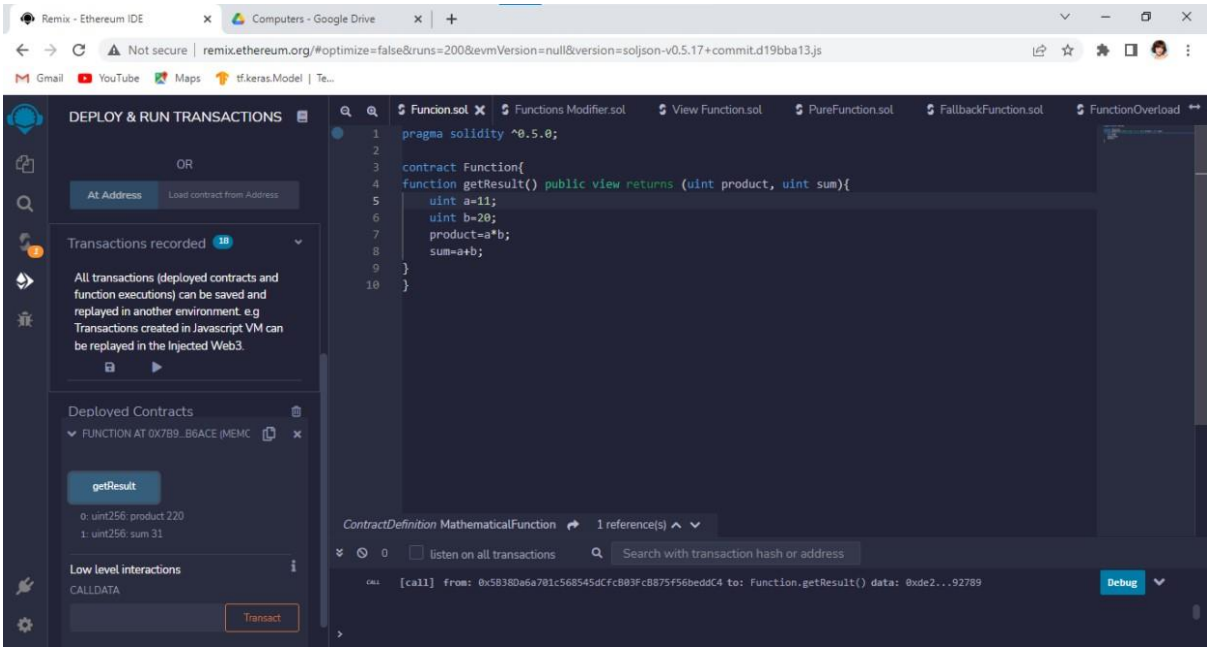
2b) Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

Function:

Code:

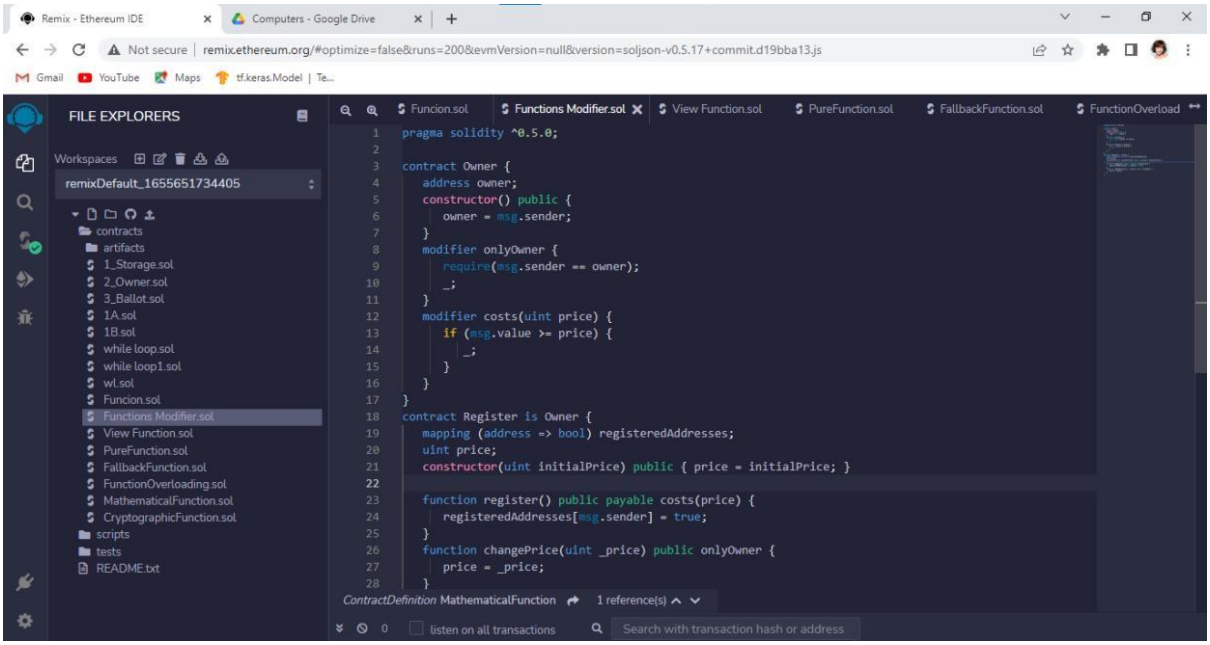


Output:

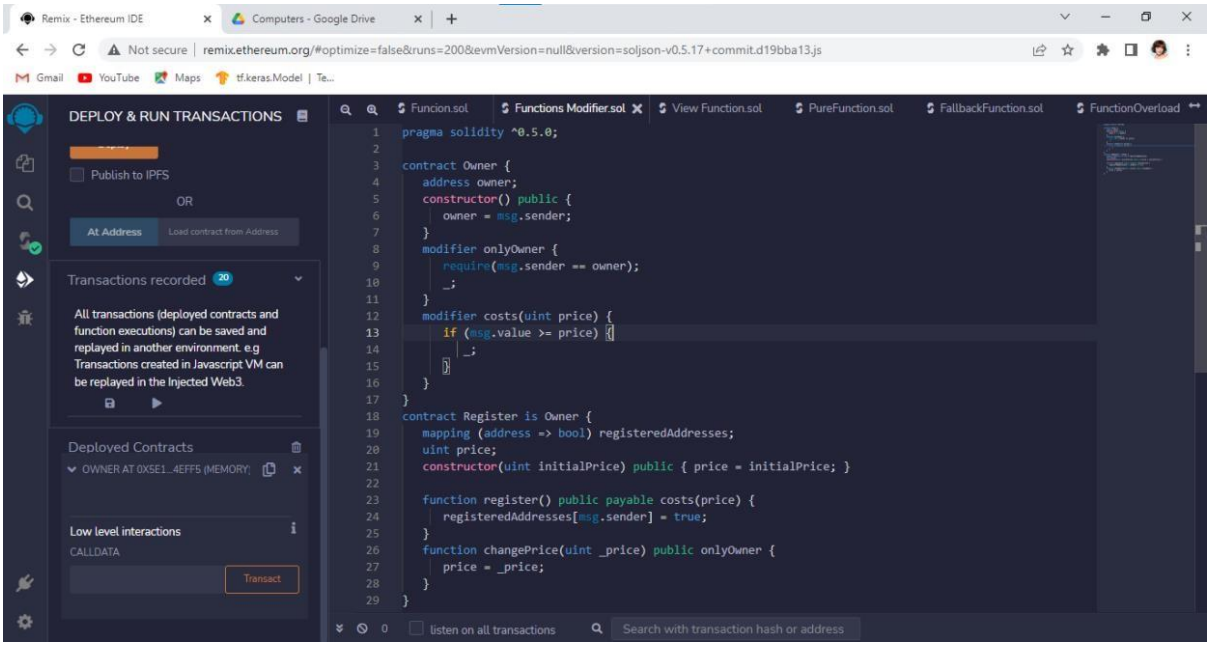


Functions Modifiers

Code:

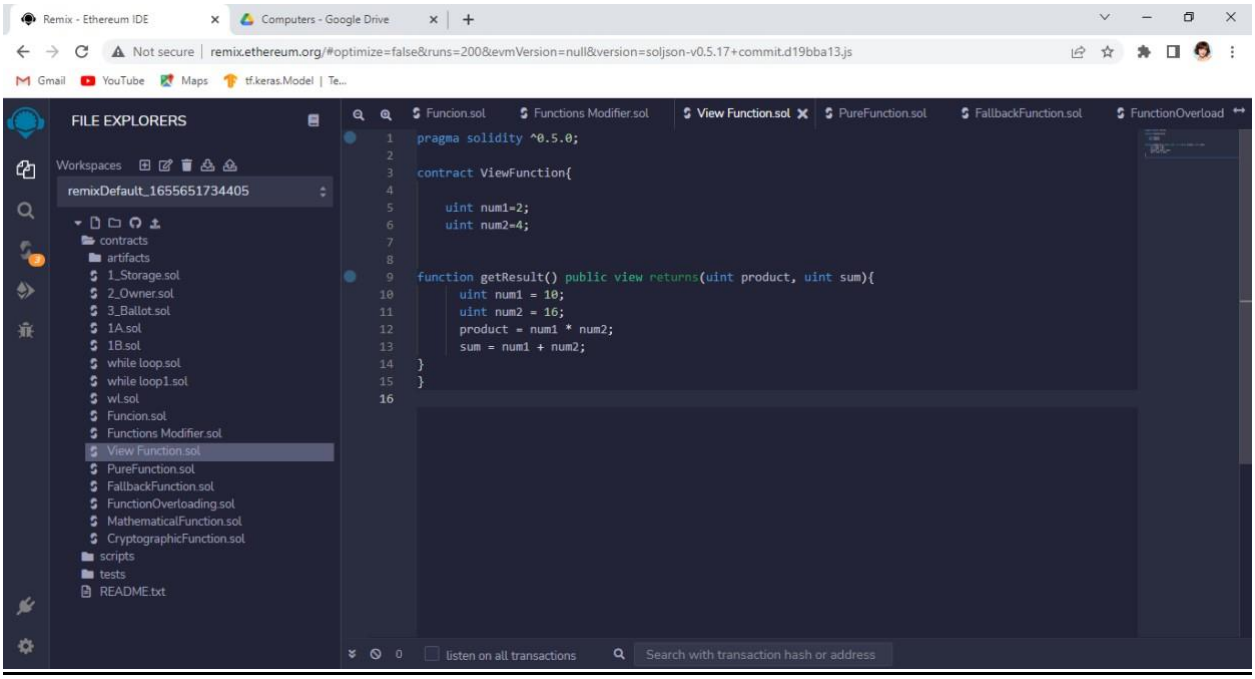


Output:

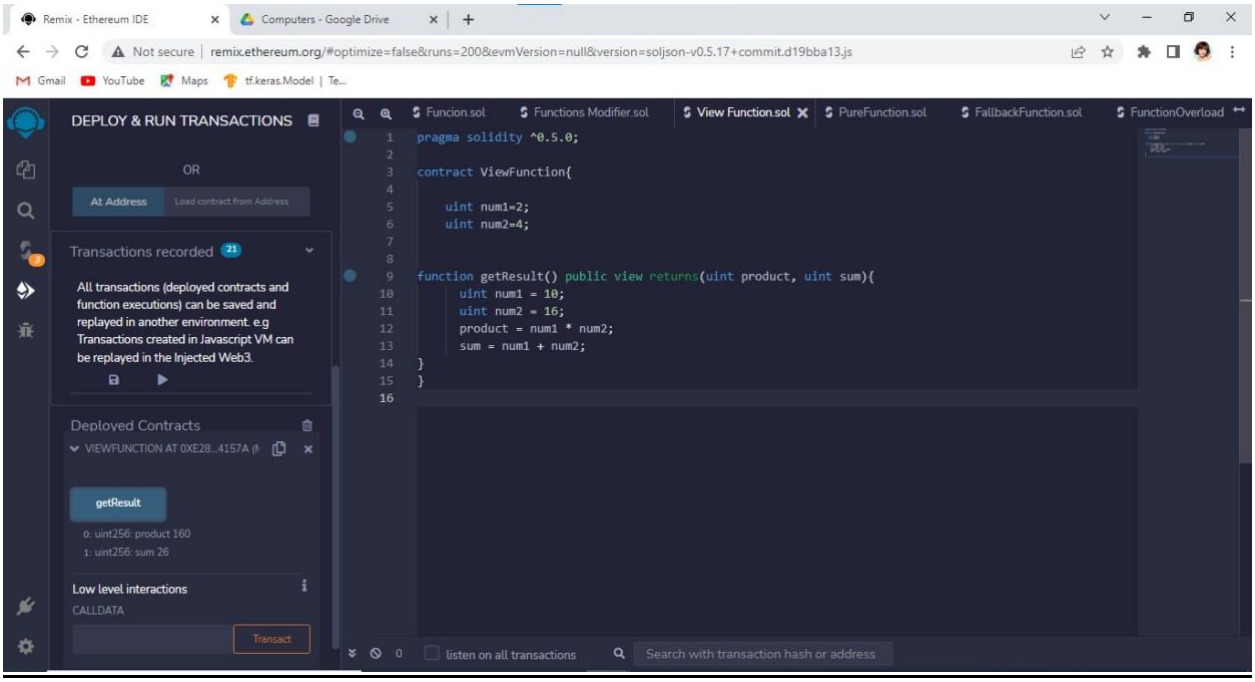


View function

Code:

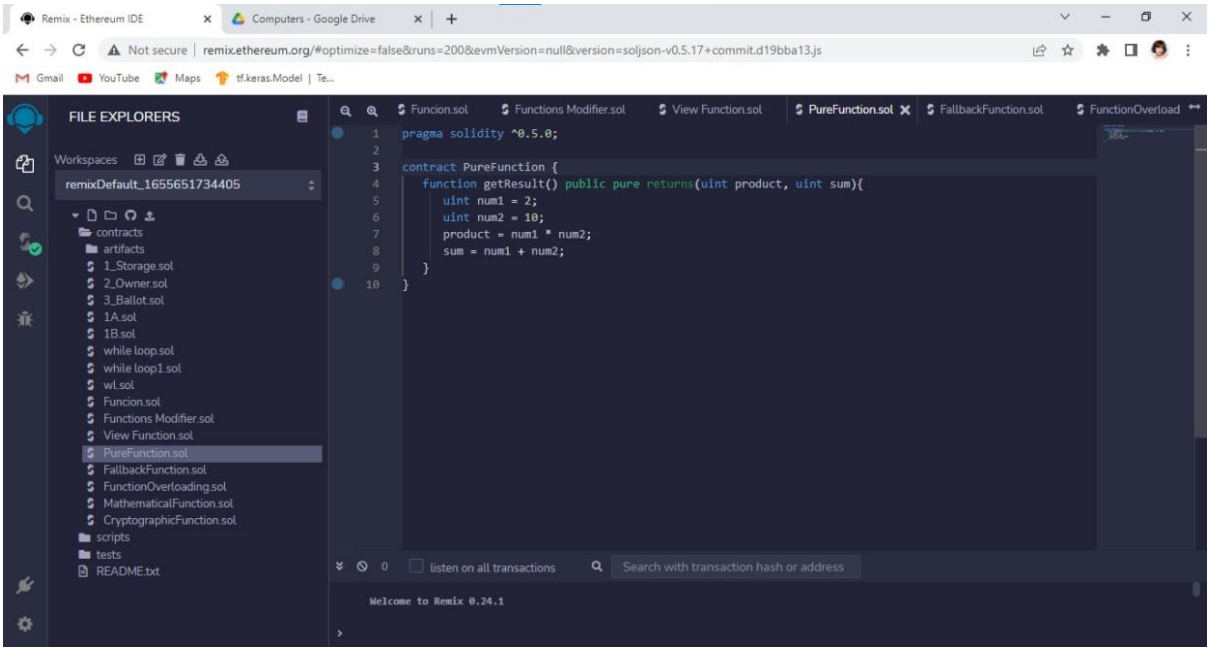


Output:

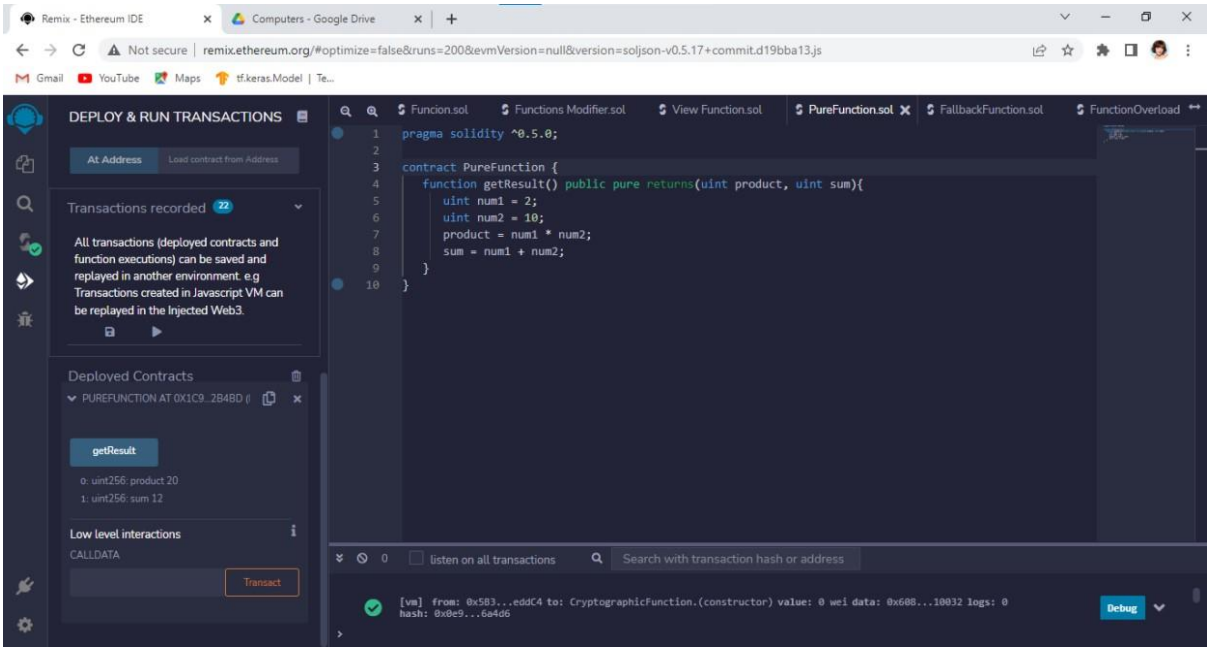


Pure function

Code:

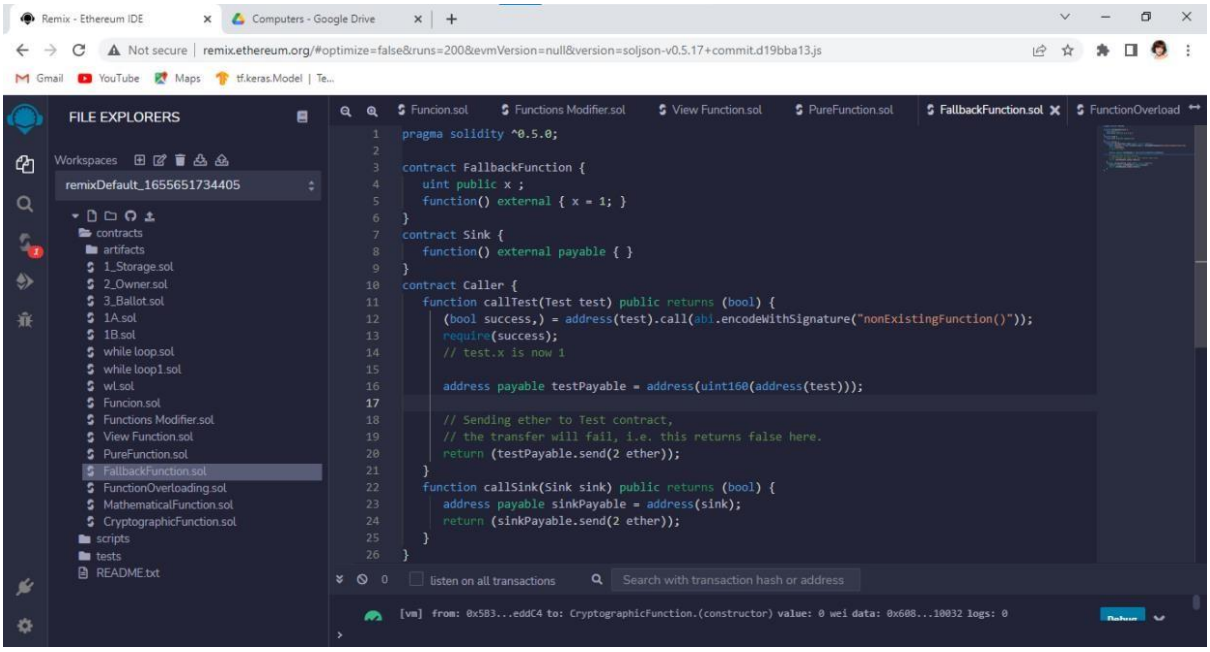


Ouput:

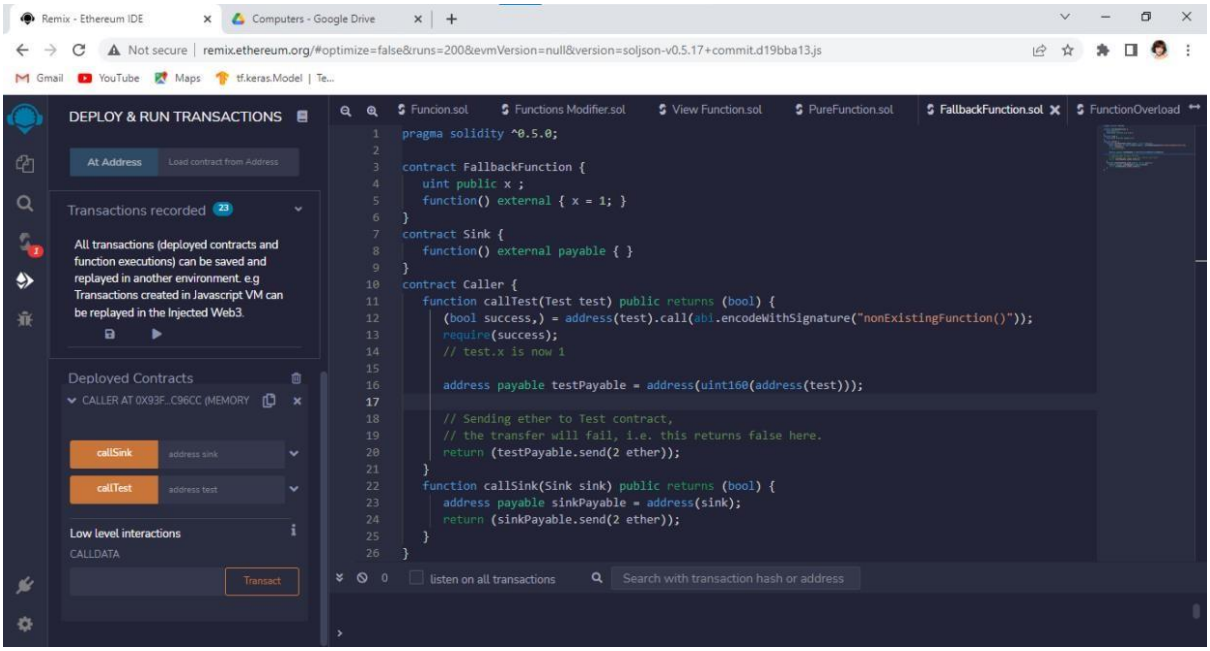


Fallback function

Code:

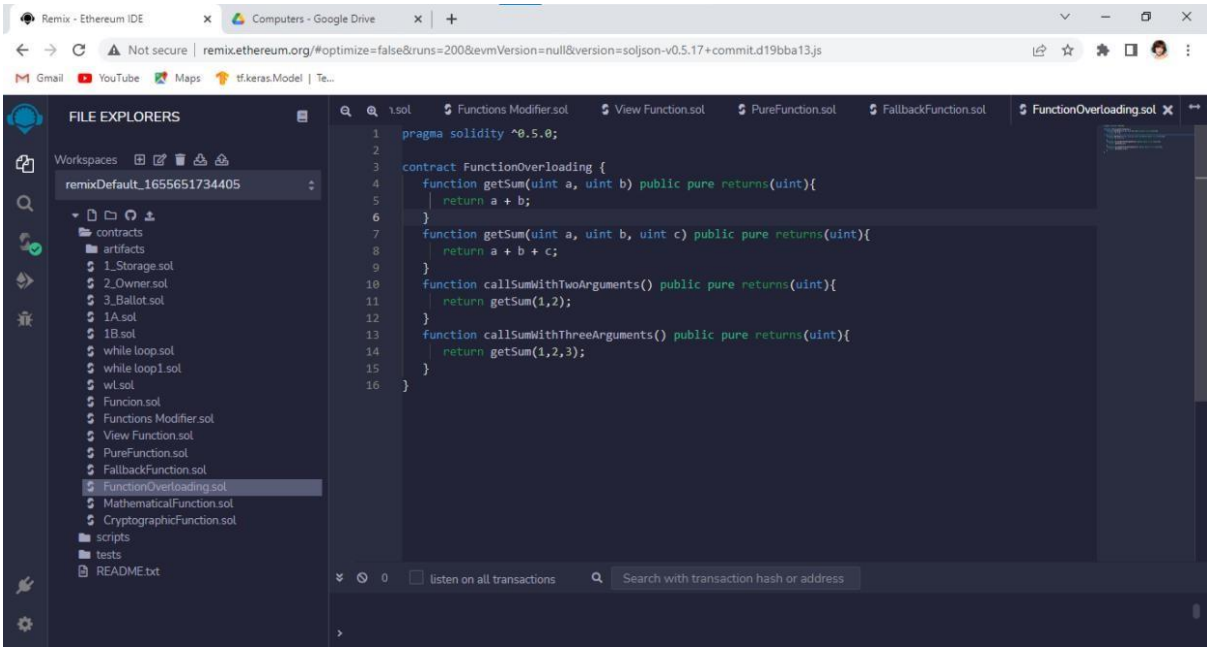


Output:

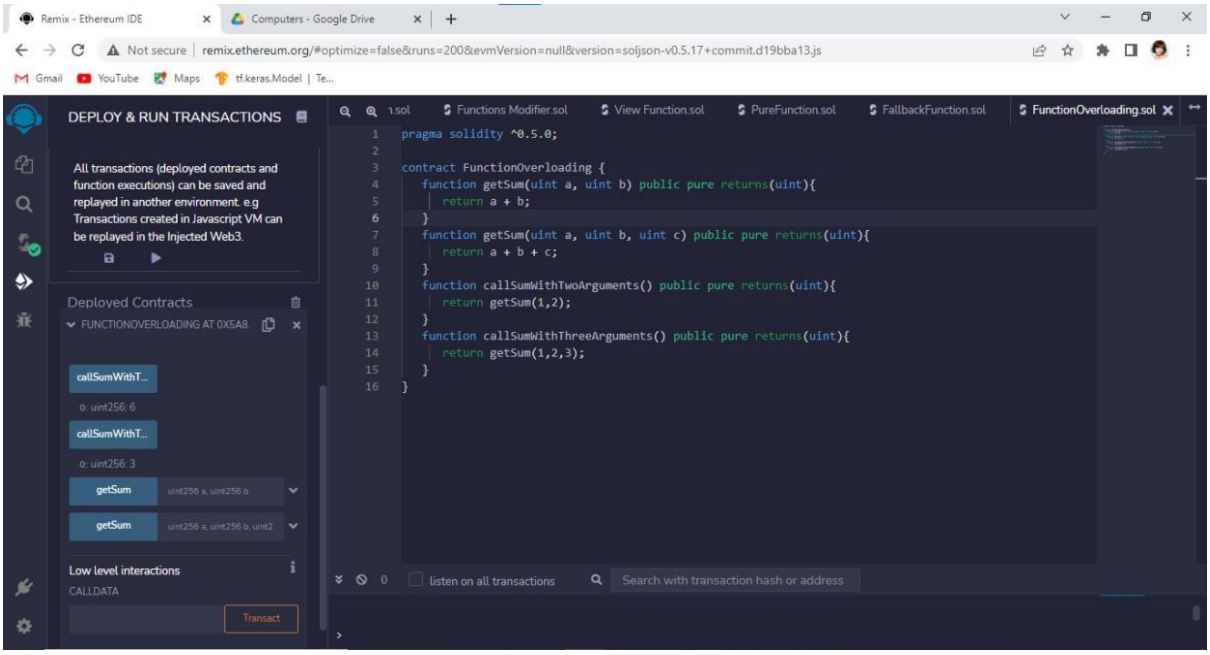


Function Overloading

Code:

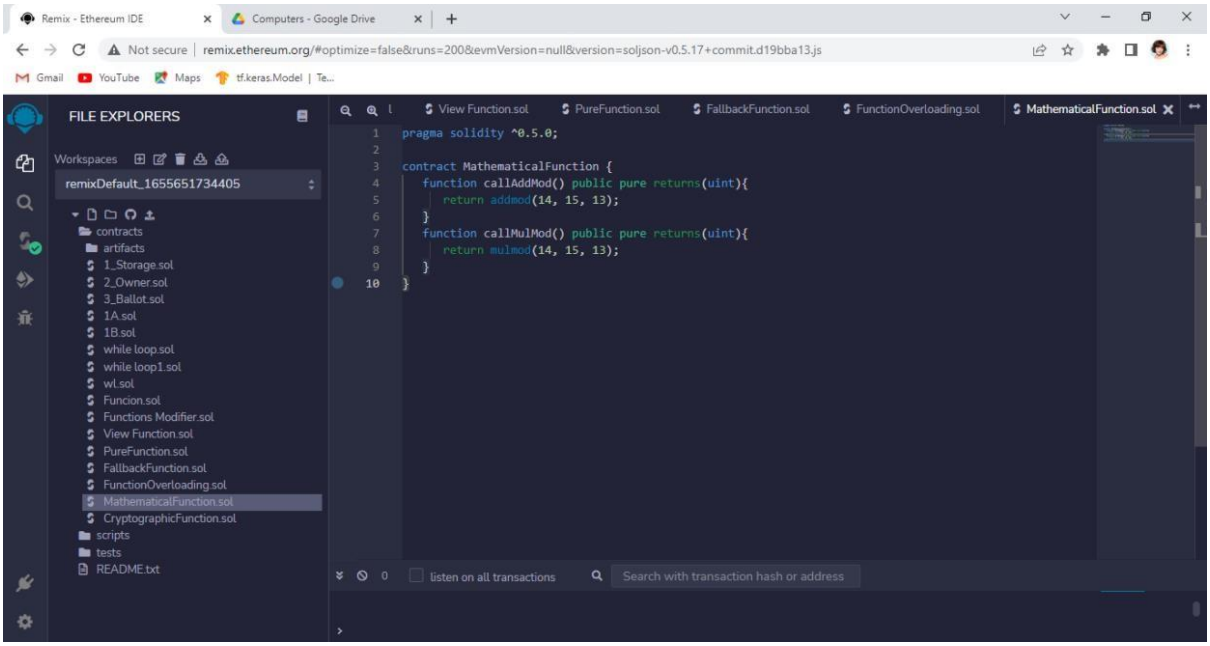


Output:

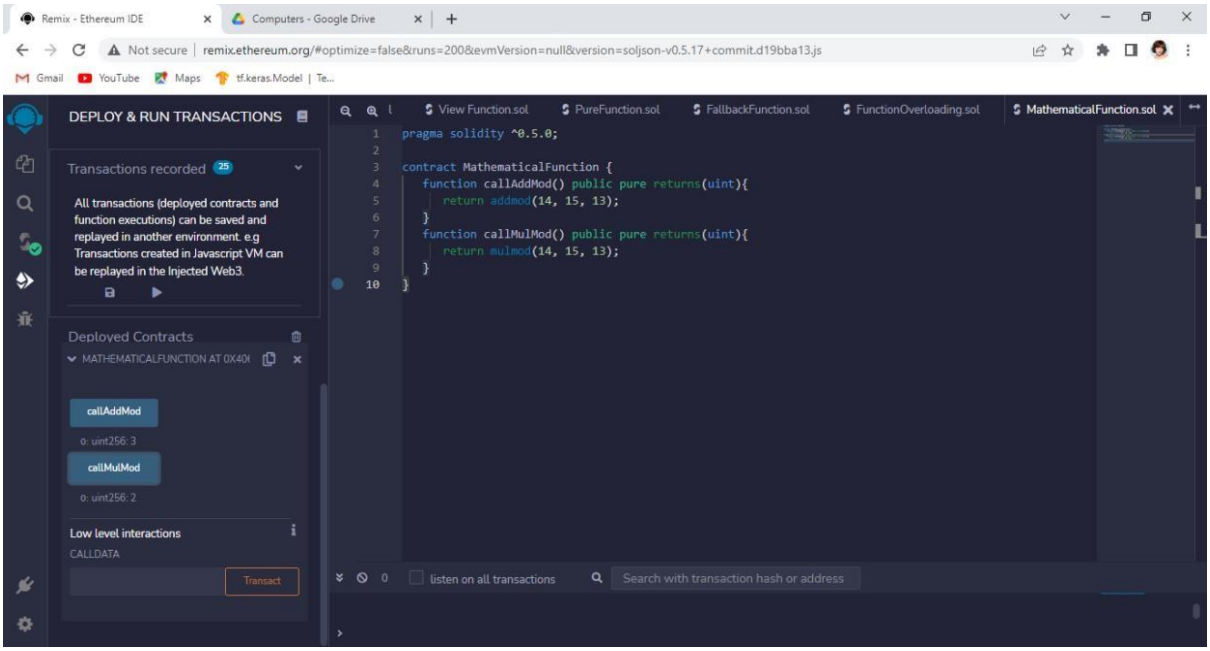


Mathematical Function

Code:

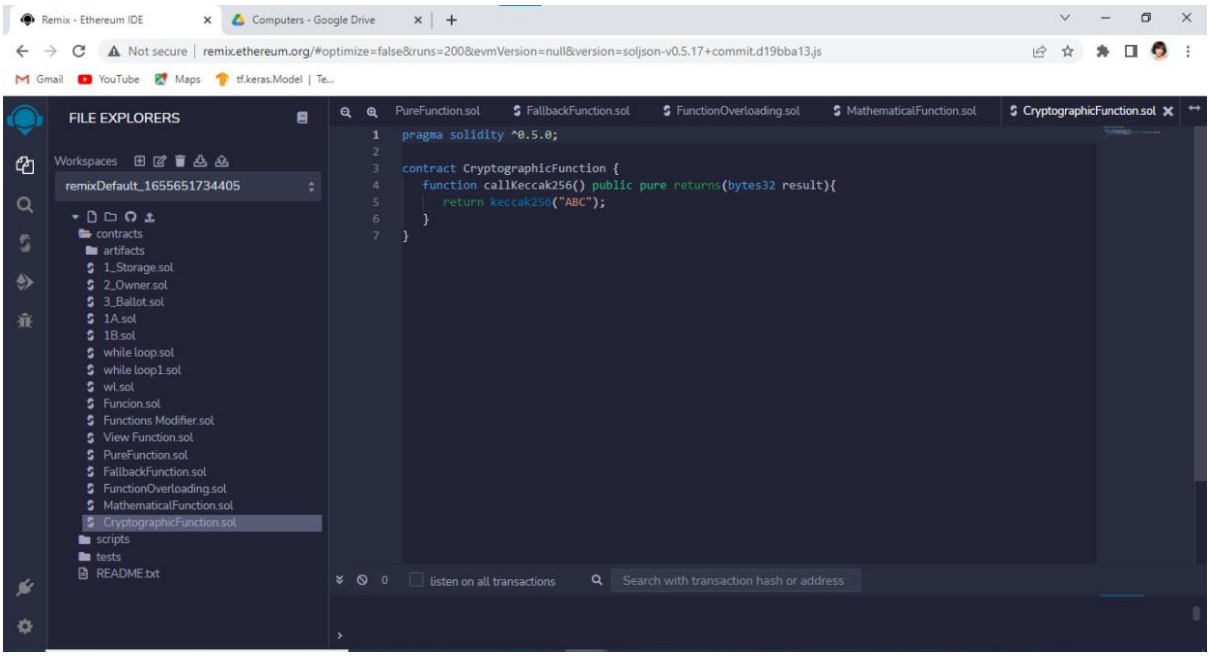


Output:

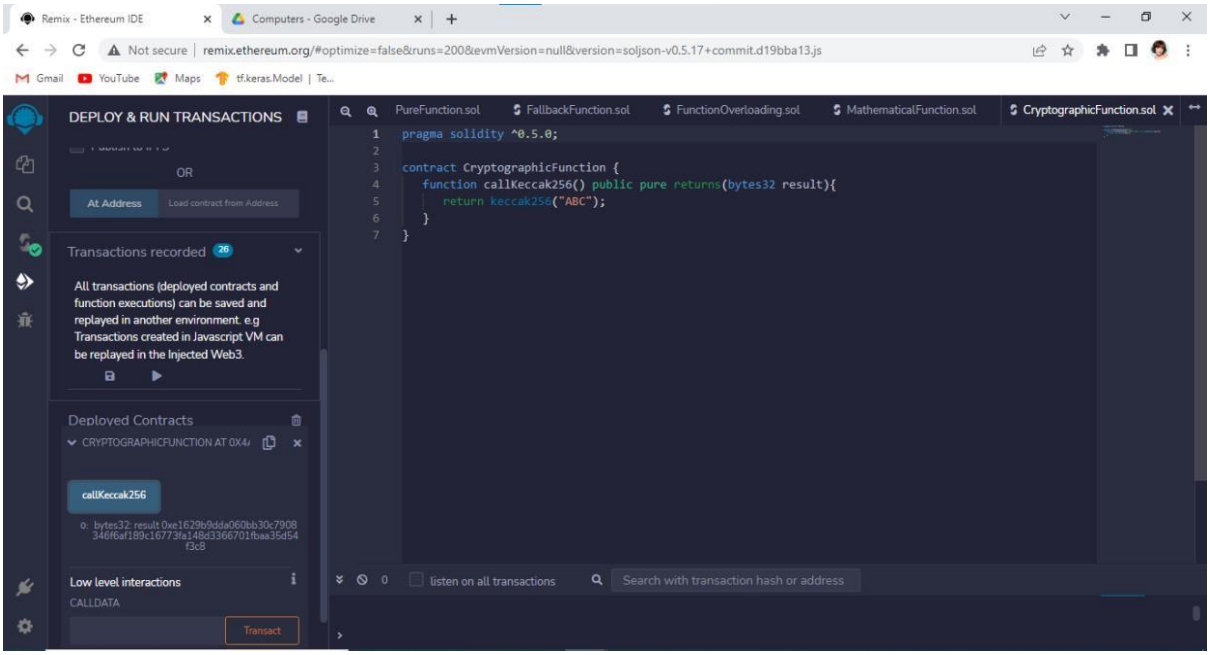


Cryptographic function

Code:



Output:


Remix - Ethereum IDE
Computers - Google Drive
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail
YouTube
Maps
tf.keras.Model | Te...
DEPLOY & RUN TRANSACTIONS
OR
At Address
Load contract from Address
Transactions recorded 26
All transactions (deployed contracts and function executions) can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.
Deployed Contracts
CRYPTOGRAPHICFUNCTION AT 0X4
callKeccak256
0: bytes32: result 0xe1629b9dda060bb30c7908346f6af189c16773fa148d3366701fbaa35d54f3c8
Low level interactions
CALLDATA
Transact
PureFunction.sol
FallbackFunction.sol
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
pragma solidity ^0.5.0;
contract CryptographicFunction {
function callKeccak256() public pure returns(bytes32 result){
return keccak256("ABC");
}
}
listen on all transactions
Search with transaction hash or address

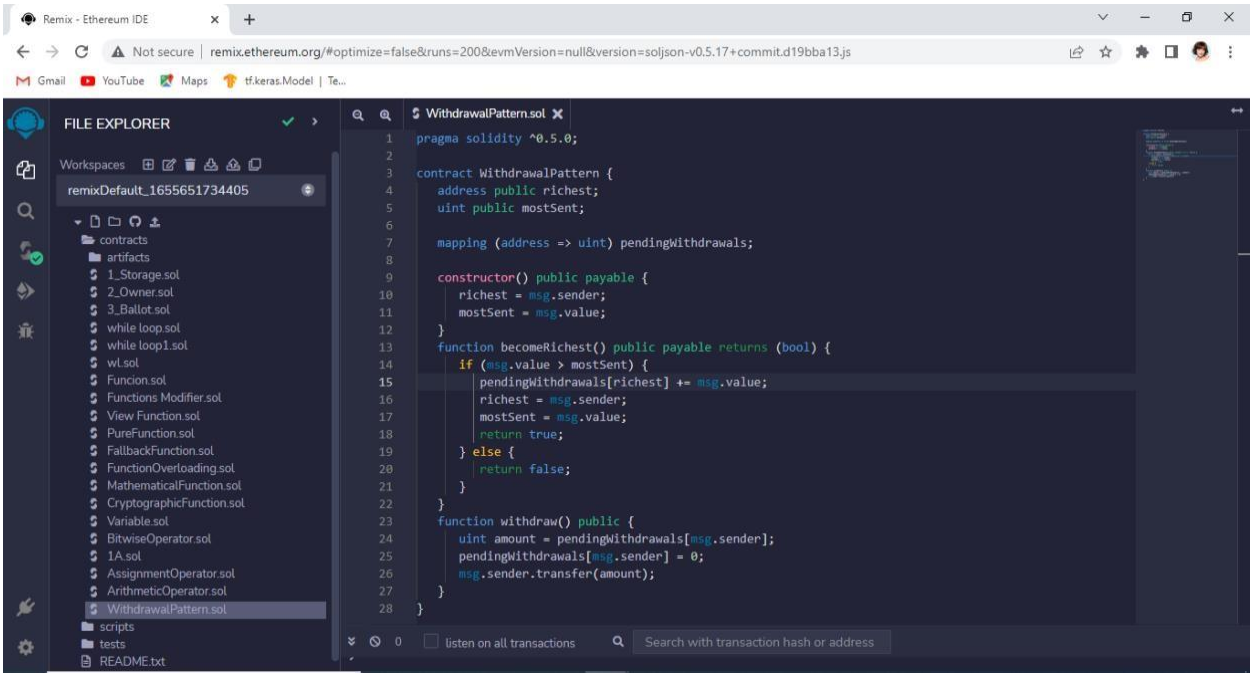
Practical No: 3

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

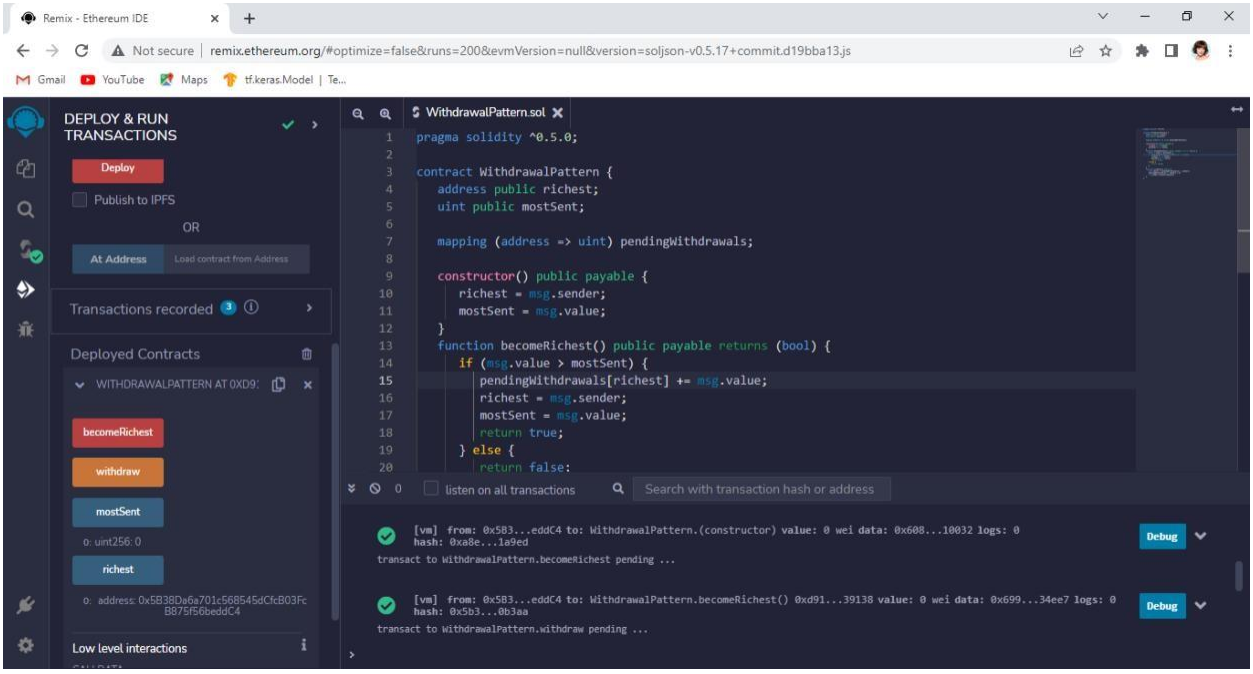
3a) Withdrawal Pattern, Restricted Access.

Withdrawal Pattern

Code:

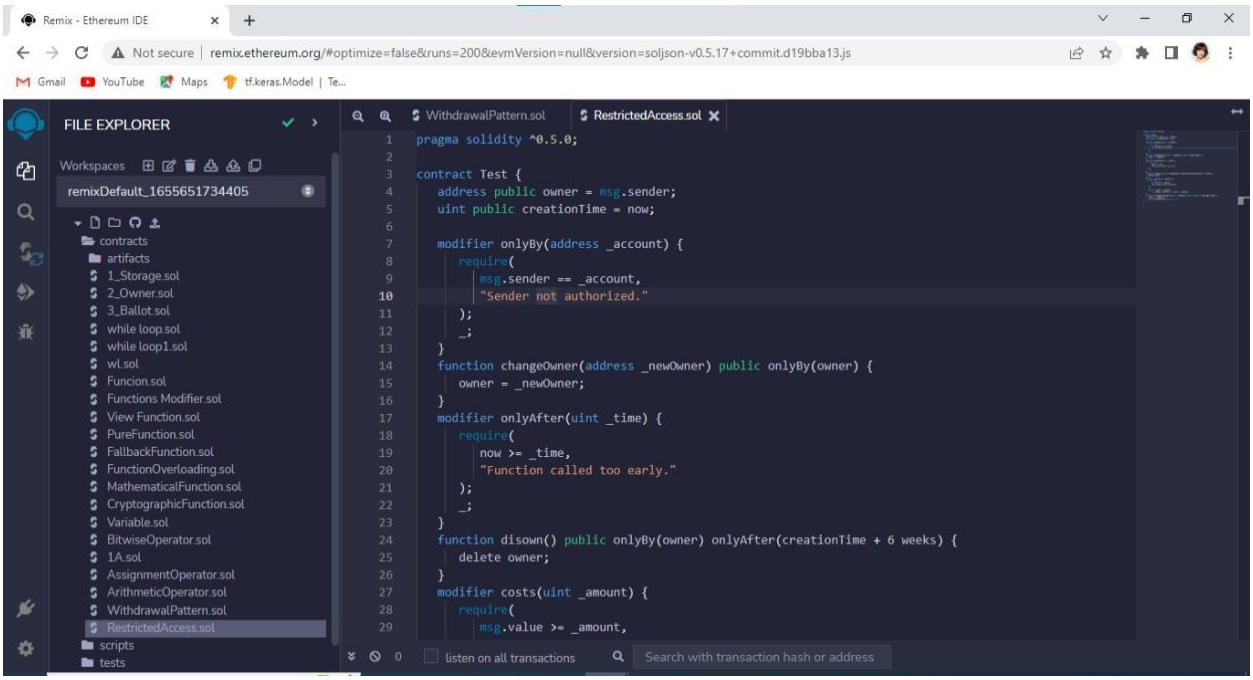


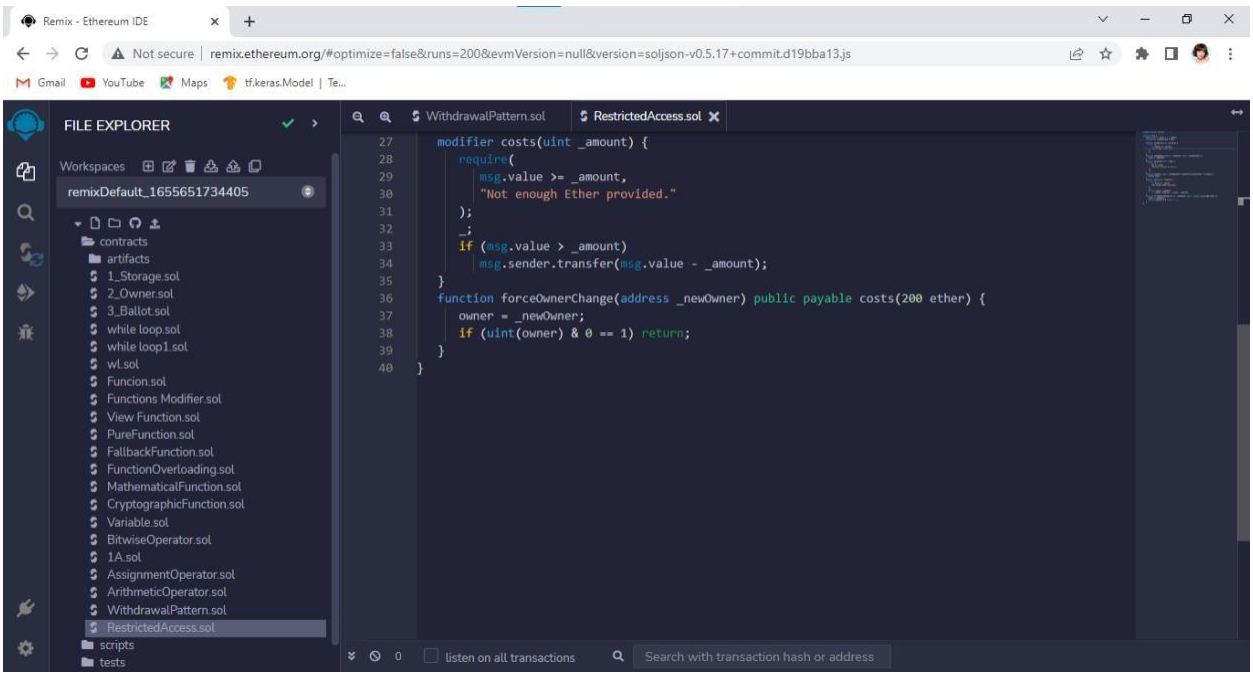
Output:



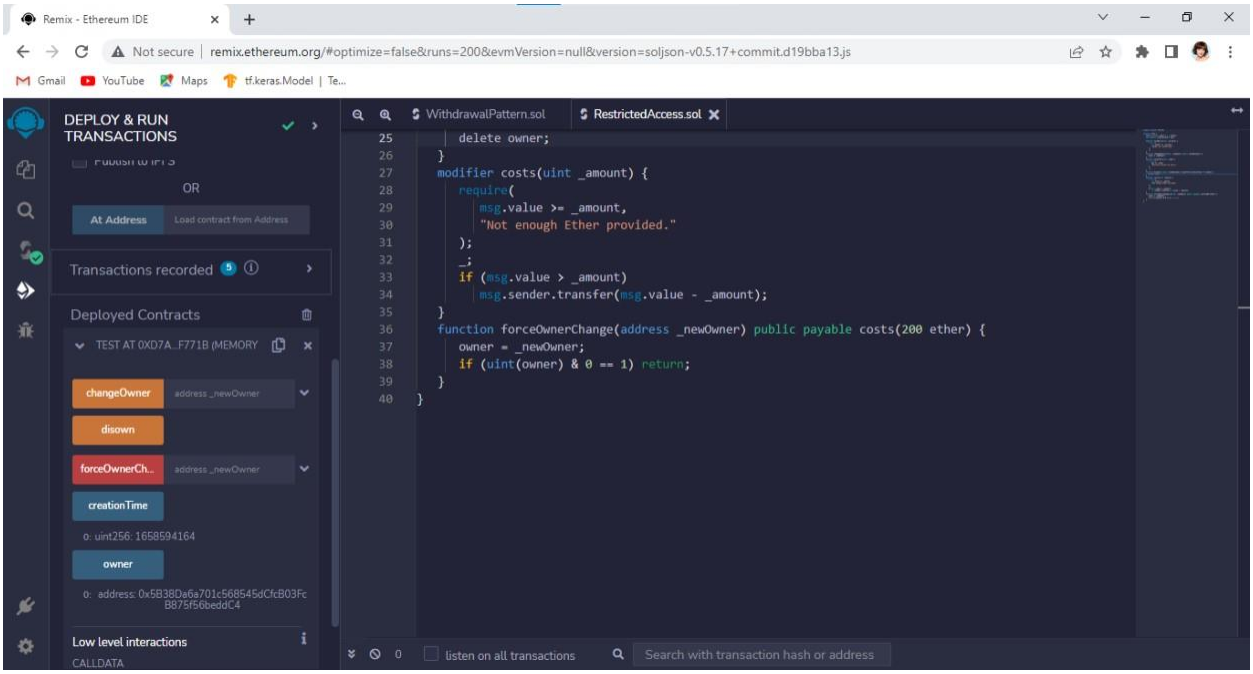
Restricted Access

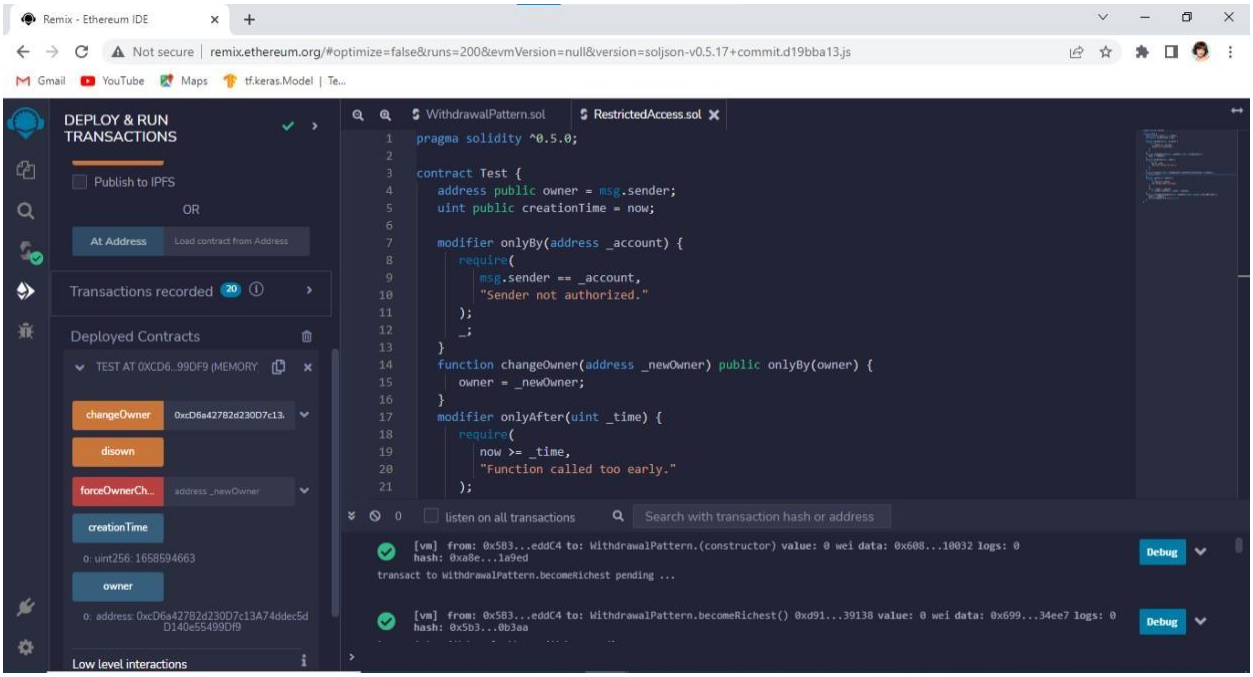
Code:




Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail YouTube Maps tf.keras.Model | Te...
FILE EXPLORER
Workspaces
remixDefault_1655651734405
contracts
artifacts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
while loop.sol
while loop1.sol
wl.sol
Funcion.sol
Functions Modifier.sol
View Function.sol
PureFunction.sol
FallbackFunction.sol
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
Variable.sol
BitwiseOperator.sol
1A.sol
AssignmentOperator.sol
ArithmeticOperator.sol
WithdrawalPattern.sol
RestrictedAccess.sol
scripts
tests
WithdrawalPattern.sol
RestrictedAccess.sol
27 modifier costs(uint _amount) {
28 require(
29 msg.value >= _amount,
30 "Not enough Ether provided."
31);
32 _;
33 if (msg.value > _amount)
34 msg.sender.transfer(msg.value - _amount);
35 }
36 function forceOwnerChange(address _newOwner) public payable costs(200 ether) {
37 owner = _newOwner;
38 if (uint(owner) & 0 == 1) return;
39 }
40 }
0
listen on all transactions
Search with transaction hash or address

Output:

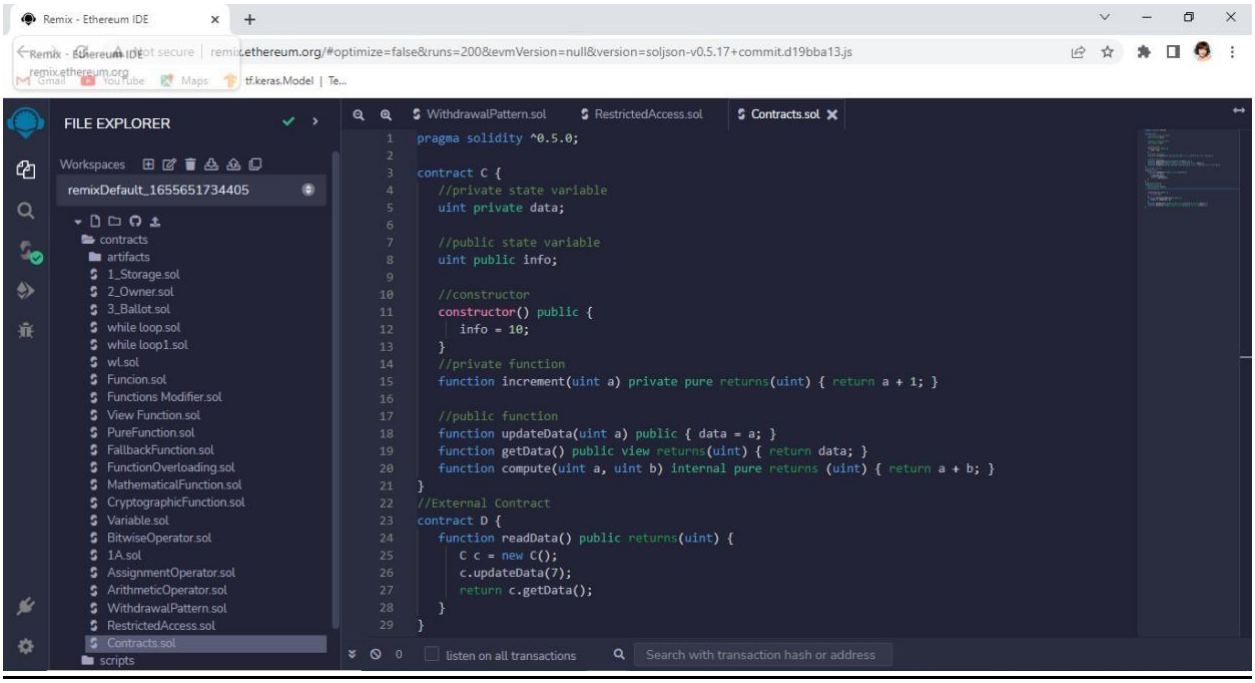

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail YouTube Maps tf.keras.Model | Te...
DEPLOY & RUN TRANSACTIONS
OR
At Address
Load contract from Address
Transactions recorded 5
Deployed Contracts
TEST AT 0XD7A...F771B (MEMORY
changeOwner
address _newOwner
disown
forceOwnerCh...
address _newOwner
creationTime
0: uint256: 1658594164
owner
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
Low level interactions
CALLDATA
WithdrawalPattern.sol
RestrictedAccess.sol
25 delete owner;
26 }
27 modifier costs(uint _amount) {
28 require(
29 msg.value >= _amount,
30 "Not enough Ether provided."
31);
32 _;
33 if (msg.value > _amount)
34 msg.sender.transfer(msg.value - _amount);
35 }
36 function forceOwnerChange(address _newOwner) public payable costs(200 ether) {
37 owner = _newOwner;
38 if (uint(owner) & 0 == 1) return;
39 }
40 }
0
listen on all transactions
Search with transaction hash or address

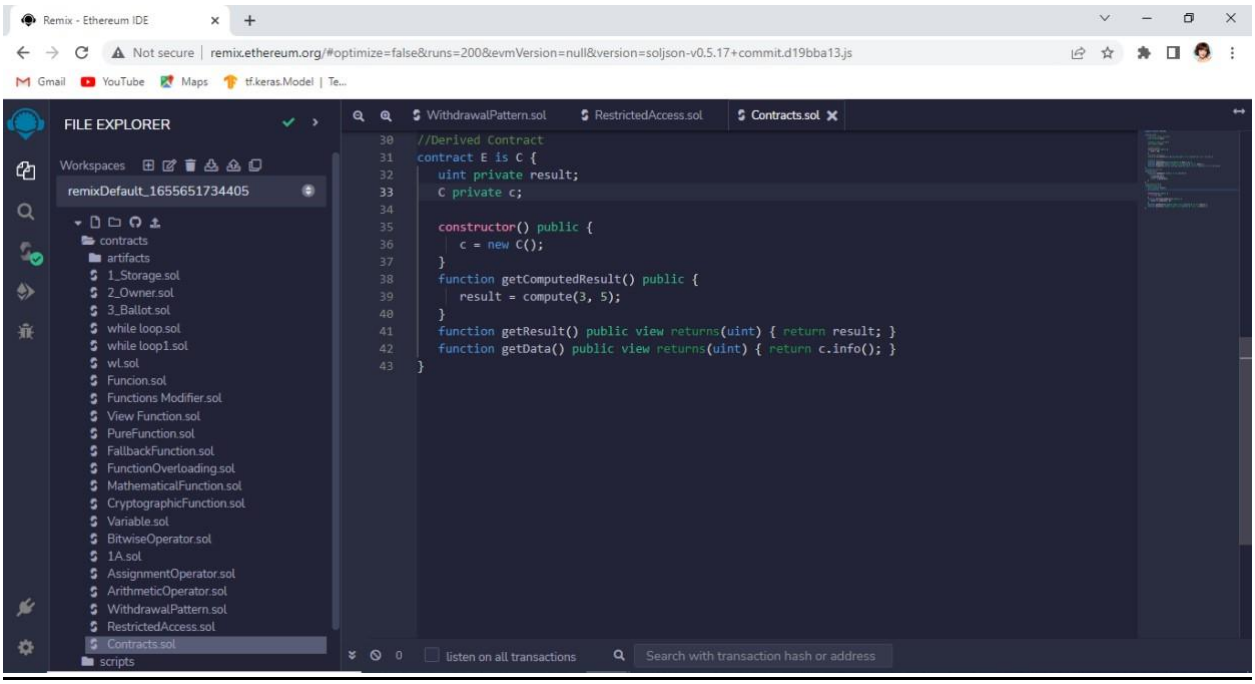


3b) Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.

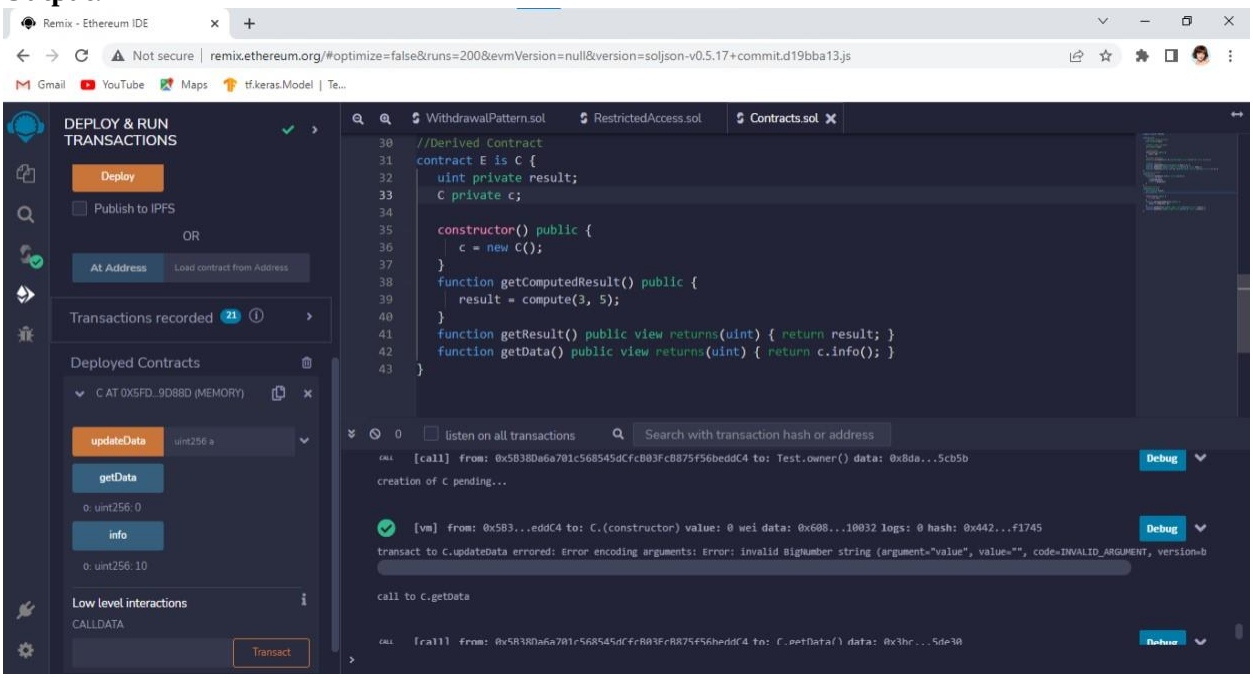
Contracts

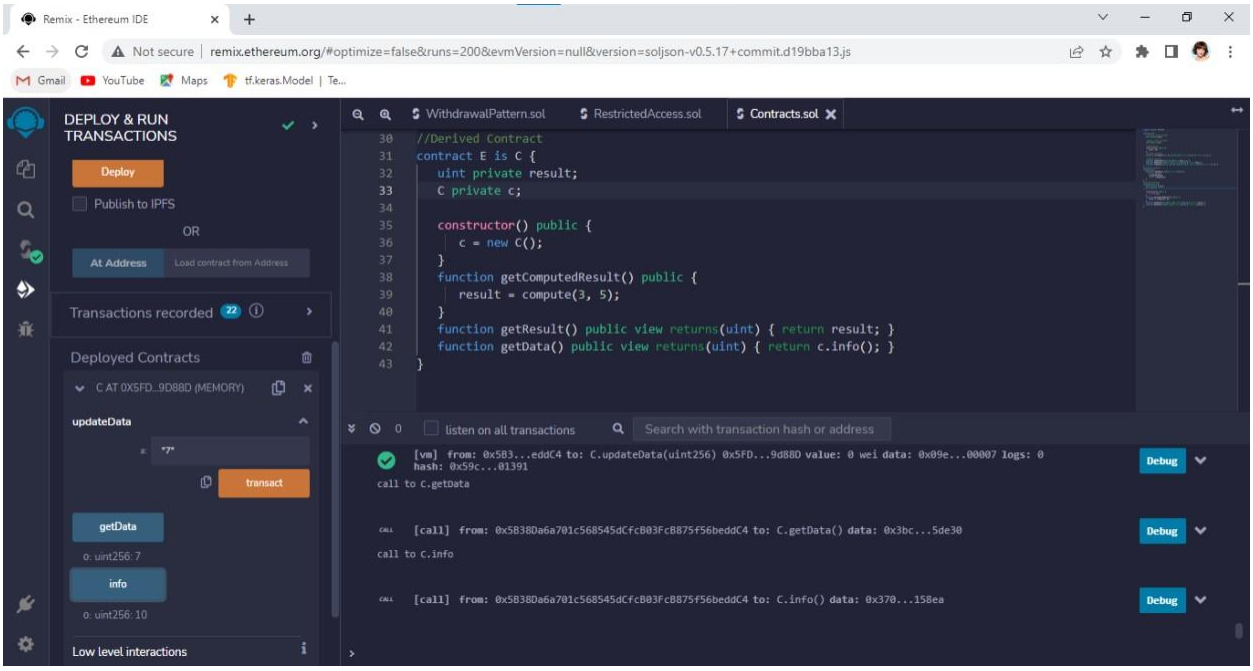
Code:




Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail
YouTube
Maps
tf.keras.Model | Te...
FILE EXPLORER
Workspaces
remixDefault_1655651734405
contracts
artifacts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
while loop.sol
while loop1.sol
wl.sol
Funcion.sol
Functions Modifier.sol
View Function.sol
PureFunction.sol
FallbackFunction.sol
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
Variable.sol
BitwiseOperator.sol
1A.sol
AssignmentOperator.sol
ArithmeticOperator.sol
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
scripts
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
30 //Derived Contract
31 contract E is C {
32 uint private result;
33 C private c;
34
35 constructor() public {
36 c = new C();
37 }
38 function getComputedResult() public {
39 result = compute(3, 5);
40 }
41 function getResult() public view returns(uint) { return result; }
42 function getData() public view returns(uint) { return c.info(); }
43 }
0
listen on all transactions
Search with transaction hash or address

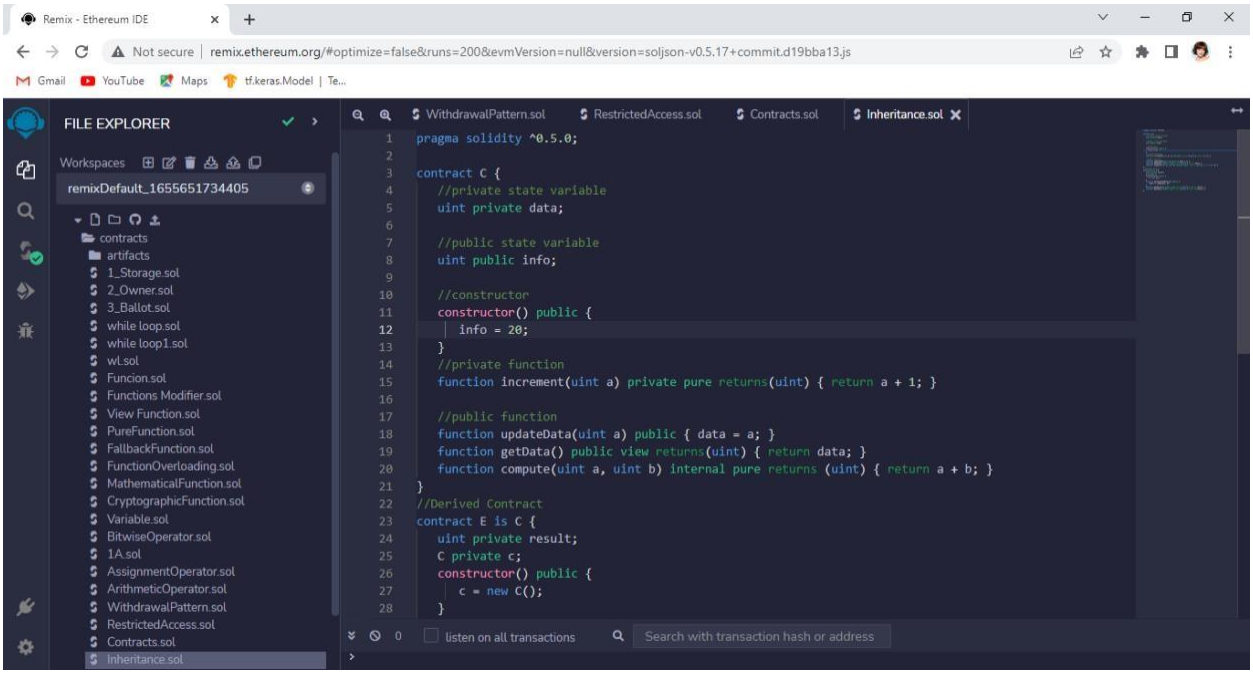
Output:

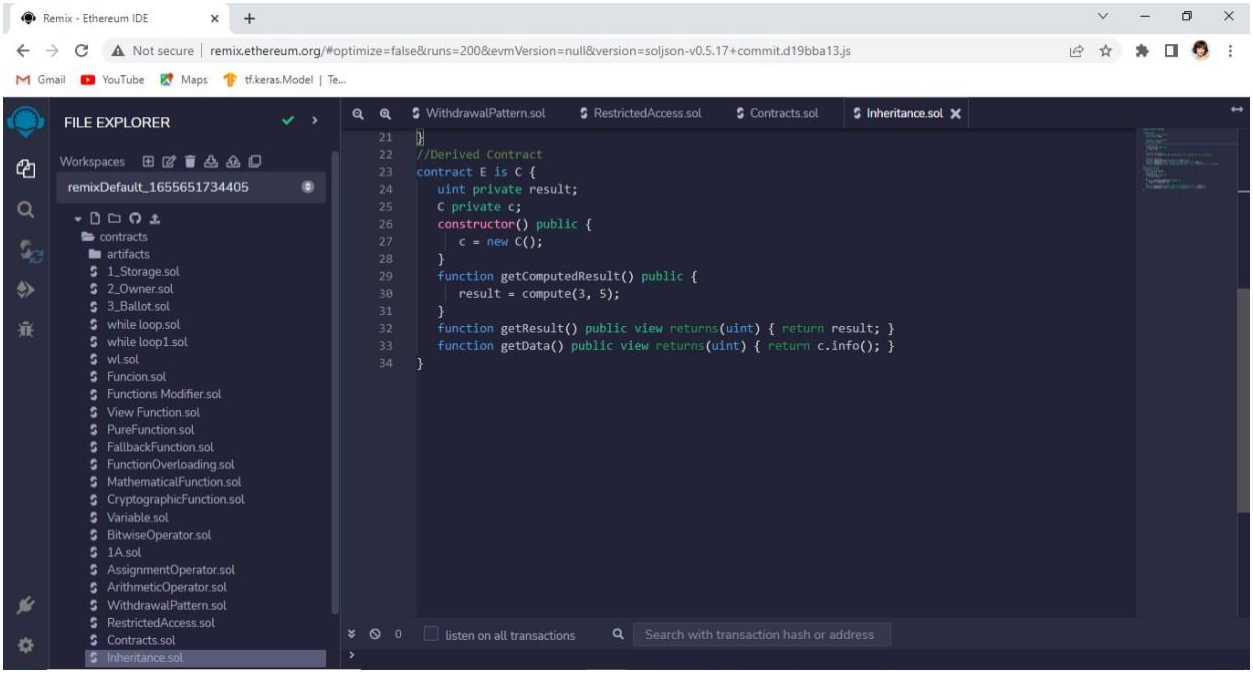

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail
YouTube
Maps
tf.keras.Model | Te...
DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 21
Deployed Contracts
C AT 0X5FD...9D88D (MEMORY)
updateData
uint256 a
getData
0: uint256: 0
info
0: uint256: 10
Low level interactions
CALLDATA
Transact
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
30 //Derived Contract
31 contract E is C {
32 uint private result;
33 C private c;
34
35 constructor() public {
36 c = new C();
37 }
38 function getComputedResult() public {
39 result = compute(3, 5);
40 }
41 function getResult() public view returns(uint) { return result; }
42 function getData() public view returns(uint) { return c.info(); }
43 }
0
listen on all transactions
Search with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Test.owner() data: 0x8da...5cb5b
Debug
creation of C pending...
[vm] from: 0x583...eddC4 to: C.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x442...f1745
Debug
transact to C.updateData errored: Error encoding arguments: Error: invalid BigNumber string (argument="value", value="", code=INVALID_ARGUMENT, version=b
call to C.getData



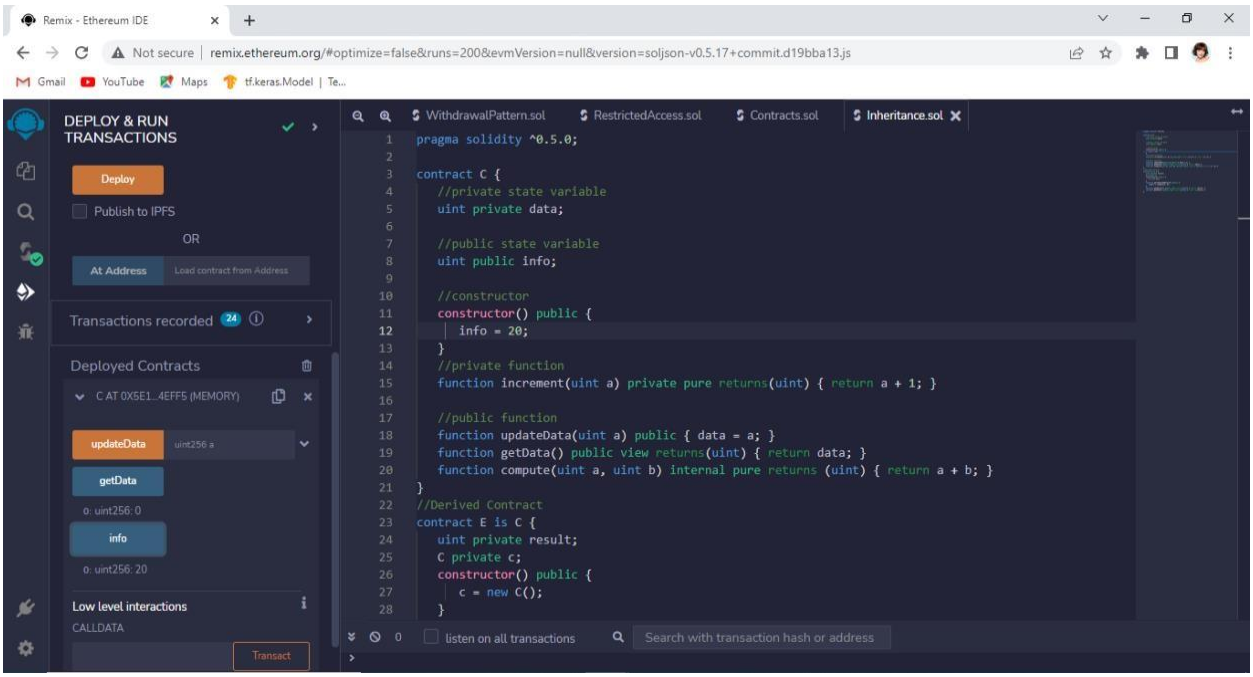
Inheritance

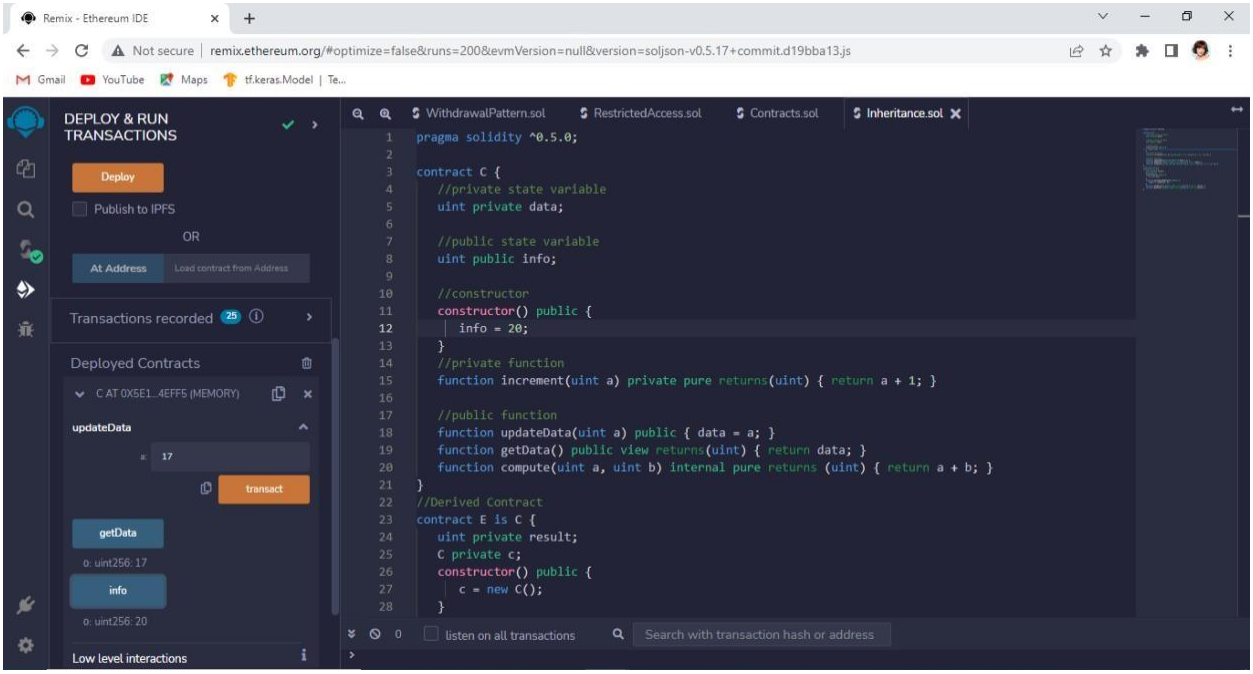
Code:




Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail YouTube Maps tf.keras.Model | Te...
FILE EXPLORER
Workspaces
remixDefault_1655651734405
contracts
artifacts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
while loop.sol
while loop1.sol
wl.sol
Funcion.sol
Functions Modifier.sol
View Function.sol
PureFunction.sol
FallbackFunction.sol
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
Variable.sol
BitwiseOperator.sol
1A.sol
AssignmentOperator.sol
ArithmeticOperator.sol
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
Inheritance.sol
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
Inheritance.sol
21 }
22 //Derived Contract
23 contract E is C {
24 uint private result;
25 C private c;
26 constructor() public {
27 c = new C();
28 }
29 function getComputedResult() public {
30 result = compute(3, 5);
31 }
32 function getResult() public view returns(uint) { return result; }
33 function getData() public view returns(uint) { return c.info(); }
34 }
0
listen on all transactions
Search with transaction hash or address

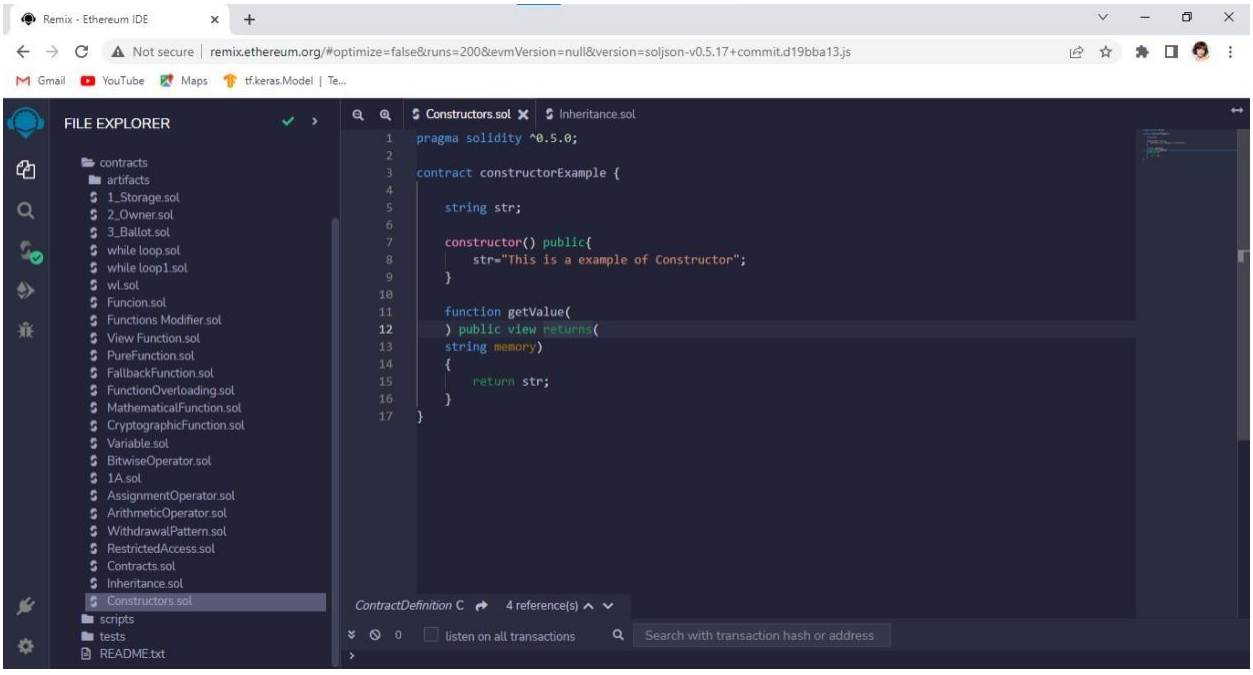
Output:


Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail YouTube Maps tf.keras.Model | Te...
DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 24
Deployed Contracts
C AT 0X5E1...4EFF5 (MEMORY)
updateData
uint256 a
getData
0: uint256: 0
info
0: uint256: 20
Low level interactions
CALLDATA
Transact
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
Inheritance.sol
1 pragma solidity ^0.5.0;
2
3 contract C {
4 //private state variable
5 uint private data;
6
7 //public state variable
8 uint public info;
9
10 //constructor
11 constructor() public {
12 info = 20;
13 }
14 //private function
15 function increment(uint a) private pure returns(uint) { return a + 1; }
16
17 //public function
18 function updateData(uint a) public { data = a; }
19 function getData() public view returns(uint) { return data; }
20 function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
21 }
22 //Derived Contract
23 contract E is C {
24 uint private result;
25 C private c;
26 constructor() public {
27 c = new C();
28 }
0
listen on all transactions
Search with transaction hash or address

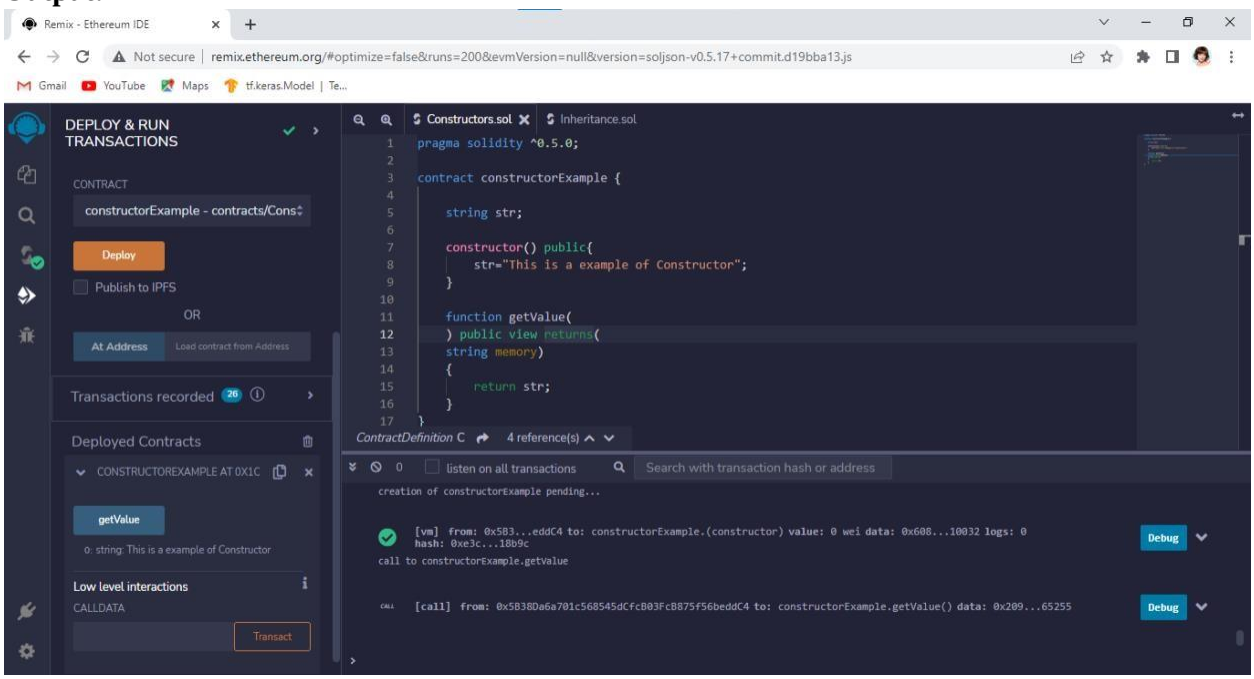


Constructors

Code:

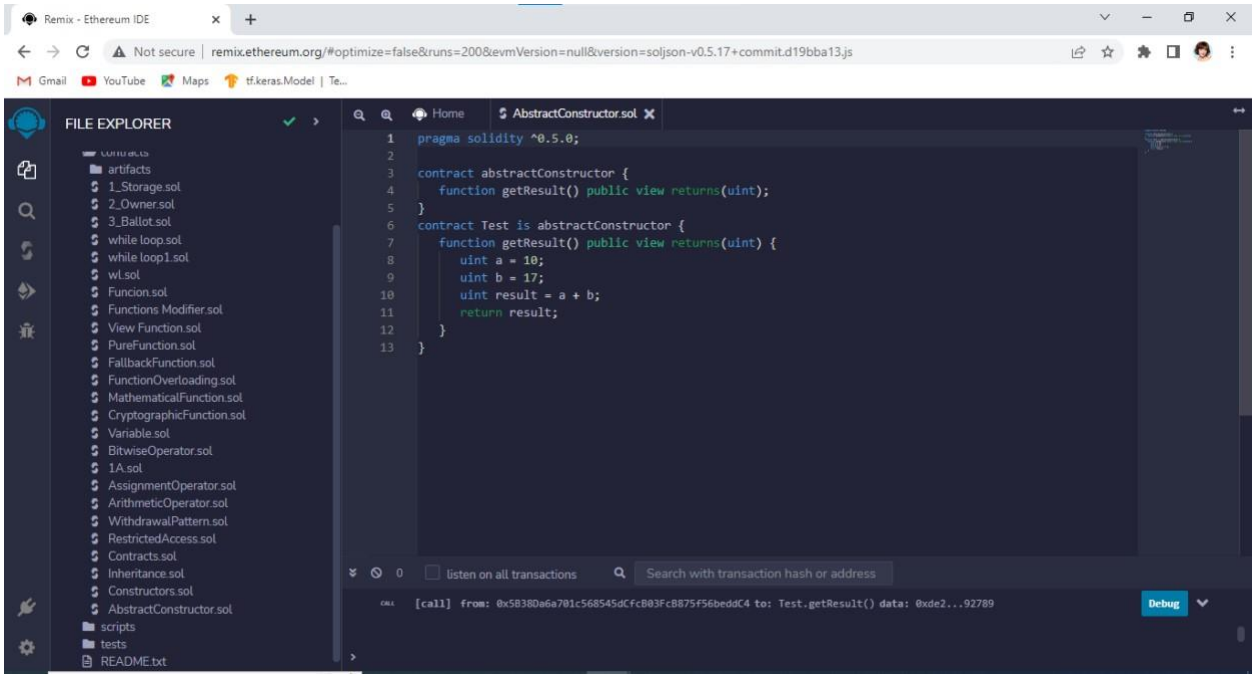


Output:

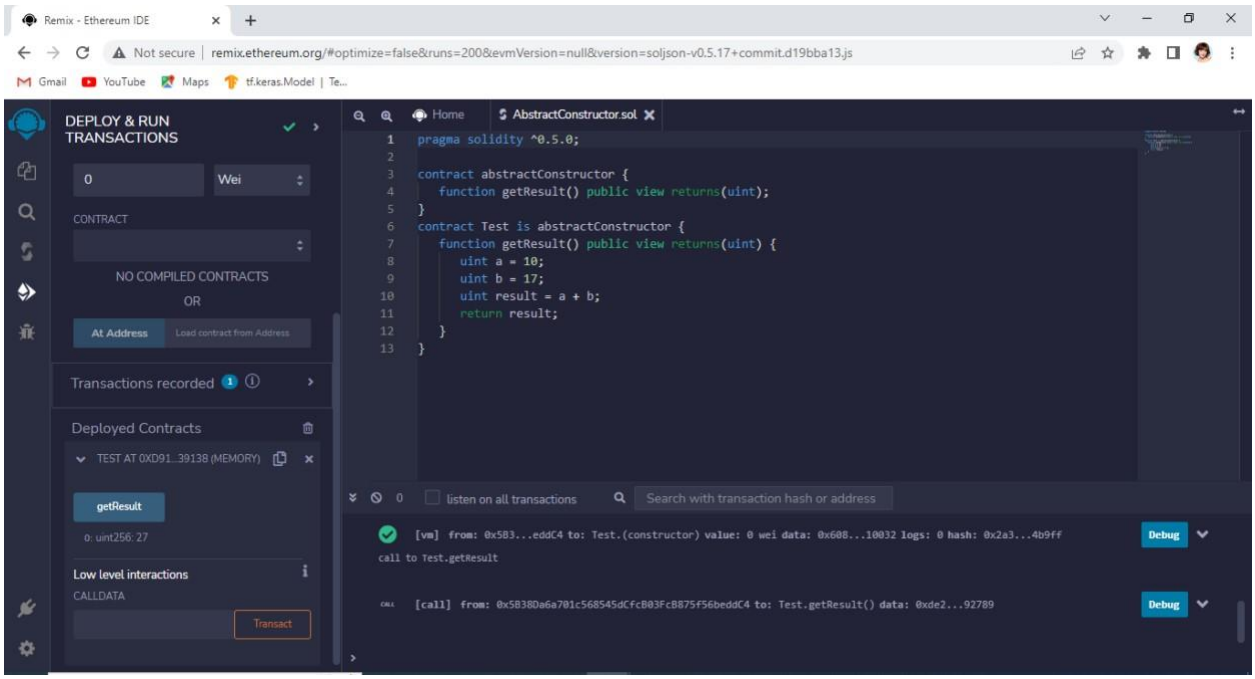

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
Gmail
YouTube
Maps
tf.keras.Model | Te...
DEPLOY & RUN TRANSACTIONS
CONTRACT
constructorExample - contracts/Cons
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 20
Deployed Contracts
CONSTRUCTOREXAMPLE AT 0X1C
getValue
0: string: This is a example of Constructor
Low level interactions
CALLDATA
Transact
Constructors.sol
Inheritance.sol
pragma solidity ^0.5.0;
contract constructorExample {
string str;
constructor() public{
str="This is a example of Constructor";
}
function getValue(
) public view returns(
string memory)
{
return str;
}
}
ContractDefinition C
4 reference(s)
listen on all transactions
Search with transaction hash or address
creation of constructorExample pending...
[vm] from: 0x583...eddC4 to: constructorExample.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0xe3c...18b9c
Debug
call to constructorExample.getValue
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: constructorExample.getValue() data: 0x209...65255
Debug

Abstract Contracts

Code:

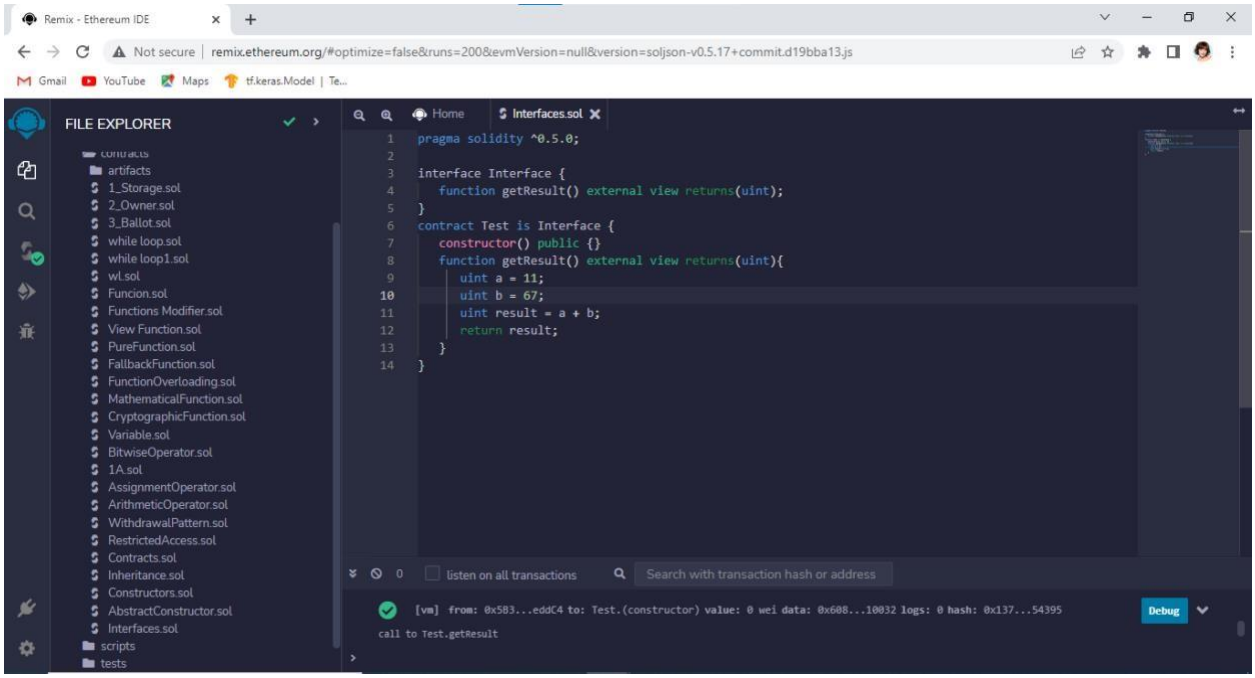


Output:

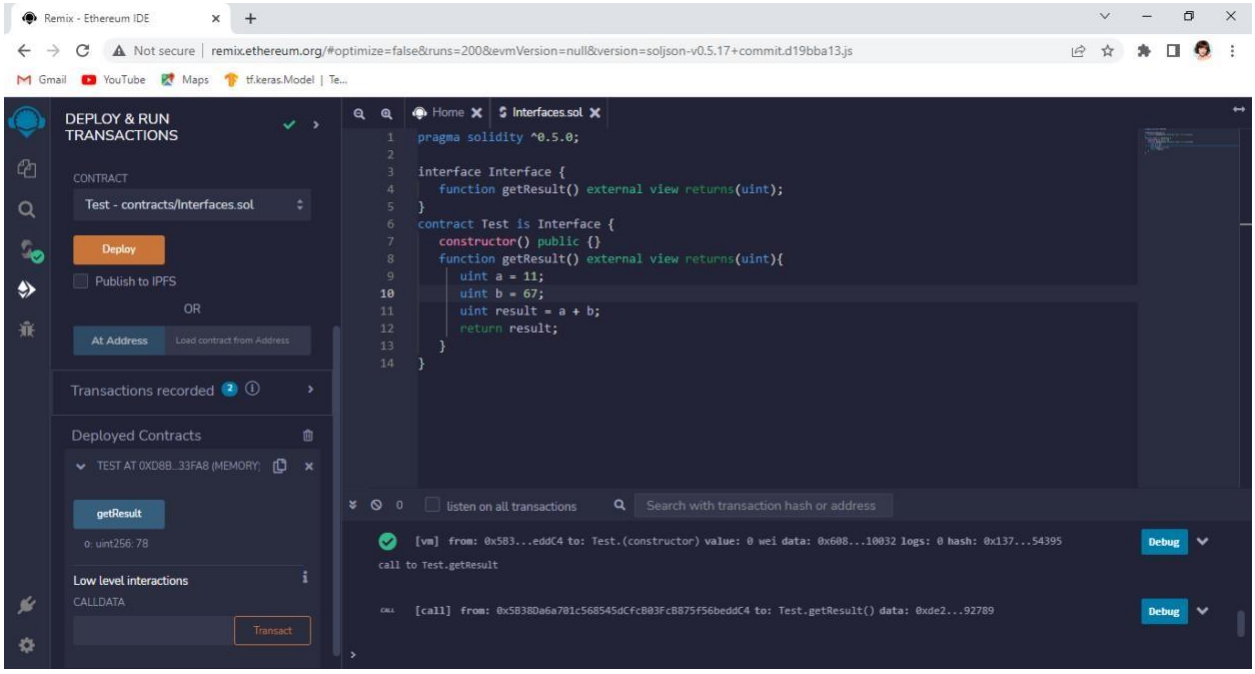


Interfaces

Code:



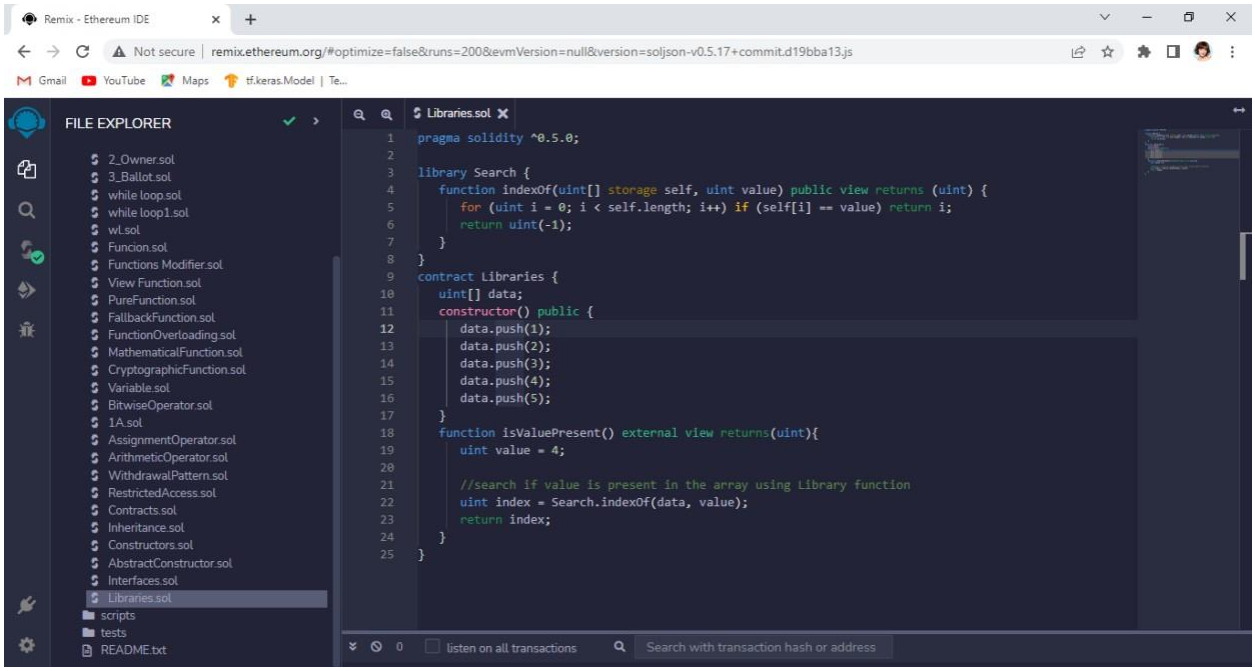
Output:



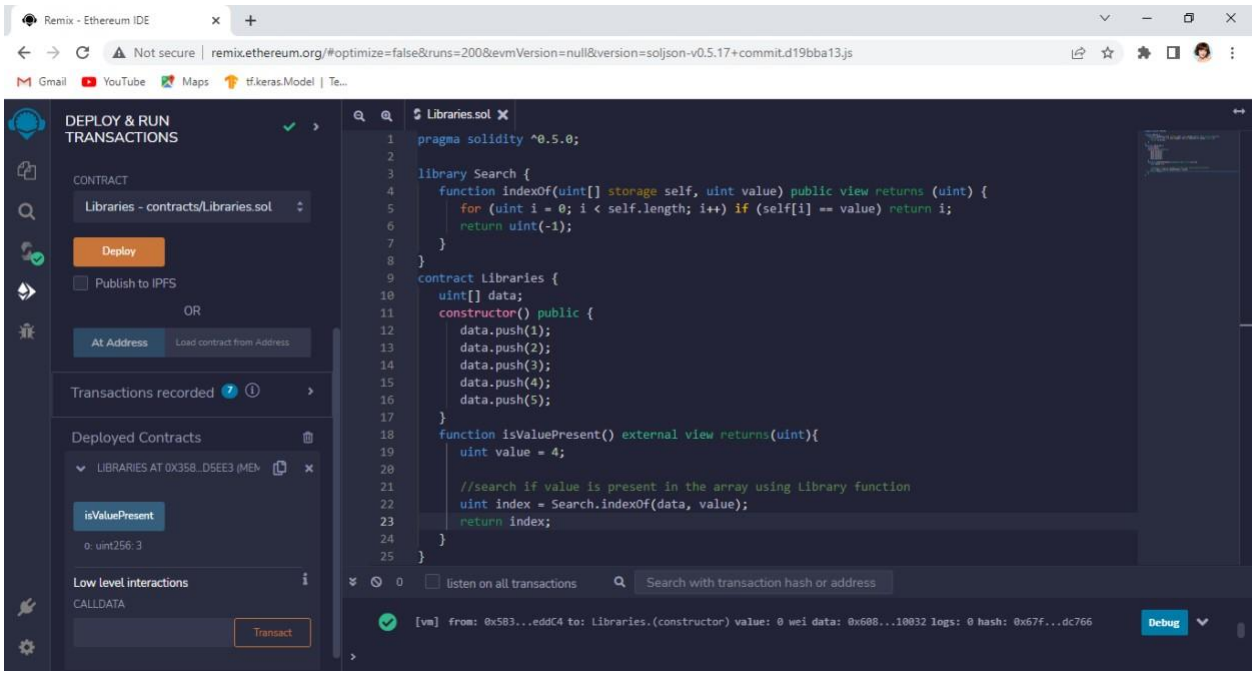
3c) Libraries, Assembly, Events, Error handling.

Libraries

Code:

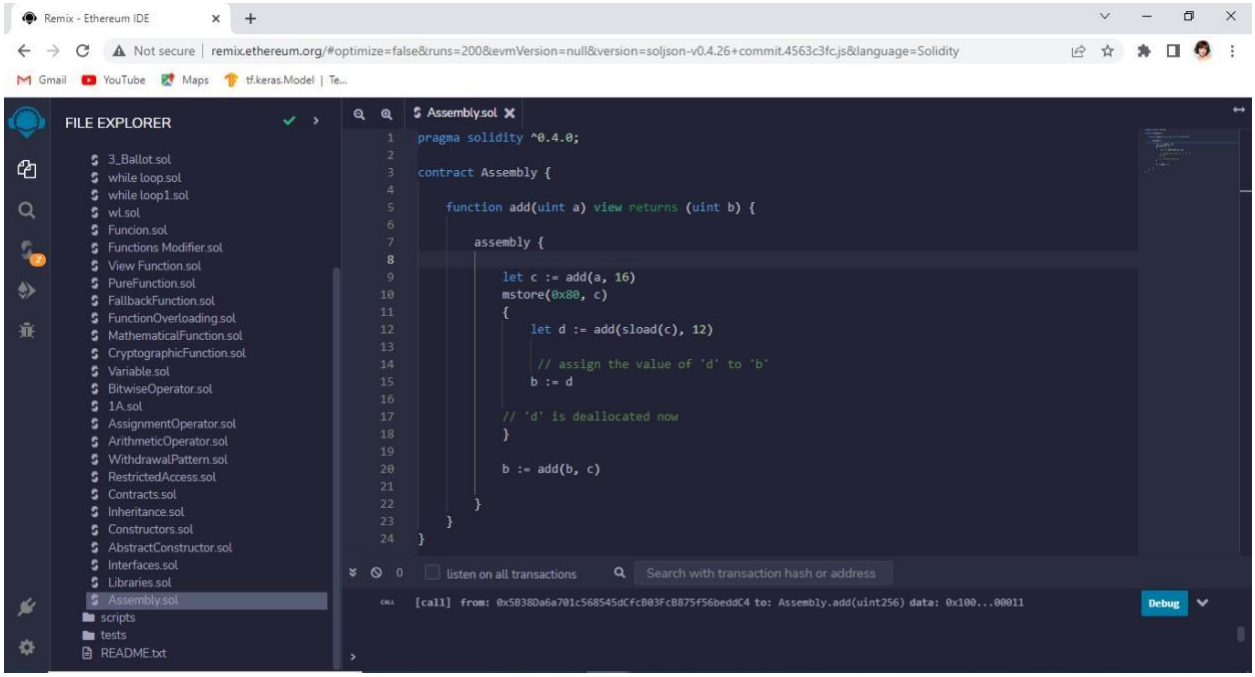


Output:

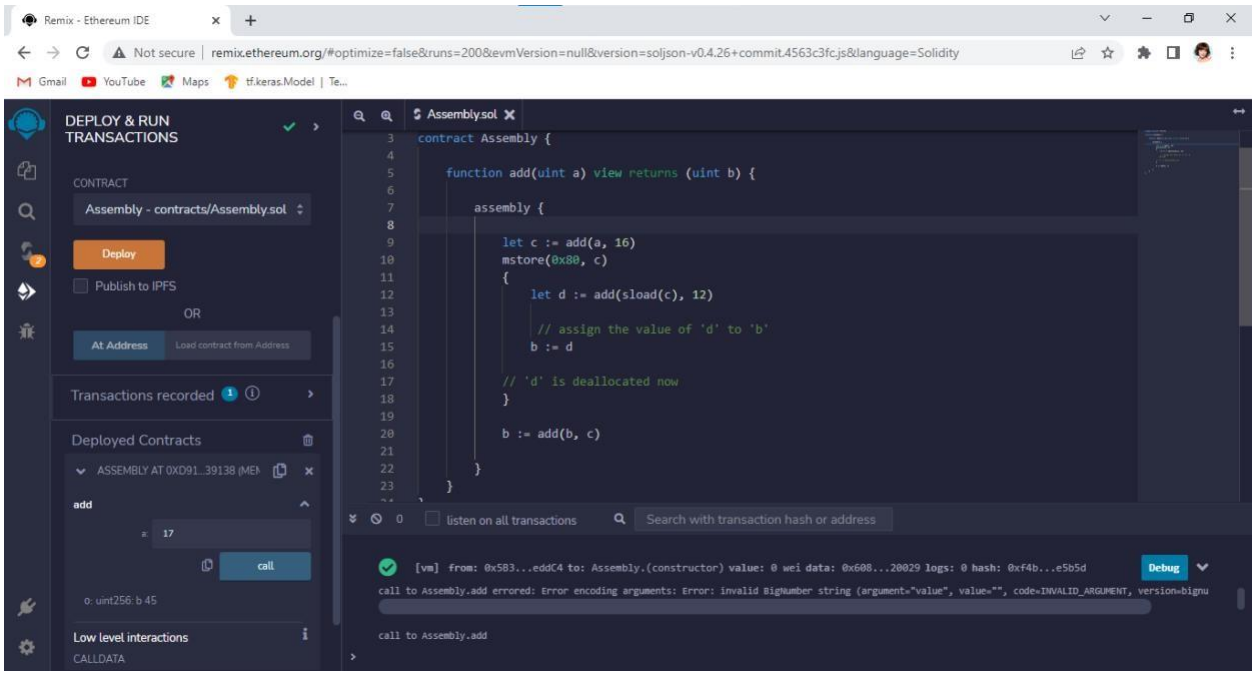


Assembly

Code:

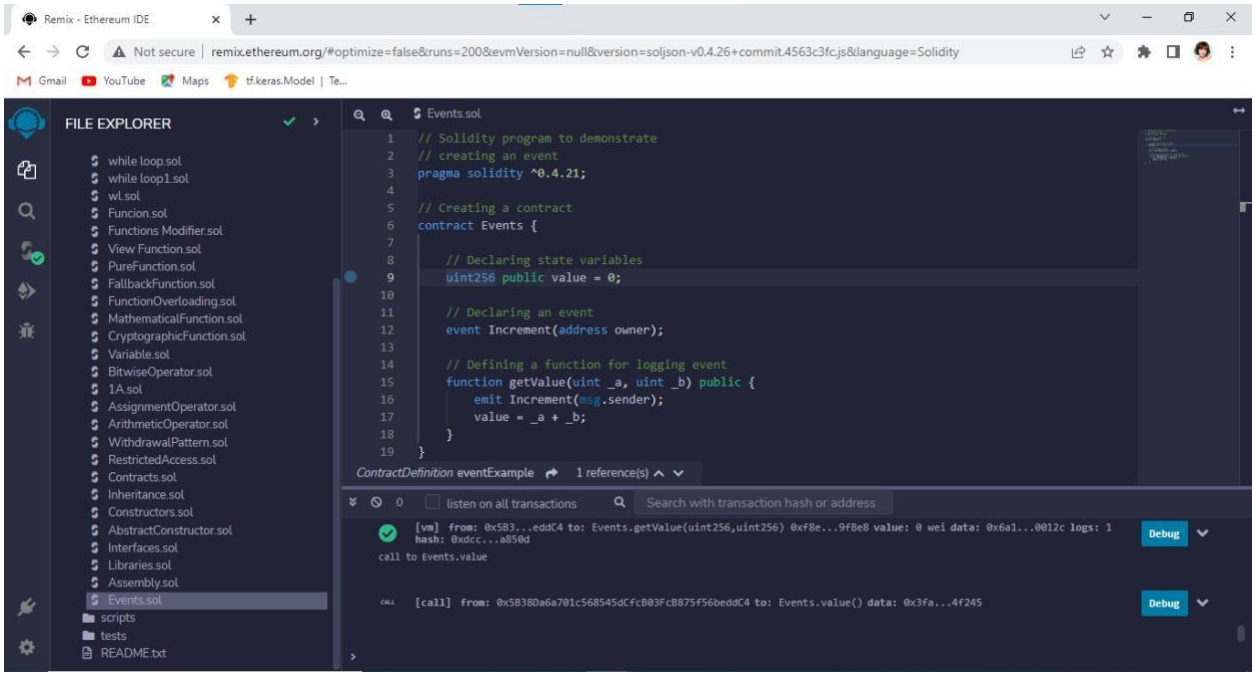


Output:

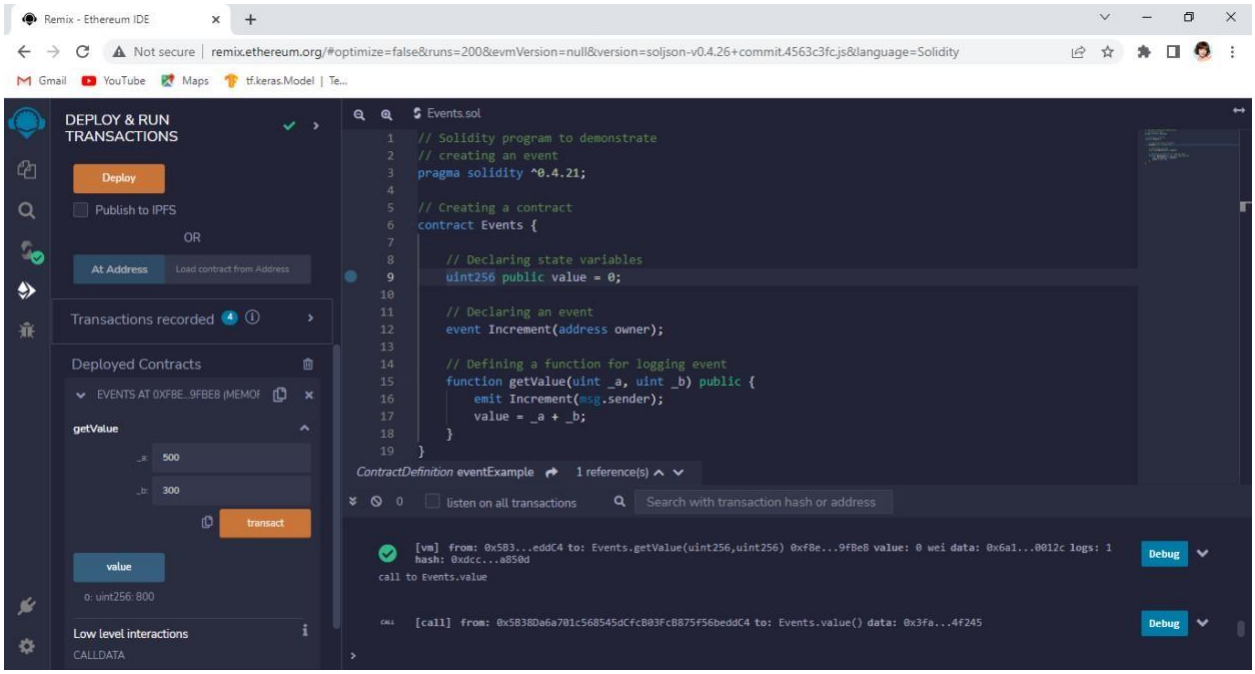


Events

Code:

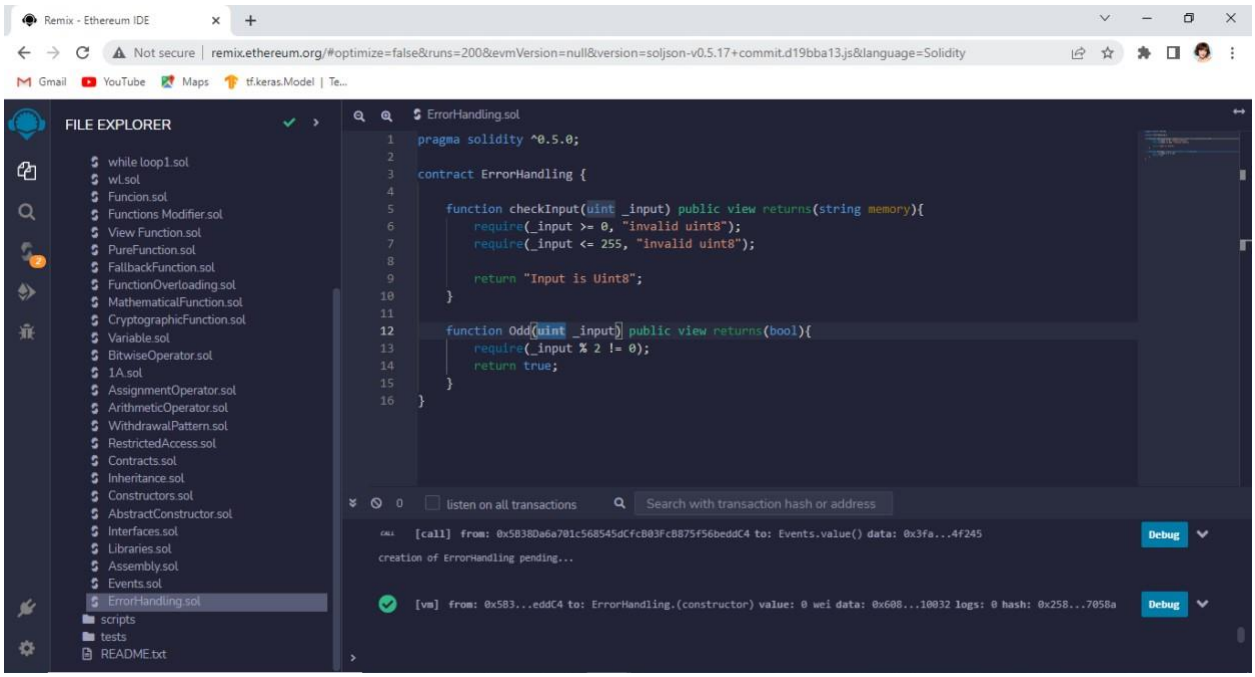


Output:

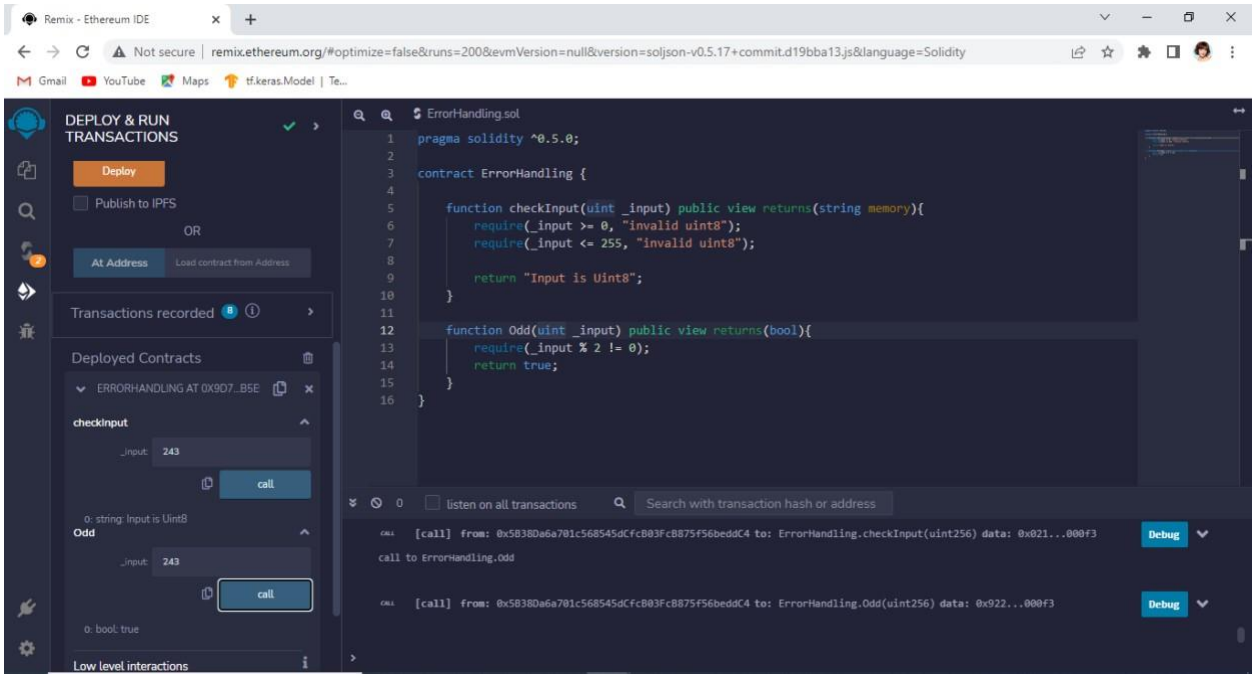


Error Handling

Code:


Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity
Gmail YouTube Maps tf.keras.Model | Te...
FILE EXPLORER
while loop1.sol
wl.sol
Funcion.sol
Functions Modifier.sol
View Function.sol
PureFunction.sol
FallbackFunction.sol
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
Variable.sol
BitwiseOperator.sol
1A.sol
AssignmentOperator.sol
ArithmeticOperator.sol
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
Inheritance.sol
Constructors.sol
AbstractConstructor.sol
Interfaces.sol
Libraries.sol
Assembly.sol
Events.sol
ErrorHandling.sol
scripts
tests
README.txt
ErrorHandling.sol
pragma solidity ^0.5.0;
contract ErrorHandling {
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;
}
}
listen on all transactions
Search with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Events.value() data: 0x3fa...4f245
creation of ErrorHandling pending...
[vm] from: 0x583...eddC4 to: ErrorHandling.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x258...7058a
Debug

Output:


Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity
Gmail YouTube Maps tf.keras.Model | Te...
DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
ERRORHANDLING AT 0X9D7...B5E
checkInput
_input: 243
call
0: string: Input is Uint8
Odd
_input: 243
call
0: bool: true
Low level interactions
ErrorHandling.sol
pragma solidity ^0.5.0;
contract ErrorHandling {
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;
}
}
listen on all transactions
Search with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.checkInput(uint256) data: 0x021...000f3
call to ErrorHandling.Odd
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.Odd(uint256) data: 0x922...000f3
Debug

Practical No: 4

Aim: Install hyperledger fabric and composer. Deploy and execute the application.

Program Steps:

The following are prerequisites for installing the required development tools:

- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (Note: version 9 is not supported)
- npm: v5.x
- git: 2.9.x or higher
- Python: 2.7.x
- A code editor of your choice, we recommend VSCode.

If installing Hyperledger Composer using Linux, the following points need to be kept in mind:

- Login as a normal user, rather than root.
- Do not use sudo su to root.
- When installing prerequisites, use curl, then unzip using sudo.
- Run prereqs-ubuntu.sh as a normal user. It may prompt for root password as some of its actions are required to be run as root.
- Do not use npm with sudo or su to root to use it.
- Avoid installing node globally as root.

Prerequisites

To download prerequisites for Hyperledger Fabric development, run following command – | curl -O
https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh
This command will download and install -
| docker-compose , docker-engine, npm, git, python etc

- Give permissions — chmod u+x prereqs-ubuntu.sh
- Run Script — ./prereqs-ubuntu.sh (restart system after it)
- Essential CLI tools — npm install -g composer-cli@0.20

Steps

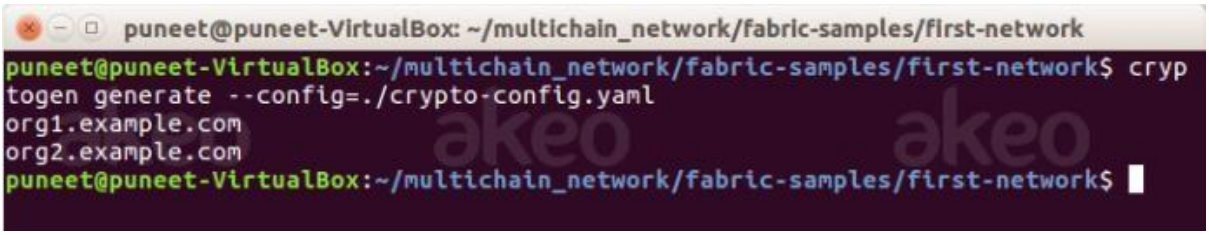
1. **Create a directory — mkdir multichain_network**

```
cd multichain_network

curl -sSL http://bit.ly/2ysbOFE | bash -s 1.2.0

export PATH=<path to download location>/multichain_network/fabric-
samples/first-network/bin:$PATH
```

2. **Generate Certificates**

```
cd first-network

export FABRIC_CFG_PATH=$PWD

cryptogen generate — config=./crypto-config.yaml
```

This will create all certificates for orderers and peers in crypto-config folder.

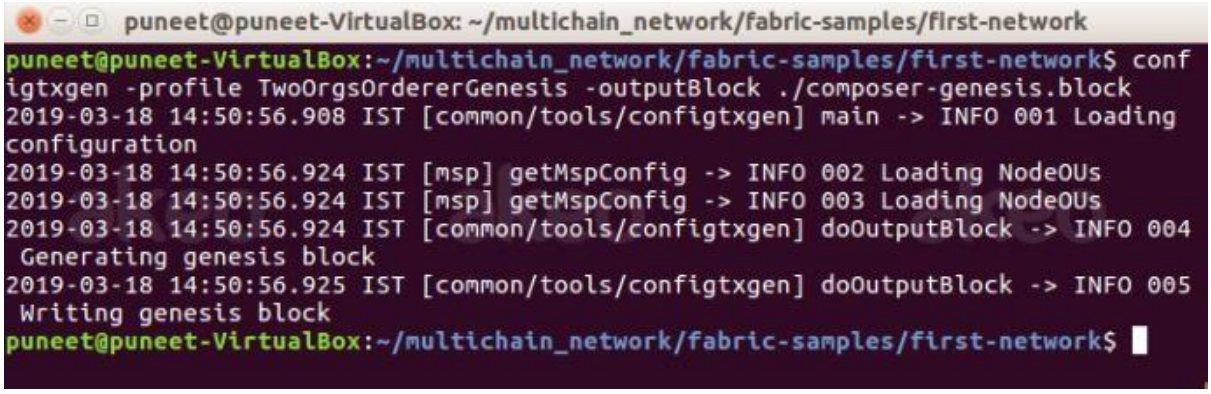


Copy certificates for all peers and orderer to temporary folder.

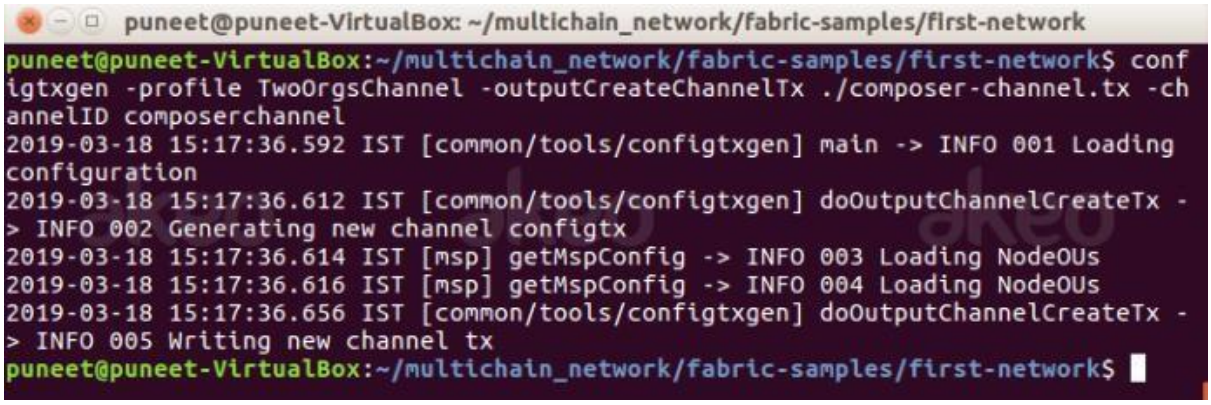
```
In first-network folder run this command — mkdir -p tmp/composer/org1

awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' ./crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com
/tls/ca.crt > ./tmp/composer/org1/ca-org1.txt

awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' ./crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.com/t
ls/ca.crt > ./tmp/composer/ca-orderer.txt

export ORG1=./crypto-
config/peerOrganizations/org1.example.com/users/Admin@org1.example.com
/msp

cp -p $ORG1/signcerts/A*.pem ./tmp/composer/org1

cp -p $ORG1/keystore/*_sk ./tmp/composer/org1
```

3. **Create genesis block and channeltx**

```
configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./composer-
genesis.block
```




```
configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./composer-
channel.tx -channelID composerchannel
```



4. **Update CA keys in docker composer file**

   Open project in code editor.

   We have to change CA keys in “docker-compose-e2e-template.yaml” file, therefore navigate to this file in vscode.

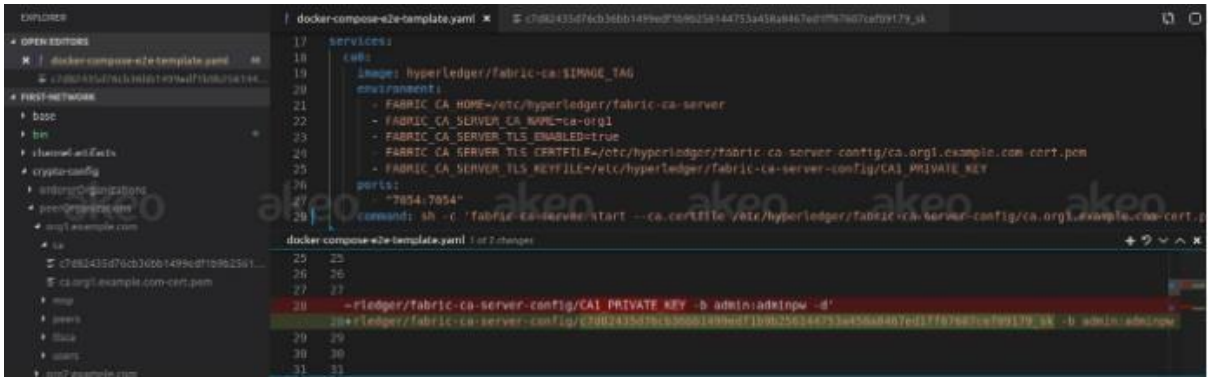
   Under services section we have two certificate authorities named “ca0” and “ca1”.

   For ca0 — go to command section under ca0 and find CA1_PRIVATE_KEY and replace it with private key –

```
C7d82435d76cb36bb1499edf1b9b256144753a458a8467ed1ff67607cef09179_sk
```

   This key can be found in –

```
"first-network/crypto-
config/peerOrganizations/org1.example.com/ca/c7d82435d76cb36bb1499edf1
b9b256144753a458a8467ed1ff67607cef09179_sk"
```

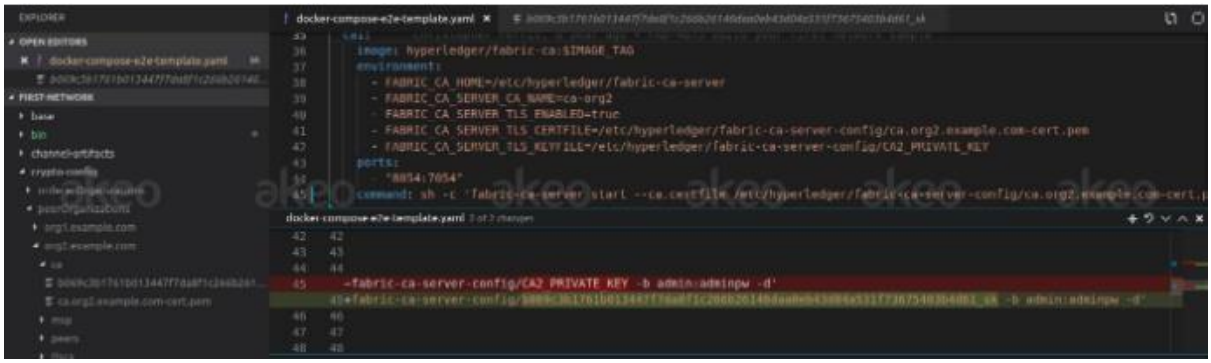


For ca1 — goto command section under ca1 and find CA2_PRIVATE_KEY and replace it with private key –

```
B069c3b1761b013447f7da8f1c266b26146daa0eb43d04a531f73675403b4d61_sk
```

This key can be found in –

```
"first-network/crypto-
config/peerOrganizations/org1.example.com/ca/b069c3b1761b013447f7da8f1
c266b26146daa0eb43d04a531f73675403b4d61_sk
```



Steps of Starting Hyperledger Fabric

1. Open docker-compose-base.yaml file which is present in the bin folder and introduce following changes.

```
Change orderer volume binding to -

../composer-
genesis.block:/var/hyperledger/orderer/orderer.genesis.block
```

As shown in the figure below –

```
orderer.example.com:
  container_name: orderer.example.com
  image: hyperledger/fabric-orderer:$IMAGE_TAG
  environment:
    - ORDERER_GENERAL_LOGLEVEL=INFO
    - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
    - ORDERER_GENERAL_GENESISMETHOD=file
    - ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block
    - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
    - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
    # enabled TLS
    - ORDERER_GENERAL_TLS_ENABLED=true
    - ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
    - ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
    - ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric
  command: orderer
  volumes:
  - ../composer-genesis.block:/var/hyperledger/orderer/orderer.genesis.block
  - ../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hyperledger/orderer/msp
  - ../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls/:/var/hyperledger/orderer/tls
  - orderer.example.com:/var/hyperledger/production/orderer
  ports:
    - 7050:7050
```

Change peer volume binding to (for all 4 peers)-

```
- ../:/etc/configtx
```

```
peer0.org1.example.com:
  extends:
    file: peer-base.yaml
    service: peer-base
  environment:
    - CORE_PEER_ID=peer0.org1.example.com
    - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org1.example.com:7051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
    - CORE_PEER_LOCALMSPID=Org1MSP
  volumes:
      - ../:/etc/configtx
      - /var/run/:/host/var/run/
      - ../crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp
      - ../crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/etc/hyperledger/fabric/tls
      - peer0.org1.example.com:/var/hyperledger/production
  ports:
    - 7051:7051
    - 7053:7053
```

2. To start fabric, run the following command –

```
docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml
up -d 2>&1
```

The output of the above command is shown below –

```
puneet@puneet-VirtualBox:~/MultiChain_network/fabric-samples/first-network$ docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml up -
d 2>&1
Creating orderer.example.com ...
Creating couchdb1 ...
Creating couchdb3 ...
Creating couchdb2 ...
Creating orderer.example.com
Creating couchdb0 ...
Creating couchdb2
Creating couchdb1
Creating couchdb3
Creating couchdb0 ... done
Creating peer0.org1.example.com ...
Creating couchdb3 ... done
Creating peer0.org2.example.com ...
Creating peer1.org1.example.com ...
Creating peer1.org2.example.com ...
Creating peer1.org1.example.com
Creating peer0.org2.example.com
Creating peer1.org2.example.com ... done
Creating cli ...
Creating cli ... done
```

3. After completing all these steps, you can run command –

```
docker ps -a
```

This will list all running containers regarding our network setup, in our case it will list 10 containers.

```
CONTAINER ID        IMAGE                               COMMAND                  CREATED             STATUS
    PORTS                                       NAMES
badfe5ea8a92        hyperledger/fabric-peer:latest      "peer node start"        4 hours ago         Up 3 hours
                                                peer1.org1.example.com
50d9fa8f5dfb        hyperledger/fabric-peer:latest      "peer node start"        4 hours ago         Up 3 hours
                                                peer0.org2.example.com
ee4b501b7d00        hyperledger/fabric-peer:latest      "peer node start"        4 hours ago         Up 3 hours
                                                peer0.org1.example.com
32f0f4426a22        hyperledger/fabric-peer:latest      "peer node start"        4 hours ago         Up 3 hours
                                                peer1.org2.example.com
ec8f9df17258        hyperledger/fabric-couchdb          "tini -- /docker-ent…"   4 hours ago         Up 4 hours
    4369/tcp, 9100/tcp, 0.0.0.0:8984->5984/tcp  couchdb3
2b4be64efcfe        hyperledger/fabric-couchdb          "tini -- /docker-ent…"   4 hours ago         Up 4 hours
    4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp  couchdb2
f922625a027e        hyperledger/fabric-couchdb          "tini -- /docker-ent…"   4 hours ago         Up 4 hours
    4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp  couchdb0
8de729e174b0        hyperledger/fabric-couchdb          "tini -- /docker-ent…"   4 hours ago         Up 4 hours
    4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp  couchdb1
18830d7ccf5e        hyperledger/fabric-tools:latest     "/bin/bash"              11 hours ago        Up 11 hours
                                                cli
c7ef15868cd5        hyperledger/fabric-orderer:latest   "orderer"                11 hours ago        Up 11 hours
    0.0.0.0:7050->7050/tcp                      orderer.example.com
```

Proposed network setup is complete, our network have -

- One orderer
- Two Organizations
- Four peers (two peers on each organization)
- Couchdb for all peers

4. Creating channel

```
docker exec peer0.org1.example.com peer channel create -o
orderer.example.com:7050 -c composerchannel -f /etc/configtx/composer-
channel.tx — tls true — cafile /etc/configtx/crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.com/m
sp/tlscacerts/tlsca.example.com-cert.pem
```

5. Joining first peer where channel is created —

```
docker exec -e

"CORE_PEER_MSPCONFIGPATH=/etc/configtx/tmp/composer/org1/Admin@org1.ex
ample.com/msp" peer0.org1.example.com peer channel join -b composer-
genesis.block
```

6. For other peers, we have to first fetch the config of block from orderer

For Fetching –

```
docker exec -e

"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example
.com/msp"

peer1.org1.example.com peer channel fetch config -o

orderer.example.com:7050 -c TwoOrgsChannel
```

For joining channel –

```
docker exec -e

"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example
.com/msp"

peer1.org1.example.com peer channel join -b
composerchannel_config.block
```

Install chainCode on every machine

```
composer network install -a device-network.bna -c

PeerAdmin@multi_org1
```

Start chaincode on one machine

```
composer network start -n device-network -V 0.0.2-
deploy.${bna_deployment_version} -A admin -S adminpw -c
PeerAdmin@multi_org1
```

Conclusion

We learnt about installing Hyperledger Fabric development tools and Hyperledger Composer using Linux. We also successfully deployed a Hyperledger Fabric network having one orderer, two organizations and two peers in each organization.

Practical No: 5

Aim: Demonstrate the running of the blockchain node.

Refer Practical No: 1, 2 3

Practical No: 6

Aim: Demonstrate the use of Bitcoin Core API.

Code:

```
from hashlib import sha256
MAX_NONCE = 100000000000
 def
SHA25
6(text):
   return sha256(text.encode("ascii")).hexdigest()
 def mine(block_number, transactions, previous_hash, prefix_zeros):
   prefix_str = '0'*prefix_zeros     for
nonce in range(MAX_NONCE):
       text = str(block_number) + transactions + previous_hash + str(nonc e)         new_hash =
SHA256(text)        if new_hash.startswith(prefix_str):
          print(f"Yay! Successfully mined bitcoins with nonce value:{non ce}")             return
new_hash

   raise BaseException(f"Couldn't find correct has after trying {MAX_NONC
E} times")  if
__name__=='__m
ain__':
transactions='''
Dhaval->Bhavin-
>20,
   Mando->Cara->45
   '''     difficulty=4 # try changing this to higher number and you will see it will take more
time for mining as difficulty increases     import time     start = time.time()     print("start
mining")
   new_hash = mine(5,transactions,'0000000xa036944e29568d0cff17edbe038f81
208fecf9a66be9a2b8321c6ec7', difficulty)     total_time =
str((time.time() - start))     print(f"end mining. Mining took:
{total_time} seconds")     print(new_hash)
```

Output:

```
start mining
Yay! Successfully mined bitcoins with nonce value:2425
end mining. Mining took: 0.011358022689819336 seconds
0000de957fbfdfc77582e0d0b20c53d2d1d83d8bb8cfe3693521f672bf2a6021
```

Practical No: 7

Aim: Create your own blockchain and demonstrate its use.

Refer Practical No: 1, 2 3

Advance IOT

1	**Displaying different LED patterns with Raspberry Pi.**

```
import RPi.GPIO as GPIO
import time

x=1
numTimes=int(input("Enter tottal number of times to blink"))
speed=float(input("Enter length of each blink(seconds) : "))

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(5,GPIO.OUT)
GPIO.setup(10,GPIO.OUT)
GPIO.setup(19,GPIO.OUT)
GPIO.setup(26,GPIO.OUT)
GPIO.setup(29,GPIO.OUT)

def Blink(numTimes,speed):
   for i in range(0,numTimes):
      GPIO.output(5,True)
      print ("Iteration ", (i+1))


      GPIO.output(10,True)
      print ("Iteration ", (i+1))


      GPIO.output(19,True)
      print ("Iteration ", (i+1))


      GPIO.output(26,True)
      print ("Iteration ", (i+1))


      GPIO.output(29,True)
      print ("Iteration ", (i+1))


      GPIO.output(29,False)
      print ("Iteration ", (i+1))
      time.sleep(speed)

      GPIO.output(26,False)
      print ("Iteration ", (i+1))
      time.sleep(speed)

      GPIO.output(19,False)
      print ("Iteration ", (i+1))
      time.sleep(speed)
```

```
        GPIO.output(10,False)
        print ("Iteration ", (i+1))
        time.sleep(speed)

        GPIO.output(5,False)
        print ("Iteration ", (i+1))
        time.sleep(speed)

Blink(numTimes,speed)
print("Done")
```

2	**Displaying Time over 4-Digit 7-Segment Display using Raspberry Pi.**

Control 4 digits-7 segments LED display with TM1637 controller

Connection scheme Raspberry Pi

Connect the LED to your Raspberry Pi according to the following diagram:

TM1637 script

In order to control the LED, we use a special script with pre-defined functions. Various functions are available in the script, for example you can display numbers and adjust the intensity of the LEDs. Download the script with the command:

```
wget https://raspberrytips.nl/files/tm1637.py
```

Code:

```
import sys
import time
import datetime
import RPi.GPIO as GPIO
import tm1637

#CLK -> GPIO23 (Pin 16)
#Di0 -> GPIO24 (Pin 18)

Display = tm1637.TM1637(23,24,tm1637.BRIGHT_TYPICAL)

Display.Clear()
Display.SetBrightnes(1)

while(True):
  now = datetime.datetime.now()
  hour = now.hour
  minute = now.minute
  second = now.second
  currenttime = [ int(hour / 10), hour % 10, int(minute / 10), minute % 10 ]

  Display.Show(currenttime)
  Display.ShowDoublepoint(second % 2)

  time.sleep(1)
```

3	**Raspberry Pi Based Oscilloscope**

Project Requirements

The requirement for this project can be classified into two:

1. Hardware Requirements
2. Software Requirements

Hardware requirements

To build this project, the following components/part are required;

1. Raspberry pi 2 (or any other model)
2. 8 or 16GB SD Card
3. LAN/Ethernet Cable
4. Power Supply or USB cable
5. ADS1115 ADC
6. LDR (Optional as its meant for test)
7. 10k or 1k resistor
8. Jumper wires
9. Breadboard
10. Monitor or any other way of seeing the pi's Desktop(VNC inclusive)

Software Requirements

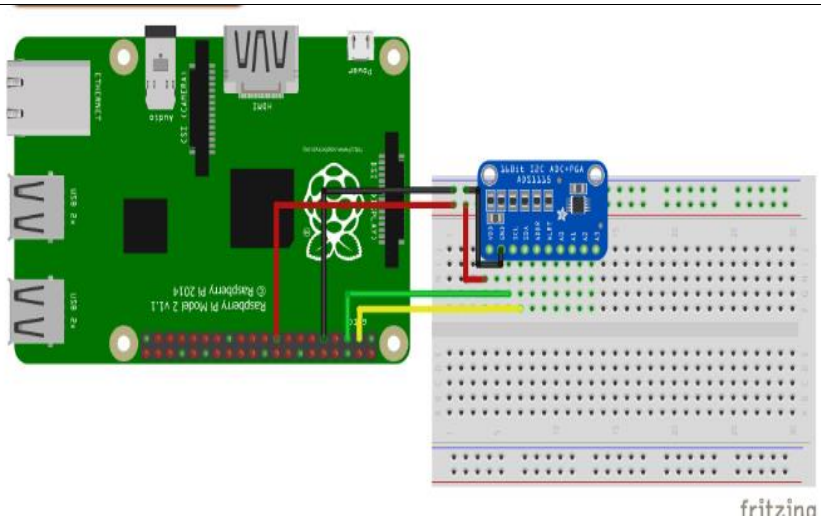
The software requirements for this project are basically the python modules (***matplotlib and drawnow***) that will be used for data visualization and the Adafruit module for interfacing with the ADS1115 ADC chip. I will show how to install these modules on the Raspberry Pi as we proceed.

While this tutorial will work irrespective of the raspberry pi OS used, I will be using the Raspberry Pi stretch OS and I will assume you are familiar with setting up the Raspberry Pi with the Raspbian stretch OS, and you know how to SSH into the raspberry pi using a terminal software like putty. If you have issues with any of this, there are tons of Raspberry Pi Tutorials on this website that can help.

With all the hardware components in place, let's create the schematics and connect the components together.

Circuit Diagram:

To convert the analog input signals to digital signals which can be visualized with the Raspberry Pi, we will be using the **ADS1115 ADC chip**. This chip becomes important because the Raspberry Pi, unlike Arduino and most micro-controllers, does not have an on-board analog to digital converter(ADC). While we could have used any raspberry pi compatible ADC chip, I prefer this chip due to its high resolution(16bits) and its well documented datasheet and use instructions by Adafruit. You can also check our Raspberry Pi ADC tutorial to learn more about it.



ADS1115 and Raspberry Pi Connections:

VDD – 3.3v

GND – GND

SDA – SDA

SCL – SCL

With the connections all done, power up your pi and proceed to install the dependencies mentioned below.

Install Dependencies for Raspberry Pi Oscilloscope:

Before we start writing the python script to pull data from the ADC and plot it on a live graph, we need to **enable the I2C communication interface** of the raspberry pi and install the software requirements that were mentioned earlier. This will be done in below steps so its easy to follow:

Step 1: Enable Raspberry Pi I2C interface

To enable the I2C, from the terminal, run;

```
sudo raspi-config
```

When the configuration panels open, select interface options, select I2C and click enable.

Step 2: Update the Raspberry pi

The first thing I do before starting any project is updating the Pi. Through this, I am sure every thing on the OS is up to date and I won't experience compatibility issue with any latest software I choose to install on the Pi. To do this, run below two commands:

```
sudo apt-get update
sudo apt-get upgrade
```

Step 3: Install the Adafruit ADS1115 library for ADC

With the update done, we are now ready to install the dependencies starting with the Adafruit python module for the ADS115 chip. Ensure you are in the Raspberry Pi home directory by running;

cd ~

then install the build-essentials by running;

sudo apt-get install build-essential python-dev python-smbus git

Next, clone the Adafruit git folder for the library by running;

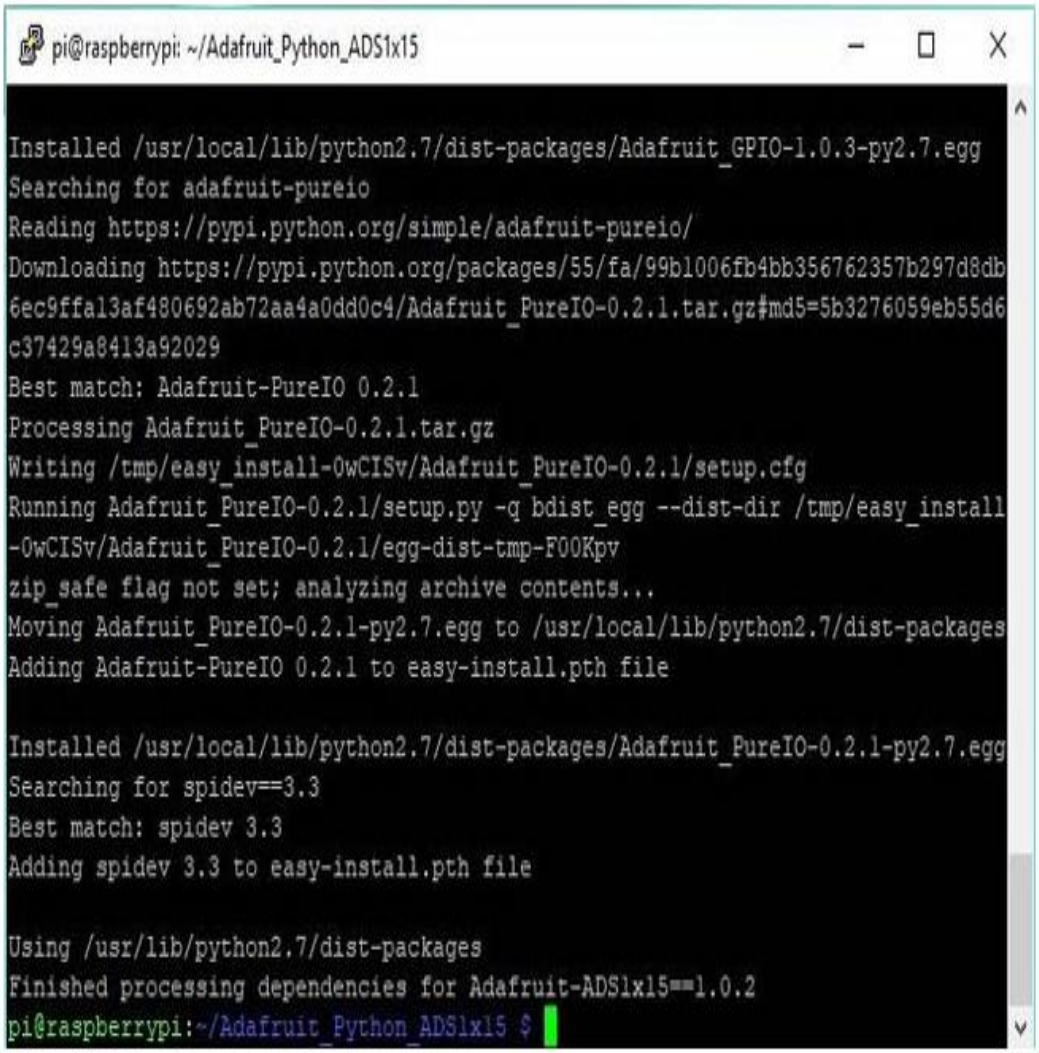
git clone https://github.com/adafruit/Adafruit_Python_ADS1x15.git

Change into the cloned file's directory and run the setup file;

cd Adafruit_Python_ADS1x1z

sudo python setup.py install

After installation, your screen should look like the image below.



Step 4: Test the library and 12C communication.

Before we proceed with the rest of the project, it is important to test the library and ensure the ADC can communicate with the raspberry pi over I2C. To do this we will use an example script that comes with the library.

While still in the Adafruit_Python_ADS1x15 folder, change directory to the examples directory by running;

cd examples

Next, run the sampletest.py example which displays the value of the four channels on the ADC in a tabular form.

Run the example using:

python simpletest.py

If the I2C module is enabled and connections good, you should see the data as shown in the image below.

```
pi@raspberrypi:~ $ cd Adafruit_Python_ADS1x15
pi@raspberrypi:~/Adafruit_Python_ADS1x15 $ cd examples
pi@raspberrypi:~/Adafruit_Python_ADS1x15/examples $ python simpletest.py
Reading ADS1x15 values, press Ctrl-C to quit...
|      0 |      1 |      2 |      3 |
-----------------------------------
|   4699 |   4584 |   4625 |   4665 |
|   4583 |   4587 |   4601 |   4614 |
|   4563 |   4604 |   4600 |   4612 |
|   4601 |   4630 |   4609 |   4585 |
|   4614 |   4606 |   4577 |   4636 |
|   4616 |   4580 |   4621 |   4630 |
|   4566 |   4630 |   4618 |   4631 |
|   4614 |   4619 |   4615 |   4620 |
|   4577 |   4622 |   4609 |   4625 |
|   4624 |   4615 |   4626 |   4648 |
|   4636 |   4660 |   4656 |   4607 |
|   4609 |   4616 |   4629 |   4651 |
```

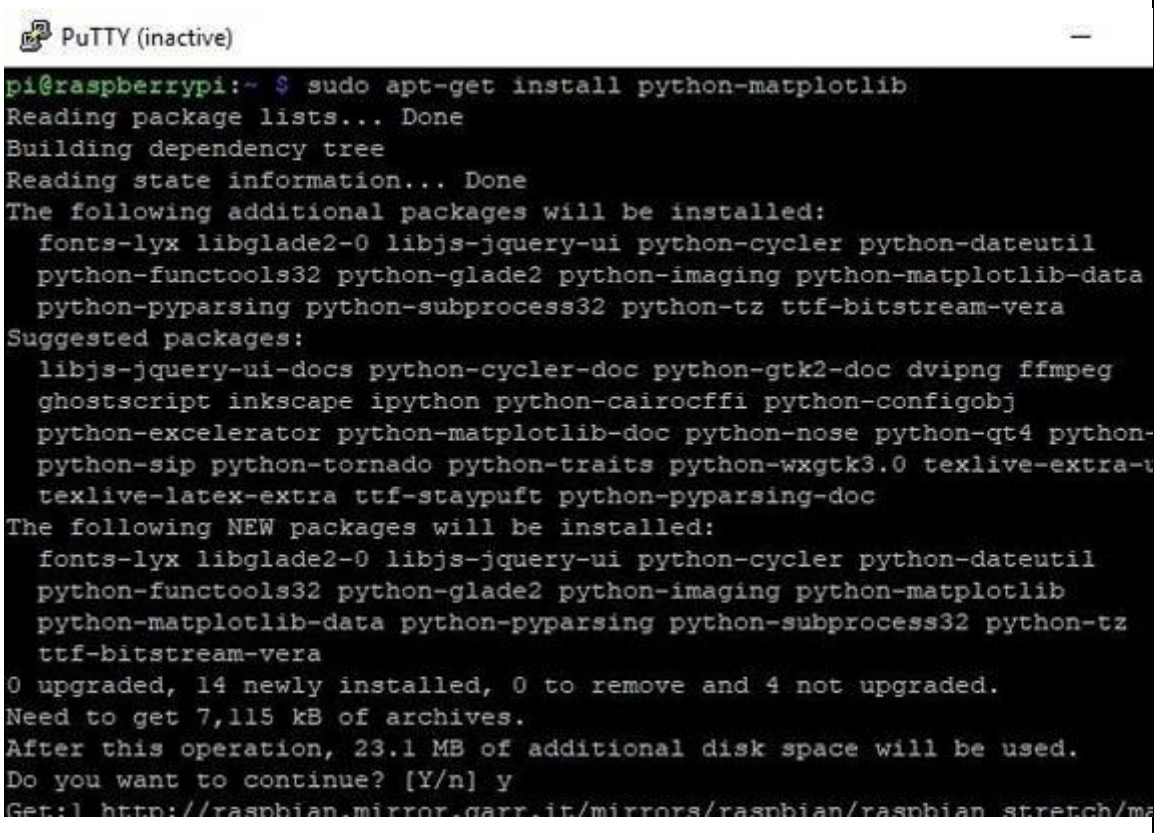
If an error occurs, check to ensure the ADC is well connected to the PI and I2C communication is enabled on the Pi.

Step 5: Install *Matplotlib*

To visualize the data we need to install the *matplotlib* module which is used to plot all kind of graphs in python. This can be done by running;

sudo apt-get install python-matplotlib

You should see an outcome like the image below.



Step6: Install the *Drawnow* python module

Lastly, we need to install the *drawnow* python module. This module helps us provide live updates to the data plot.

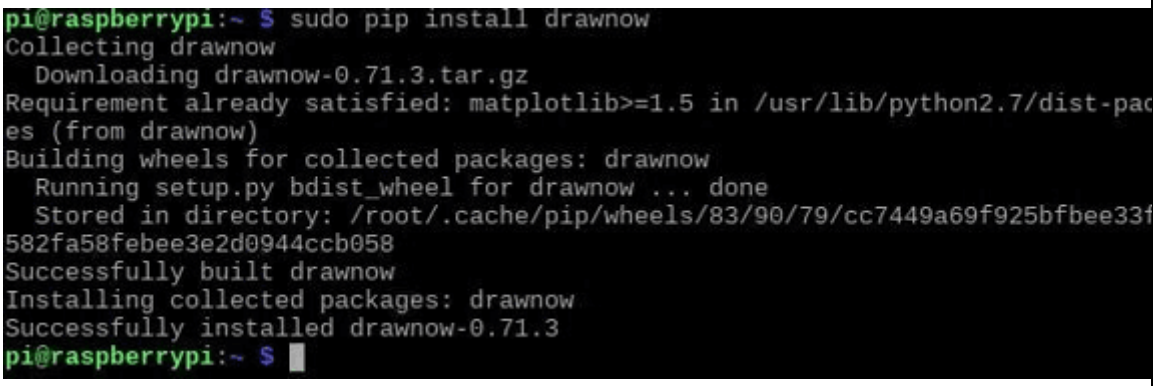
We will be installing *drawnow* via the python package installer; *pip*, so we need to ensure it is installed. This can be done by running;

```
sudo apt-get install python-pip
```

We can then use pip to install the *drawnow* package by running:

```
sudo pip install drawnow
```

You should get an outcome like the image below after running it.



With all the dependencies installed, we are now ready to write the code.

Python Code for Raspberry Pi Oscilloscope:

The python code for this **Pi Oscilloscope** is fairly simple especially if you are familiar with the python *matplotlib* module. Before showing us the whole code, I will try to break it into part and explain what each part of the code is doing so you can have enough knowledge to extend the code to do more stuffs.

At this stage it is important to switch to a monitor or use the VNC viewer, anything through which you can see your Raspberry Pi's desktop, as the graph being plotted won't show on the terminal.

With the monitor as the interface **open a new python file**. You can call it any name you want, but I will call it scope.py.

```
sudo nano scope.py
```

With the file created, the first thing we do is import the modules we will be using;

```
import time
import matplotlib.pyplot as plt
from drawnow import *
```

import Adafruit_ADS1x15

Next, we **create an instance of the ADS1x15 library** specifying the ADS1115 ADC

adc = Adafruit_ADS1x15.ADS1115()

Next, we set the gain of the ADC. There are different ranges of gain and should be chosen based on the voltage you are expecting at the input of the ADC. For this tutorial, we are estimating a 0 – 4.09v so we will be using a gain of 1. For more info on gain you can check the ADS1015/ADS1115 datasheet.

GAIN = 1

Next, we need to create the array variables that will be used to store the data to be plotted and another one to serve as count.

Val = [ ]

cnt = 0

Next, we make know our intentions of making the plot interactive known so as to **enable us plot the data live**.

plt.ion()

Next, we start continuous ADC conversion **specifying the ADC channel**, in this case, channel 0 and we also specify the gain.

It should be noted that all the four ADC channels on the ADS1115 can be read at the same time, but 1 channel is enough for this demonstration.

adc.start_adc(0, gain=GAIN)

Next we create a function *def makeFig*, to **create and set the attributes of the graph** which will hold our live plot. We first of all set the limits of the y-axis using *ylim*, after which we input the title of the plot, and the label name before we specify the data that will be plotted and its plot style and color using *plt.plot()*. We can also state the channel (as channel 0 was stated) so we can identify each signal when the four channels of the ADC are being used. *plt.legend* is used to specify where we want the information about that signal(e.g Channel 0) displayed on the figure.

```
plt.ylim(-5000,5000)
plt.title('Osciloscope')
plt.grid(True)
plt.ylabel('ADC outputs')
plt.plot(val, 'ro-', label='lux')
plt.legend(loc='lower right')
```

Next we write the *while* loop which will be used constantly read data from the ADC and update the plot accordingly.

The first thing we do is **read the ADC conversion value**

```
value = adc.get_last_result()
```

Next we print the value on the terminal just to give us another way of confirming the plotted data. We wait a few seconds after printing then we append the data to the list (val) created to store the data for that channel.

```
print('Channel 0: {0}'.format(value))
time.sleep(0.5)
val.append(int(value))
```

We then call *drawnow* to update the plot.

```
drawnow(makeFig)
```

To ensure the latest data is what is available on the plot, we delete the data at index 0 after every 50 data counts.

```
cnt = cnt+1
if(cnt>50):
val.pop(0)
```

That's all!

The **complete Python code** is given at the end of this tutorial.

Raspberry Pi Oscilloscope in Action:

Copy the complete python code and paste in the python file we created earlier, remember we will need a monitor to view the plot so all of this should be done by either VNC or with a connected monitor or screen.

Save the code and run using;

sudo python scope.py

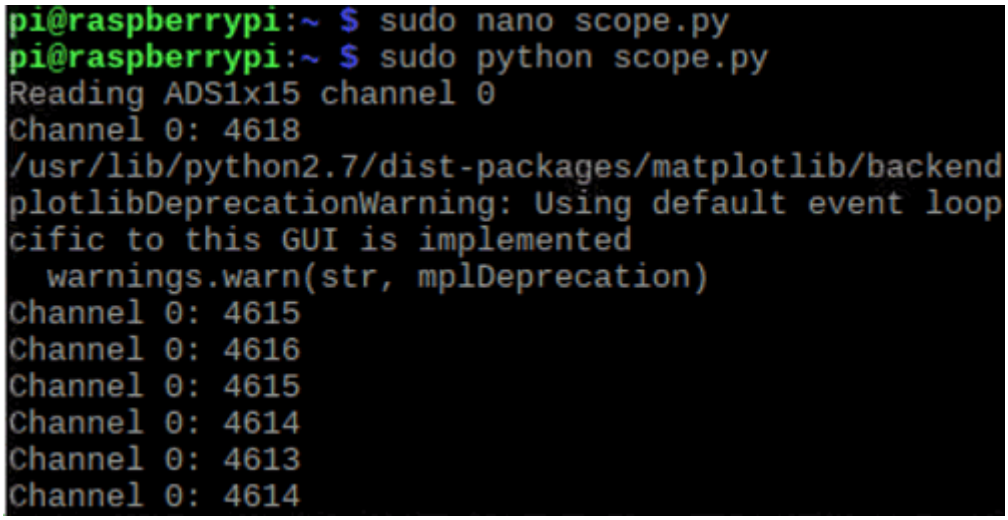
If you used a different name other than scope.py, don't forget to change this to match.

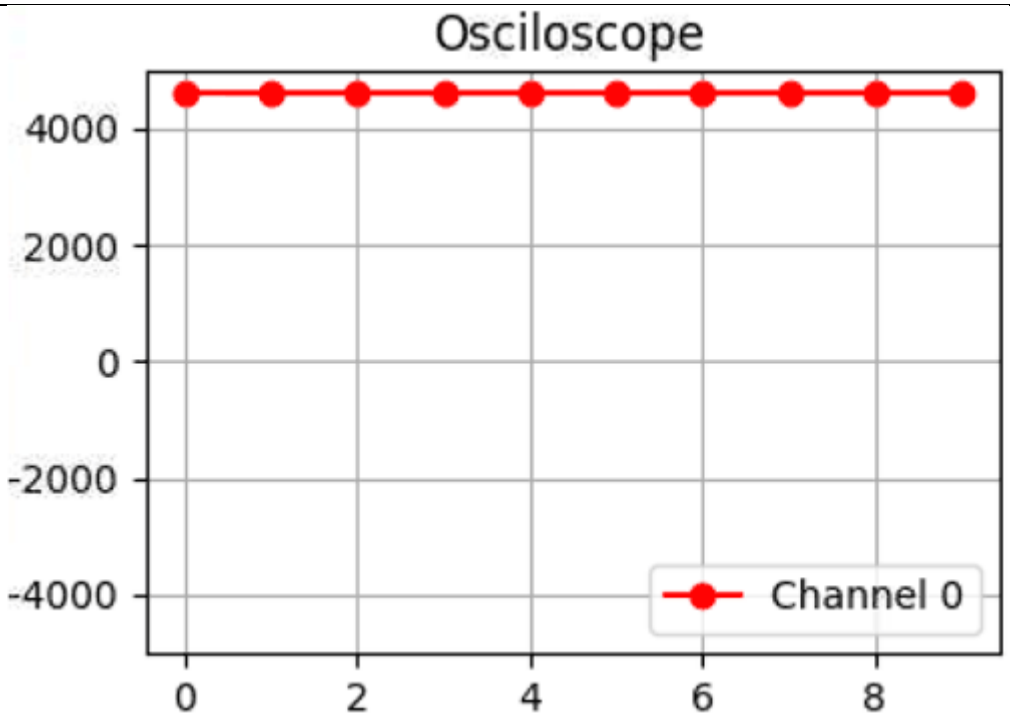
After a few minutes, you should see the ADC data being printed on the terminal. Occasionally you may get a warning from *matplotlib* (as shown in the image below) which should be suppressed but it doesn't affect the data being displayed or the plot in anyway. To suppress the warning however, the following lines of code can be added after the import lines in our code.

Import warnings

import matplotlib.cbook

warnings.filterwarnings("ignore", category=matplotlib.cbook.mplDeprecation)

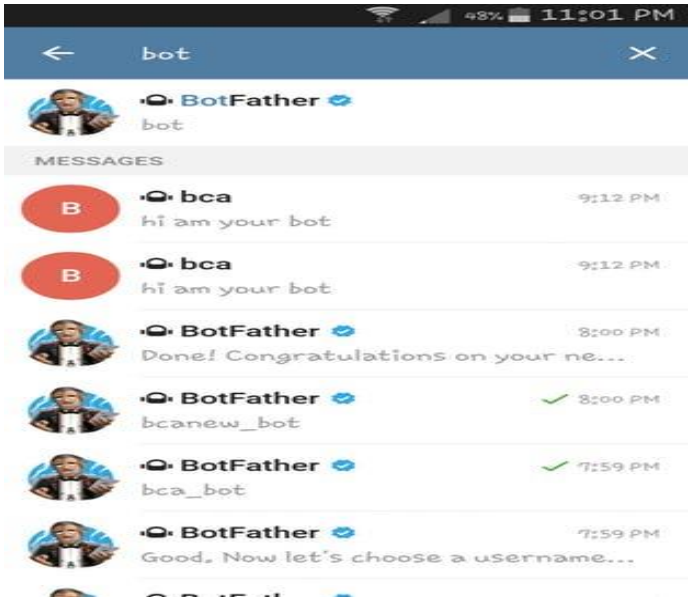




Code:

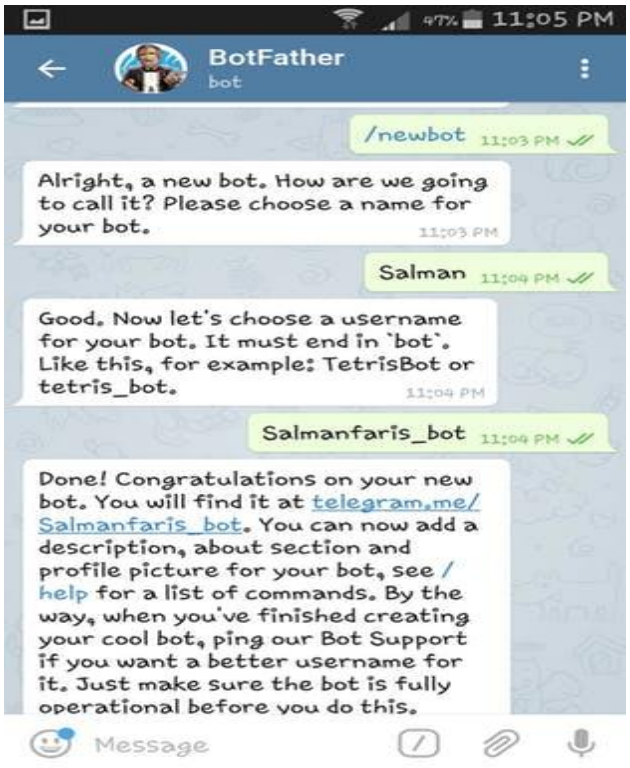
```
import time
import matplotlib.pyplot as plt
#import numpy
from drawnow import *
# Import the ADS1x15 module.
import Adafruit_ADS1x15
# Create an ADS1115 ADC (16-bit) instance.
adc = Adafruit_ADS1x15.ADS1115()

GAIN = 1
val = [ ]
cnt = 0
plt.ion()
# Start continuous ADC conversions on channel 0 using the previous gain value.
adc.start_adc(0, gain=GAIN)
print('Reading ADS1x15 channel 0')
#create the figure function
def makeFig():
    plt.ylim(-5000,5000)
    plt.title('Osciloscope')
    plt.grid(True)
    plt.ylabel('ADC outputs')
    plt.plot(val, 'ro-', label='Channel 0')
    plt.legend(loc='lower right')
while (True):
    # Read the last ADC conversion value and print it out.
    value = adc.get_last_result()
    print('Channel 0: {0}'.format(value))
    # Sleep for half a second.
```

```
    time.sleep(0.5)
    val.append(int(value))
    drawnow(makeFig)
    plt.pause(.000001)
    cnt = cnt+1
    if(cnt>50):
        val.pop(0)
```

4	**Controlling Raspberry Pi with Telegram.**
	Step 1: Open Telegram app in your system **or mobile** **1.1 Open Telegram app in your system or mobile** **1.2 Start "BotFather"**  **1.3 Open "BotFather"**  **1.4 Start "BotFather"**

/start

1.5 Create a new Bot



1.6 Obtain access token

3.3 Install "Python Package Index"

```
sudo apt-get install python-pip
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jan  4 16:25:26 2017 from 192.168.100.4
pi@raspberrypi:~ $ sudo apt-get install python-pip
```

Note: Make sure Pi has internet access

```
Last login: Wed Jan  4 16:25:26 2017 from 192.168.100.4
pi@raspberrypi:~ $ sudo apt-get install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-pip is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 81 not upgraded.
pi@raspberrypi:~ $
```

3.4 Install "telepot"

```
 sudo pip install telepot
```

```
pi@raspberrypi:~ $ sudo pip install telepot
```

Step 4: Run the Python Code

4.1 Clone the git

```
git clone https://github.com/salmanfarisvp/TelegramBot.git
```

4.2 Paste your Bot Token here

```
 bot = telepot.Bot('Bot Token')
```

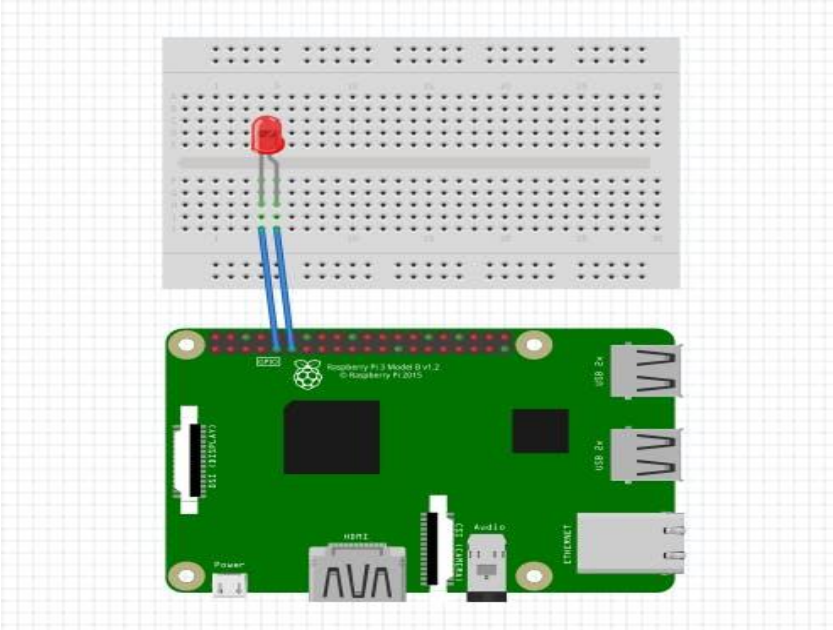
Note: 1.6 for more details

4.3 Run the Code

```
python telegrambot.py
```

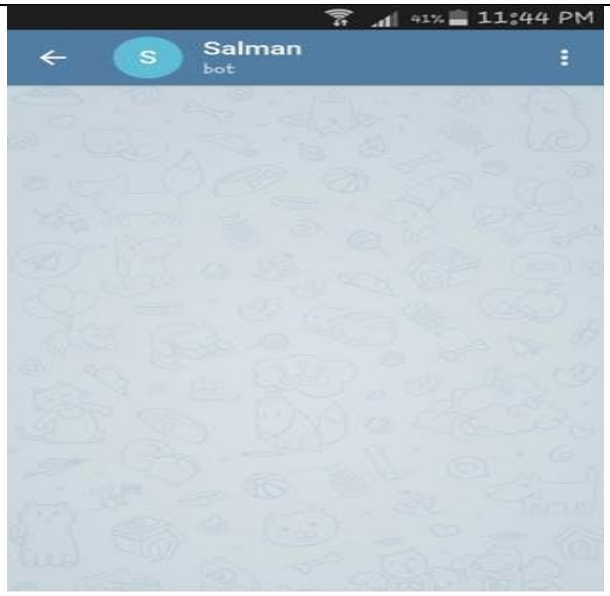
All set, now time to connect the Pi and LED.

Step 5: Connect LED to Pi

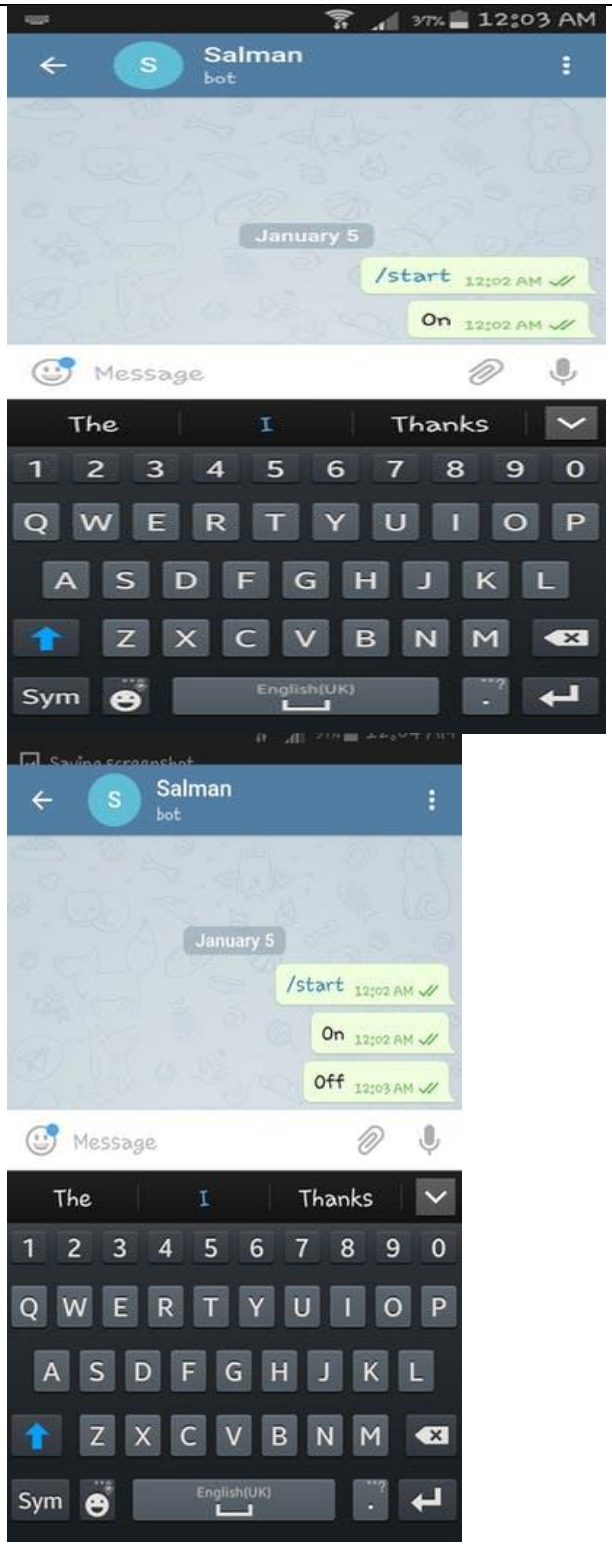


Step 6: Send Command

6.1 Start our Bot



6.2 Send "on" & "off"



Look at your Pi, you can see the LED on and off when you send "on" and "off" to our bot.

Code:

```
import sys
import time
import random
import datetime
```

```
import telepot
import RPi.GPIO as GPIO

#LED
def on(pin):
        GPIO.output(pin,GPIO.HIGH)
        return
def off(pin):
        GPIO.output(pin,GPIO.LOW)
        return
# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BOARD)
# set up GPIO output channel
GPIO.setup(11, GPIO.OUT)

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    print 'Got command: %s' % command

    if command == 'on':
       bot.sendMessage(chat_id, on(11))
    elif command =='off':
       bot.sendMessage(chat_id, off(11))

bot = telepot.Bot('Bot Token')
bot.message_loop(handle)
print 'I am listening...'

while 1:
     time.sleep(10)
```

5	**Setting up Wireless Access Point using Raspberry Pi**

Required Components:

The following components will be needed to set up a raspberry pi as a wireless access point:

1. Raspberry Pi 2
2. 8GB SD card
3. WiFi USB dongle
4. Ethernet cable
5. Power supply for the Pi.
6. Monitor (optional)
7. Keyboard (optional)
8. Mouse (optional)

Steps for Setting up Raspberry Pi as Wireless Access Point:

Step 1: Update the Pi

As usual, we update the raspberry pi to ensure we have the latest version of everything. This is done using;

```
sudo apt-get update
```

followed by;

```
sudo apt-get upgrade
```

With the update done, reboot your pi to effect changes.

Step 2: Install "*dnsmasq*" and "*hostapd*"

Next, we install the software that makes it possible to setup the pi as a wireless access point and also the software that helps assign network address to devices that connect to the AP. We do this by running;

```
sudo apt-get install dnsmasq
```

followed by;

```
sudo apt-get install hostapd
```

or you could combine it by running;

```
sudo apt-get install dnsmasq hostapd
```

Step 3: Stop the software from Running

Since we don't have the software configured yet there is no point running it, so we disable them from running in the underground. To do this we run the following commands to stop the *systemd* operation.

```
sudo systemctl stop dnsmasq
sudo systemctl stop hostapd
```

Step 4: Configure a Static IP address for the wireless Port

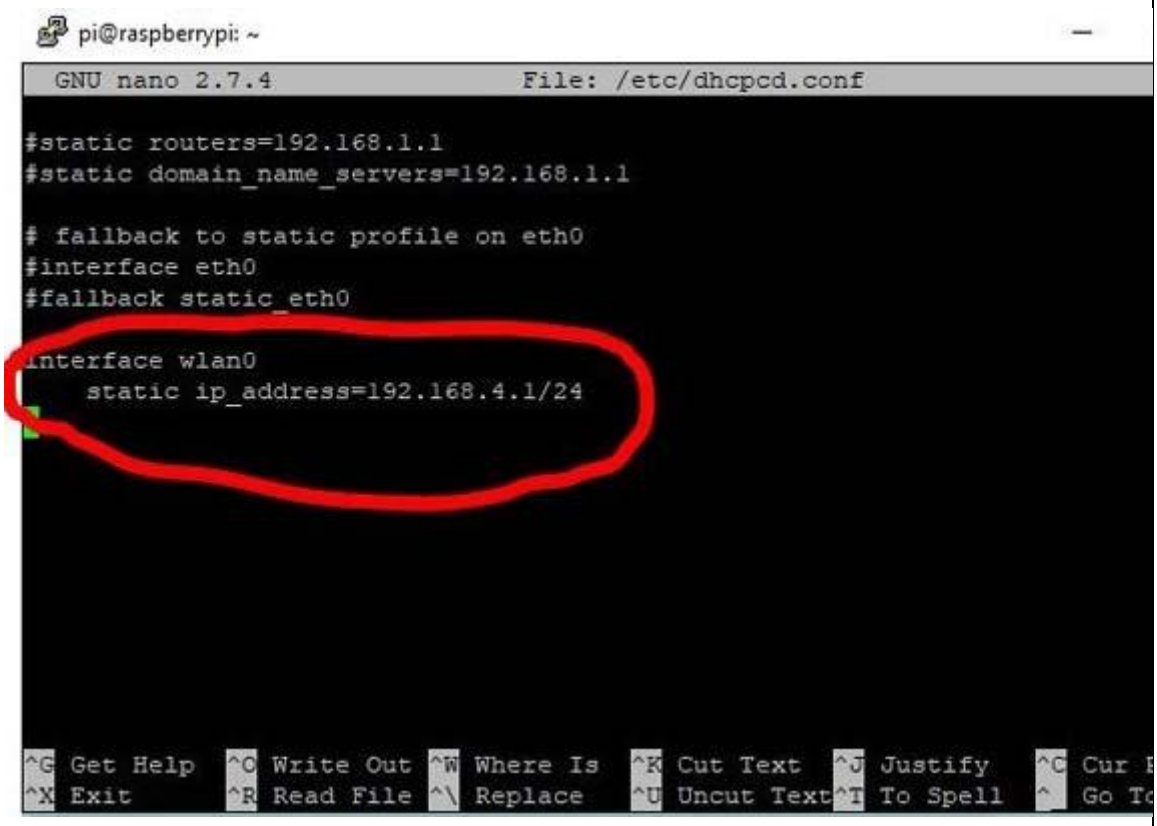
Confirm the *wlan* port on which the wireless device being used is connected. For my Pi, the wireless is on wlan0. **Setting up the Raspberry Pi to act as a server** requires us to assign a static IP address to the wireless port. This can be done by editing the *dhcpcd* config file. To edit the configuration file, run;

```
sudo nano /etc/dhcpcd.conf
```

Scroll to the bottom of the config file and add the following lines.

```
interface wlan0
static ip_address=192.168.1.200/24 #machine ip address
```

After adding the lines, the config file should look like the image below.



Note: This IP address can be changed to suit your preferred configuration.

Save the file and exit using; ctrl+x followed by Y

Restart the *dhcpcd* service to effect the changes made to the configuration using;

```
sudo service dhcpcd restart
```

Step 5: Configure the *dhcpcd* server

With a static IP address now configured for the Raspberry Pi wlan, the next thing is for us to configure the *dhcpcd* server and provide it with the **range of IP addresses to be assigned to devices that connect to the wireless access point**. To do this, we need to edit the configuration file of the *dnsmasq* software but the config file of the software contains way too much info and a lot could go wrong If not properly edited, so instead of editing, we will be creating a new config file with just the amount of information that is needed to make the wireless access point fully functional.

Before creating the new config file, we keep the old on safe by moving and renaming it.

```
sudo mv /etc/dnsmasq.conf  /etc/dnsmasq.conf.old
```

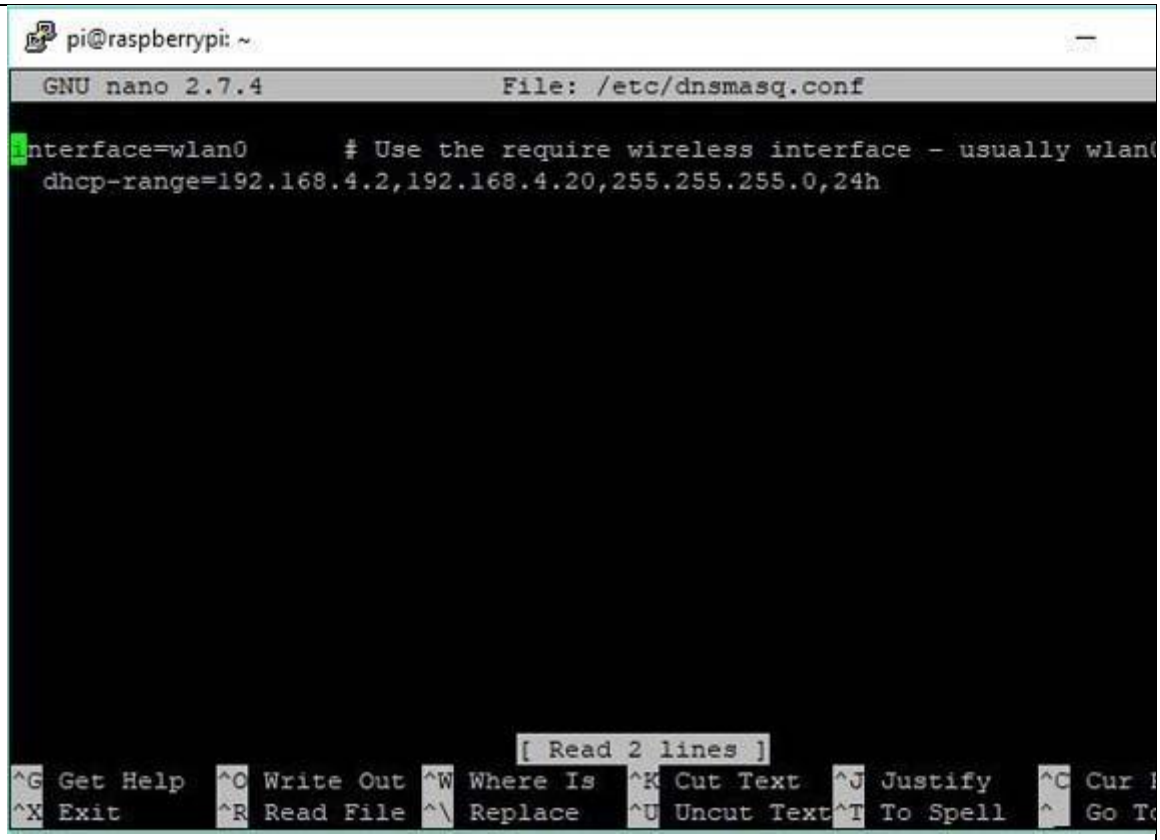
Then launch the editor to create a new configuration file;

```
sudo nano /etc/dnsmasq.conf
```

with the editor launched, copy the lines below and paste in or type directly into it.

```
interface = wlan0  #indicate the communication interface which is usually wlan0 for wire less

dhcp-range = 192.168.1.201, 192.168.1.220, 255.255.255.0,24h #start addr(other than ma chine ip assigned above), end addr, subnet mask, mask
```

the content of the file should look like the image below.



Save the file and exit. The content of this config file is just to specify the range of IP address that can be assigned to devices connected to the wireless access point.

With this done, we will be able to give an identity to devices on our network.

The next set of steps will help us configure the access point host software, setup the ssid, select the encrytpion etc.

Step 6: Configure *hostapd* for SSID and Password

We need to edit the *hostapd* config file(run ***sudo nano /etc/hostapd/hostapd.conf***) to add the various parameters for the wireless network being **setup including the ssid and password**. Its should be noted that the password (passphrase) should be between 8 and 64 characters. Anything lesser won't work.

```
interface=wlan0
driver=nl80211
ssid=piNetwork
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
```

ignore_broadcast_ssid=0

wpa=2

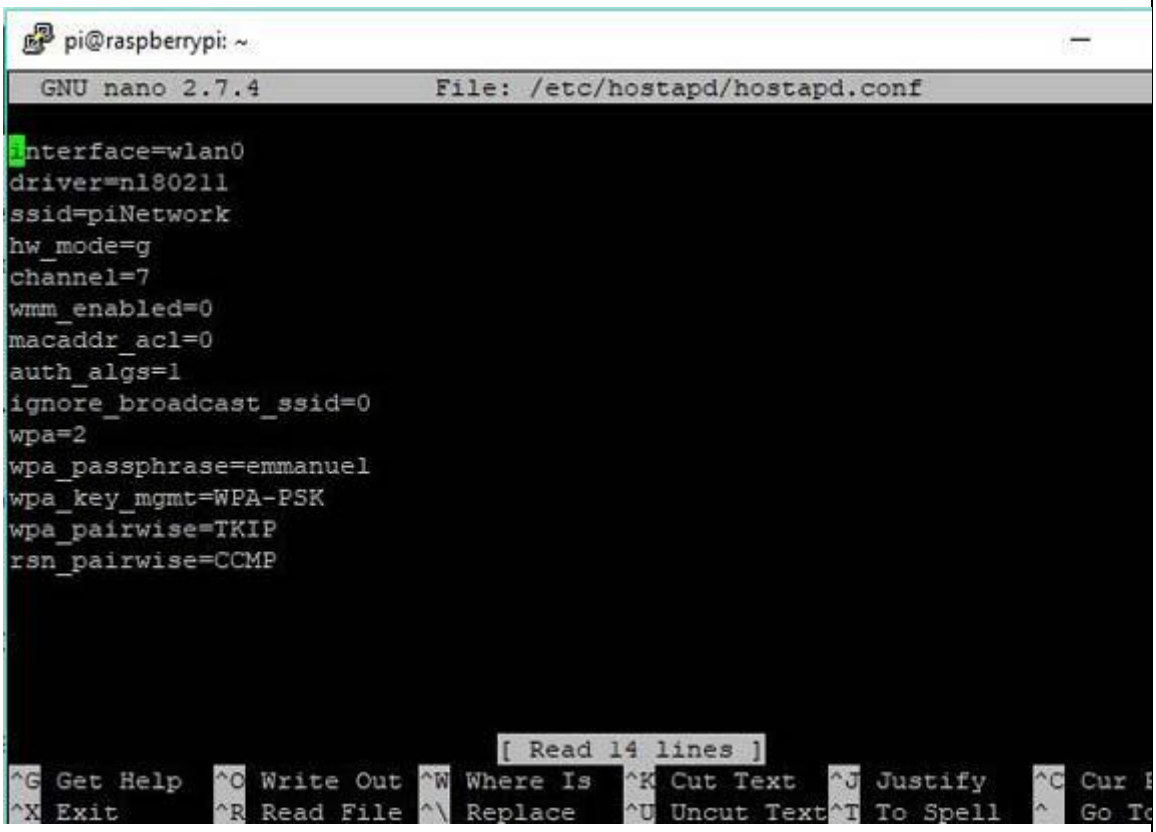
wpa_passphrase=mumbai123 # use a very secure password and not this

wpa_key_mgmt=WPA-PSK

wpa_pairwise=TKIP

rsn_pairwise=CCMP

The content of the file should look like the image below.



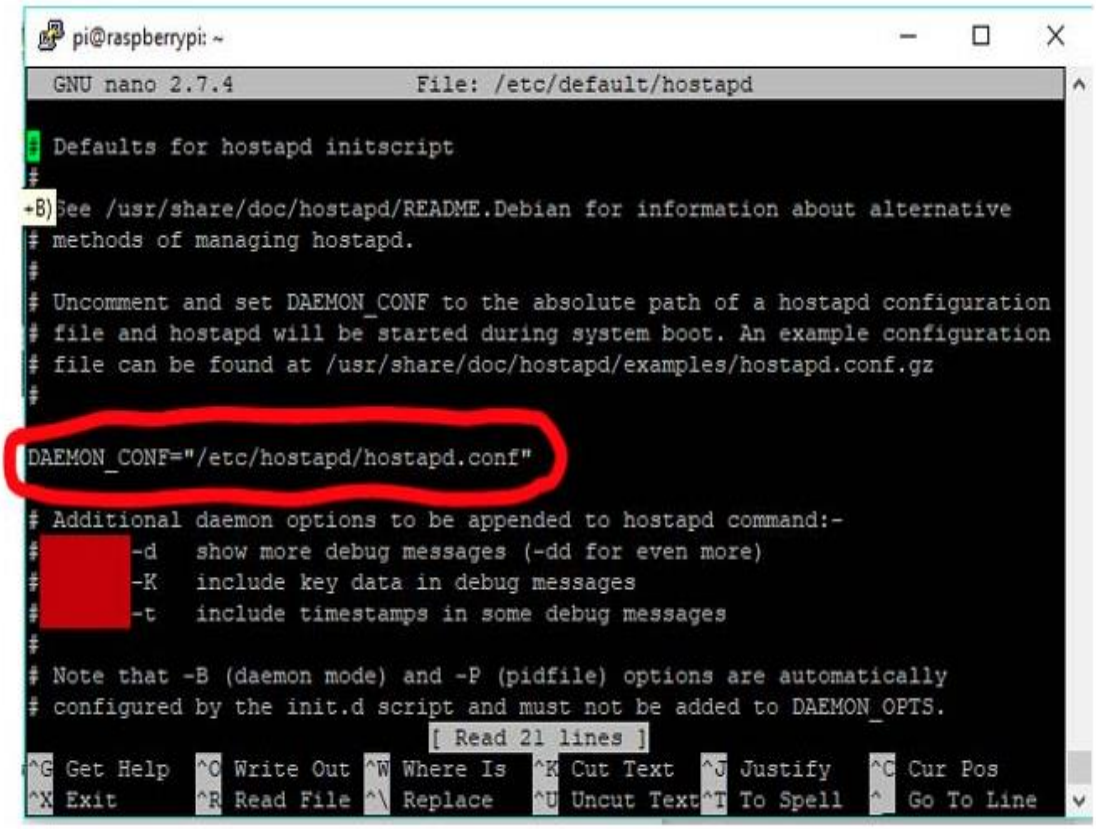
Feel free to change the ssid and password to suit your needs and desire.

Save the config file and exit.

After the config file has been saved, we need to point the hostapd software to where the config file has been saved. To do this, run;

sudo nano /etc/default/hostapd

find the line with *daemon_conf* commented out as shown in the image below.



Uncomment the DAEMON_CONF line and add the line below in between the quotes in front of the “equal to” sign.

/etc/hostapd/hostapd.conf

Step 7: Fire it up

Since we disabled the two software initially, to allow us configure them properly, we need to restart the system after configuration to effect the changes.

Use;

sudo systemctl start hostapd

sudo systemctl start dnsmasq

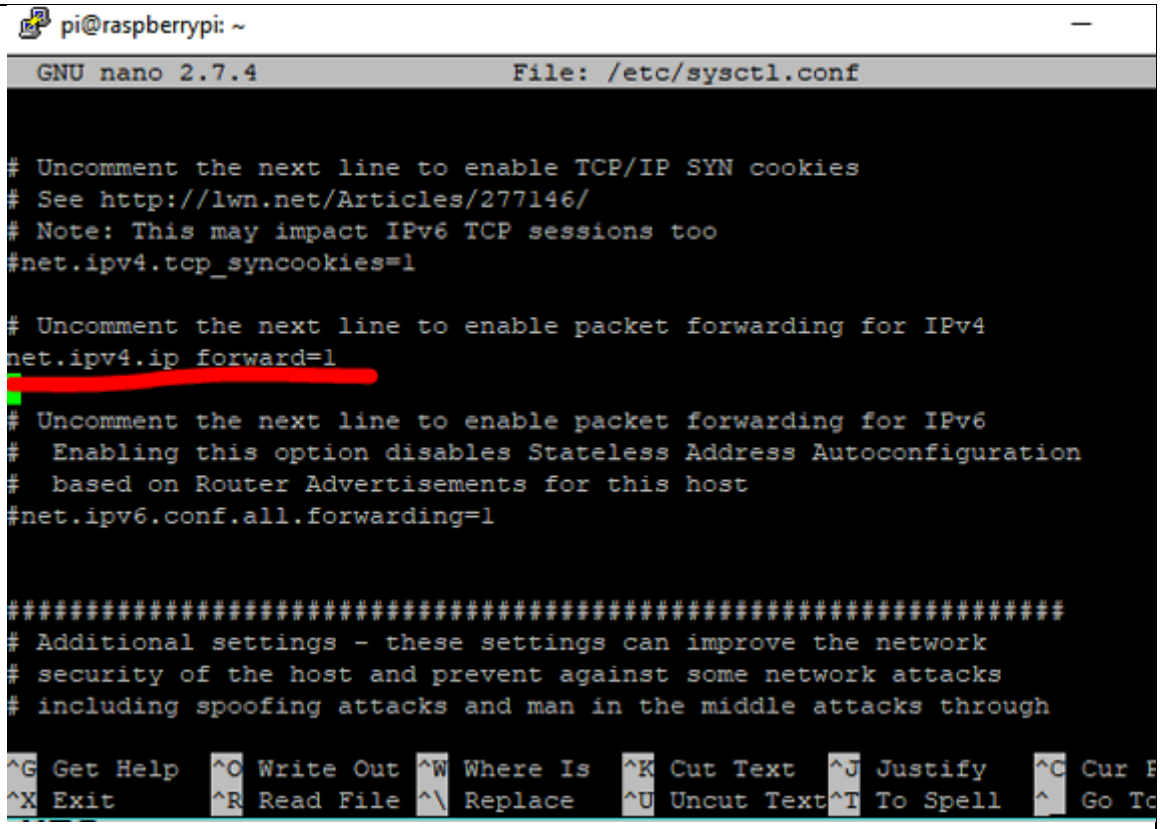
Step 8: Routing and masquerade for outbound traffic

We need to add routing and masquerade for outbound traffic.

To do this, we need to edit the config file of the *systemctl* by running:

sudo nano /etc/sysctl.conf

Uncomment this line ***net.ipv4.ip_forward=1***(highlighted in the image below)



Save the config file and exit using ctrl+x followed by y.

Next we move to masquerading the outbound traffic. This can be done by making some changes to the iptable rule. To do this, run the following commands:

```
sudo iptables -t nat -A  POSTROUTING -o eth0 -j MASQUERADE
```

then save the Iptables rule using:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Step 9: Create Wireless Access Point on startup:

For most wireless access point application, it is often desired that the access point comes up as soon as the system boots. To implement this on the raspberry pi, one of the easiest ways is to add instructions to run the software in the *rc.local* file so we put commands to install the iptable rules on boot in the rc.local file.

To edit the rc.local file, run:

```
sudo nano /etc/rc.local
```

and add the following lines at the bottom of the system, just before the exit 0 statement

```
iptables-restore < /etc/iptables.ipv4.nat
```

Step 9: Reboot! and Use

At this stage, we need to reboot the system to effect all the changes and test the wireless access point starting up on boot with the iptables rule updated.

Reboot the system using:

```
sudo reboot
```

As soon as the system comes back on, you should be able to access the wireless access point using any Wi-Fi enabled device and the password used during the setup.

Accessing the Internet from the Raspberry Pi's Wi-Fi Hotspot

To implement this, we need to put a "bridge" in between the wireless device and the Ethernet device on the Raspberry Pi (the wireless access point) to pass all traffic between the two interfaces. To set this up, we will use the *bridge-utils* software. Install *hostapd* and *bridge-utils*. While we have installed*hostapd* before, run the installation again to clear all doubts.

```
sudo apt-get install hostapd bridge-utils
```

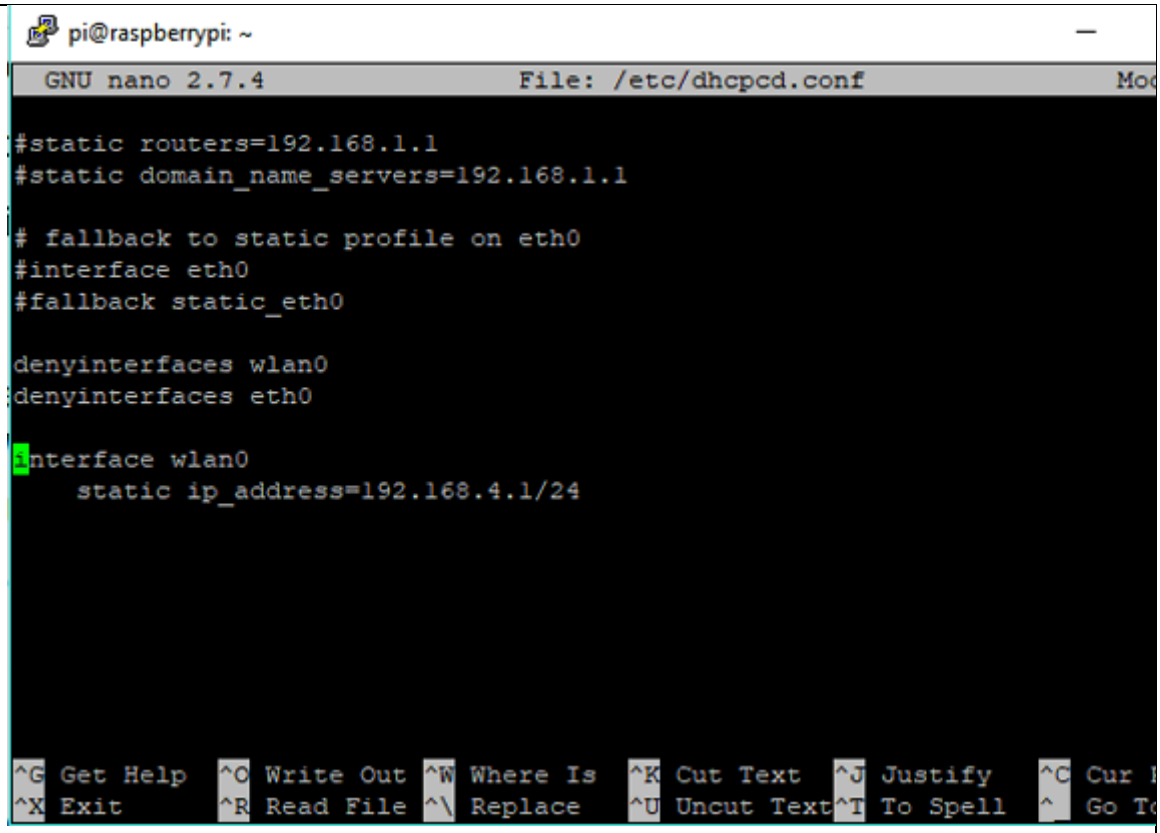
Next, we stop hostapd so as to configure the software.

```
sudo systemctl stop hostapd
```

When a bridge is created, a higher level construct is created over the two ports being bridged and the bridge thus becomes the network device. To prevent conflicts, we need to stop the allocation of IP addresses by the DHCP client running on the Raspberry Pi to the eth0 and wlan0 ports. This will be done by editing the config file of the dhcpcd client to include ***denyinterfaces wlan0*** and ***denyinterfaces eth0*** as shown in the image below.

The file can be edited by running the command;

```
sudo nano /etc/dhcpcd.conf
```



Note: *From this point on, ensure you don't disconnect the Ethernet cable from your PC if you are running in headless mode as you may not be able to connect via SSH again since we have disabled the Ethernet port. If working with a monitor, you have nothing to fear.*

Next, we create a new bridge called br0

sudo brctl addbr br0

Next, we connect the ethernet port (eth0) to the bridge (br0) using;

sudo brctl addif br0 eth0

(Note: if eth0 doesn't exists use ifconfig command to list all Ethernet adapters and use th e name from list)

Next, we edit the interfaces file using ***sudo nano /etc/network/interfaces*** so various devices can work with the bridge. Edit the interfaces file to include the information below;

#Bridge setup

auto br0

iface br0 inet manual

bridge_ports eth0 wlan0

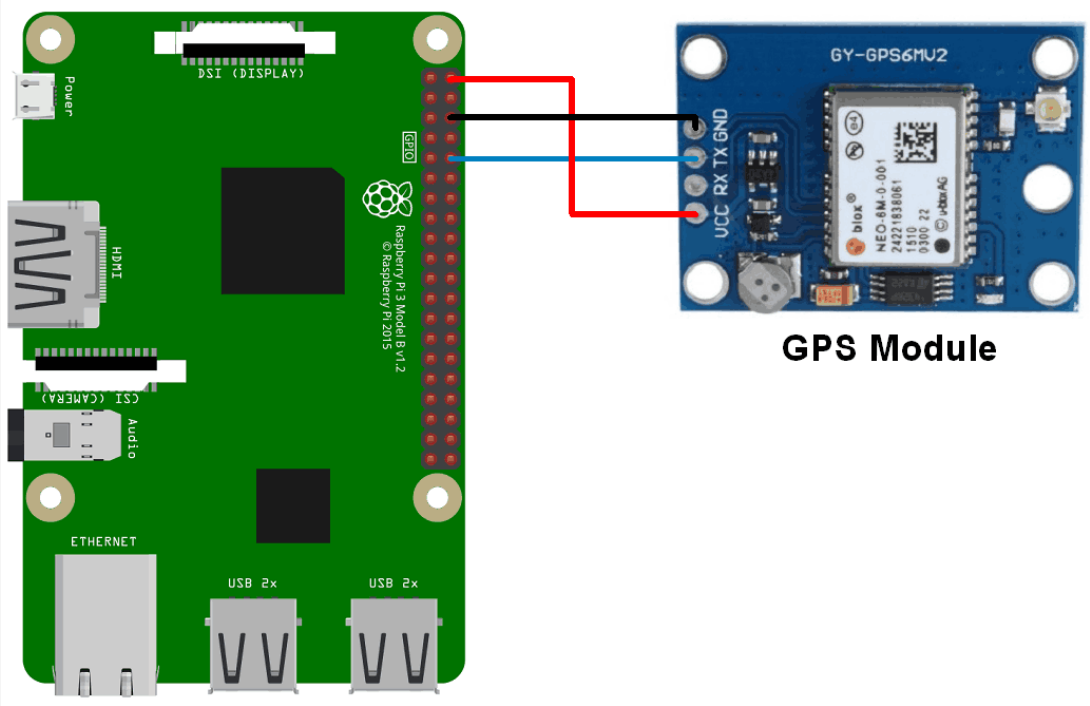
Lastly we edit the hostapd.conf file to include the bridge configuration. This can be done by running the command: ***sudo nano /etc/hostapd/hostapd.conf*** and editing the file to contain the information below. Note the bridge was added below the wlan0 interface and the driver line was commented out.

```
interface=wlan0
bridge=br0
ssid=piNetwork
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=mcctest1
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

With this done, save the config file and exit.

To effect the changes made to the Raspberry Pi, **reboot** the system. Once it comes back up, you should now be **able to access the internet by connecting to the Wireless access point created by the Raspberry Pi**. This of course will only work if internet access is available to the pi via the Ethernet port.

6	**Raspberry Pi GPS Module Interfacing.**



sudo nano /boot/config.txt
################
dtparam=spi=on
dtoverlay=pi3-disable-bt
core_freq=250
enable_uart=1
force_turbo=1
##############
sudo systemctl stop serial-getty@ttyS0.service
sudo systemctl disable serial-getty@ttyS0.service

sudo systemctl enable serial-getty@ttyAMA0.service

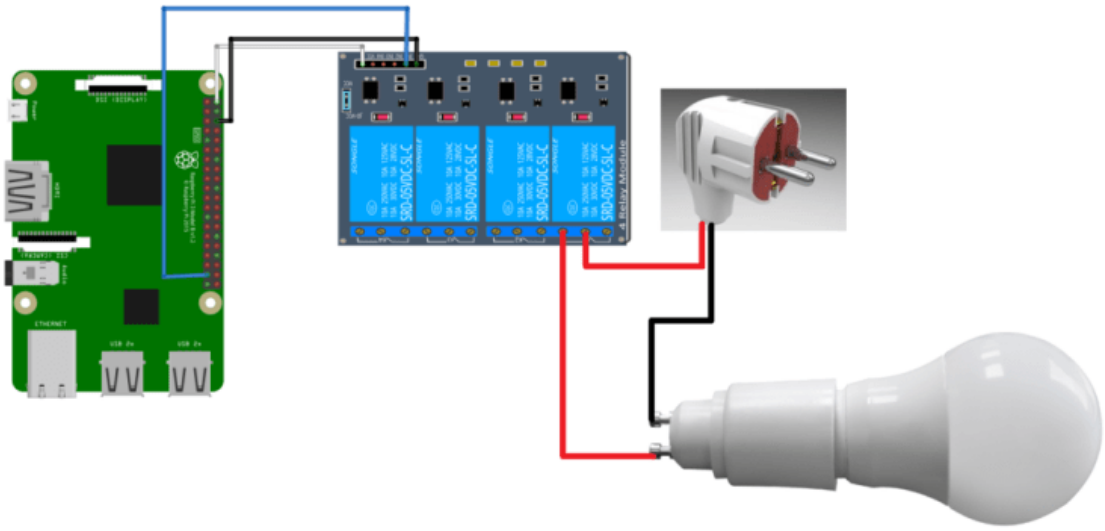
sudo apt-get install minicom
sudo pip install pynmea2

sudo cat /dev/ttyAMA0

code:
import time
import serial
import string
import pynmea2
import RPi.GPIO as gpio
gpio.setmode(gpio.BCM)
port = "/dev/ttyAMA0" # the serial port to which the pi is connected.

```
#create a serial object
ser = serial.Serial(port, baudrate = 9600, timeout = 0.5)
while 1:
   try:
      data = ser.readline()
#         print data
   except:
         print("loading")
#wait for the serial port to churn out data

   if data[0:6] == '$GPGGA':
         msg = pynmea2.parse(data)
         print msg
         time.sleep(2)
```

7	IoT based Web Controlled Home Automation using Raspberry Pi



Code:

```
import RPi.GPIO as GPIO
from time import sleep
relay_pin = 26
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay_pin, GPIO.OUT)
GPIO.output(relay_pin, 1)

try:
      while True:
            GPIO.output(relay_pin, 0)
            sleep(5)
            GPIO.output(relay_pin, 1)
            sleep(5)
except KeyboardInterrupt:
  pass
  GPIO.cleanup()
```

8	Interfacing Raspberry Pi with Pi Camera.

To capture image:

```
import picamera
from time import sleep

#create object for PiCamera class
camera = picamera.PiCamera()
#set resolution
camera.resolution = (1024, 768)
camera.brightness = 60
camera.start_preview()
#add text on image
camera.annotate_text = 'Hi Pi User'
sleep(5)
#store image
camera.capture('image1.jpeg')
camera.stop_preview()
```

To capture video:

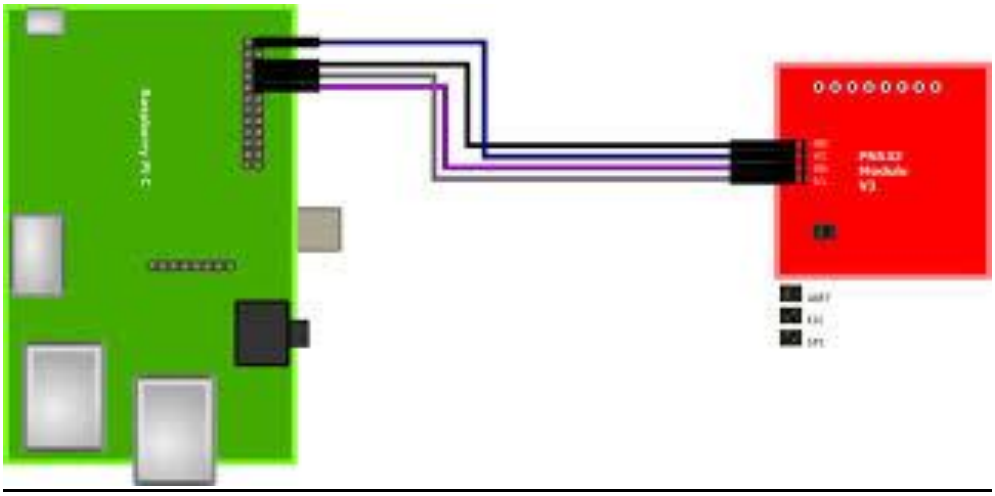
```
import picamera
from time import sleep

camera = picamera.PiCamera()
camera.resolution = (640, 480)
```

```
print()
#start recording using pi camera
camera.start_recording("/home/pi/demo.h264")
#wait for video to record
camera.wait_recording(20)
#stop recording
camera.stop_recording()
camera.close()
print("video recording stopped")
```

To Play the video:

Omxplayer demo.h264

9	Interfacing Raspberry Pi with RFID.



I2C Communication Instructions for Raspberry Pi

1. **Open I2C of the Raspberry Pi :**

sudo raspi-config

Select **5 Interfacing Options** -> **I2C** -> **yes**.

2. **Install some dependent packages**

sudo apt-get update
sudo apt-get install libusb-dev libpcsclite-dev i2c-tools

3. **Download and unzip the source code package of libnfc**

cd ~
wget http://dl.bintray.com/nfc-tools/sources/libnfc-1.7.1.tar.bz2
tar -xf libnfc-1.7.1.tar.bz2

4. **Compile and install**

cd libnfc-1.7.1
./configure --prefix=/usr --sysconfdir=/etc

```
make
sudo make install
```

5. **Write the configuration file for NFC communication**

```
cd /etc
sudo mkdir nfc
sudo nano /etc/nfc/libnfc.conf
```

Check the following details of the file *etc/nfc/libnfc.conf*:

```
# Allow device auto-detection (default: true)
# Note: if this auto-detection is disabled, user has to set manually a device
# configuration using file or environment variable
allow_autoscan = true

# Allow intrusive auto-detection (default: false)
# Warning: intrusive auto-detection can seriously disturb other devices
# This option is not recommended, user should prefer to add manually his device.
allow_intrusive_scan = false

# Set log level (default: error)
# Valid log levels are (in order of verbosity): 0 (none), 1 (error), 2 (info), 3 (debug)
# Note: if you compiled with --enable-debug option, the default log level is "debug"
log_level = 1

# Manually set default device (no default)
# To set a default device, you must set both name and connstring for your device
# Note: if autoscan is enabled, default device will be the first device available in
device list.
#device.name = "_PN532_SPI"
#device.connstring = "pn532_spi:/dev/spidev0.0:500000"
device.name = "_PN532_I2c"
device.connstring = "pn532_i2c:/dev/i2c-1"
```

6. **Wiring**

Toggle the switch to the **I2C mode**

SEL 0	SEL 1

H	L

Connect the devices:

PN532	Raspberry
5V	5V 4
GND	GND 6
SDA	SDA0 3
SCL	SCL0 5

7. Run *i2cdetect –yes 1* to check whether the I2C device is recognized.

If yes, it means both the module and the wiring work well.
Then type in *nfc-list* to check the NFC module:

```
pi@raspberrypi: ~
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- 24 -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $ nfc-list
nfc-list uses libnfc 1.7.1
NFC device: pn532_i2c:/dev/i2c-1 opened
pi@raspberrypi:~ $
```

Run *nfc-poll* to scan the RFID tag and you can read information on the card:

```
pi@raspberrypi:~ $ nfc-list
nfc-list uses libnfc 1.7.1
NFC device: pn532_i2c:/dev/i2c-1 opened
pi@raspberrypi:~ $ nfc-poll
nfc-poll uses libnfc 1.7.1
NFC reader: pn532_i2c:/dev/i2c-1 opened
NFC device will poll during 30000 ms (20 pollings of 300 ms for 5 modulations
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
       UID (NFCID1): f4  55  4e  b8
      SAK (SEL_RES): 08
nfc_initiator_target_is_present: Target Released
Waiting for card removing...done.
pi@raspberrypi:~ $
```

SPI Communication Instructions for Raspberry Pi

1. Open SPI of the Raspberry Pi:

```
sudo raspi-config
```

Select **9 Advanced Options** -> **SPI** -> **yes**.

2. Install some dependent packages

```
sudo apt-get update
sudo apt-get install libusb-dev libpcsclite-dev i2c-tools
```

3. Download and unzip the source code package of libnfc

```
cd ~
wget http://dl.bintray.com/nfc-tools/sources/libnfc-1.7.1.tar.bz2
tar -xf libnfc-1.7.1.tar.bz2
```

4. Compile and install

```
cd libnfc-1.7.1
./configure --prefix=/usr --sysconfdir=/etc
make
sudo make install
```

5. Write the configuration file for NFC communication

```
cd /etc
```

```
sudo mkdir nfc
sudo nano /etc/nfc/libnfc.conf
```

Check the following details of the file *etc/nfc/libnfc.conf*:

```
# Allow device auto-detection (default: true)
# Note: if this auto-detection is disabled, user has to set manually a device
# configuration using file or environment variable
allow_autoscan = true

# Allow intrusive auto-detection (default: false)
# Warning: intrusive auto-detection can seriously disturb other devices
# This option is not recommended, user should prefer to add manually his device.
allow_intrusive_scan = false
```

```
# Set log level (default: error)
# Valid log levels are (in order of verbosity): 0 (none), 1 (error), 2 (info), 3 (debug)
# Note: if you compiled with --enable-debug option, the default log level is "debug"
log_level = 1

# Manually set default device (no default)
# To set a default device, you must set both name and connstring for your device
# Note: if autoscan is enabled, default device will be the first device available in
device list.
device.name = "_PN532_SPI"
device.connstring = "pn532_spi:/dev/spidev0.0:500000"
#device.name = "_PN532_I2c"
#device.connstring = "pn532_i2c:/dev/i2c-1"
```

6. Wiring

Toggle the switch to the **SPI mode**

SEL0	SEL1
L	H

Connect the devices:

PN532	Raspberry
5V	5V
GND	GND
SCK	SCKL
MISO	MISO
MOSI	MOSI
NSS	CE0

7. Run *ls /dev/spidev0. to check whether the SPI is opened or not.**

If yes, it means both the module and the wiring work well.

Then type in *nfc-list* to check the NFC module:

/dev/spidev0.0 /dev/spidev0.1

If two devices are detected, it means the SPI is already opened.

Then type in *nfc-list* to check the NFC module:

```
pi@raspberrypi:~ $ nfc-list
nfc-list uses libnfc 1.7.1
NFC device: pn532_spi:/dev/spidev0.0 opened
pi@raspberrypi:~ $
```

For Raspberry Pi 3, you may be appear the following error

```
pi@raspberrypi:/etc $ nfc-list
nfc-list uses libnfc 1.7.1
error   libnfc.driver.pn532_spi Unable to wait for SPI data. (RX)
pn53x_check_communication: Timeout
error   libnfc.driver.pn532_spi Unable to wait for SPI data. (RX)
nfc-list: ERROR: Unable to open NFC device: pn532_spi:/dev/spidev0.0:500000
pi@raspberrypi:/etc $
```

You should modifiy the *libnfc.conf*

sudo nano /etc/nfc/libnfc.conf

then modify 500000 to 50000:

device.connstring = "pn532_spi:/dev/spidev0.0:50000"

Run *nfc-poll* to scan the RFID tag and you can read information on the card:

```
pi@raspberrypi:~ $ nfc-poll
nfc-poll uses libnfc 1.7.1
NFC reader: pn532_spi:/dev/spidev0.0 opened
NFC device will poll during 30000 ms (20 pollings of 300 ms for 5 modulations)
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
       UID (NFCID1): f4  55  4e  b8
      SAK (SEL_RES): 08
nfc_initiator_target_is_present: Target Released
Waiting for card removing...done.
pi@raspberrypi:~ $
```

Code:

```
import subprocess
import time

def nfc_raw():
        lines=subprocess.check_output("/usr/bin/nfc-poll",
stderr=open('/dev/null','w'))
        return lines

def read_nfc():
        lines=nfc_raw()
        return lines

try:
        while True:
                myLines=read_nfc()
                buffer=[]
                for line in myLines.splitlines():
                        line_content=line.split()
```

```
                    if(not line_content[0] =='UID'):
                        pass
                    else:
                        buffer.append(line_content)
            str=buffer[0]
            id_str=str[2]+str[3]+str[4]+str[5]
            print (id_str)

except KeyboardInterrupt:
    pass
```

10	**Installing Windows 10 IoT Core on Raspberry Pi.**

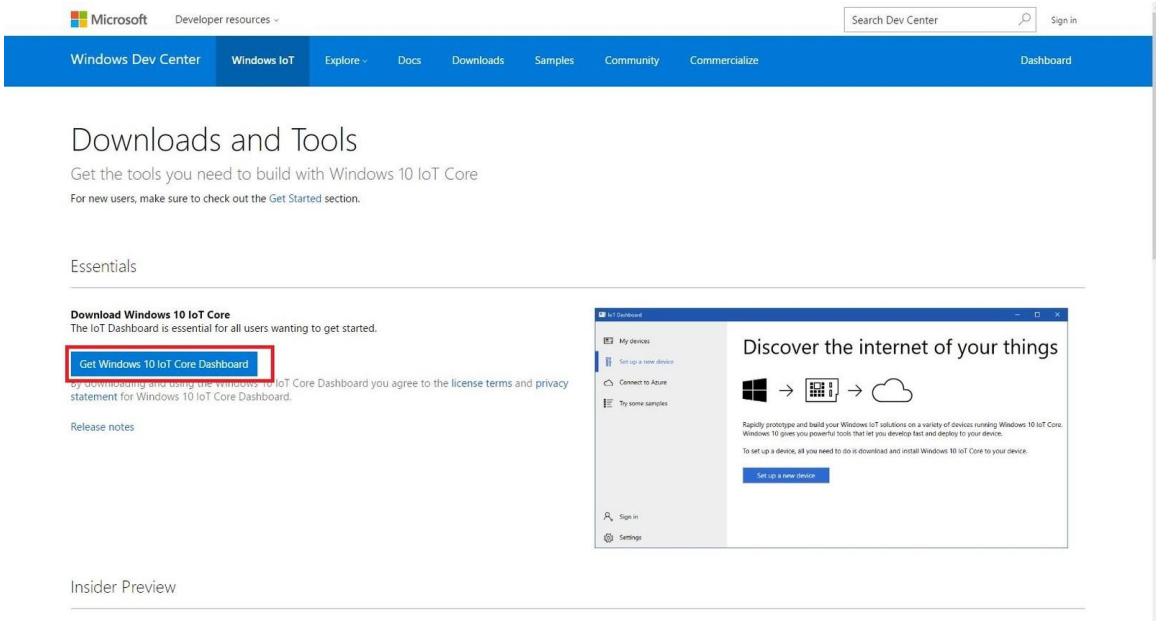
To get up and running you need a few bits and pieces:

1. Raspberry Pi 3.
2. 5V 2A microUSB power supply.
3. 8GB or larger Class 10 microSD card with full-size SD adapter.
4. HDMI cable.
5. Access to a PC.
6. USB WiFi adapter (older models of Raspberry Pi) or Ethernet cable.

At this point, the HDMI cable is only to plug the Raspberry Pi into a display so you can make sure your install worked. Some Raspberry Pi starter kits include everything you need, but the list above covers the power, display, and something to install Windows 10 IoT Core on.

Go to the Windows 10 developer center.

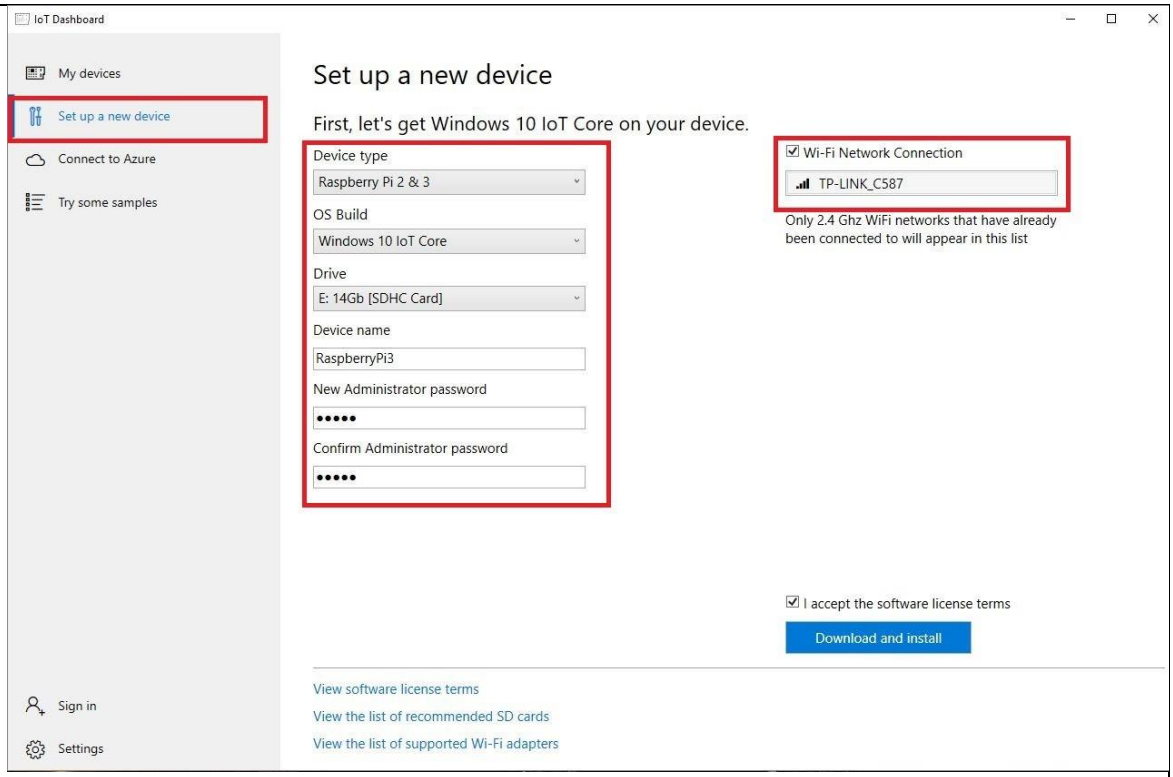
Click **Get Windows 10 IoT Core Dashboard** to download the necessary application.



Install the application and open it.

Select **set up a new device** from the sidebar.

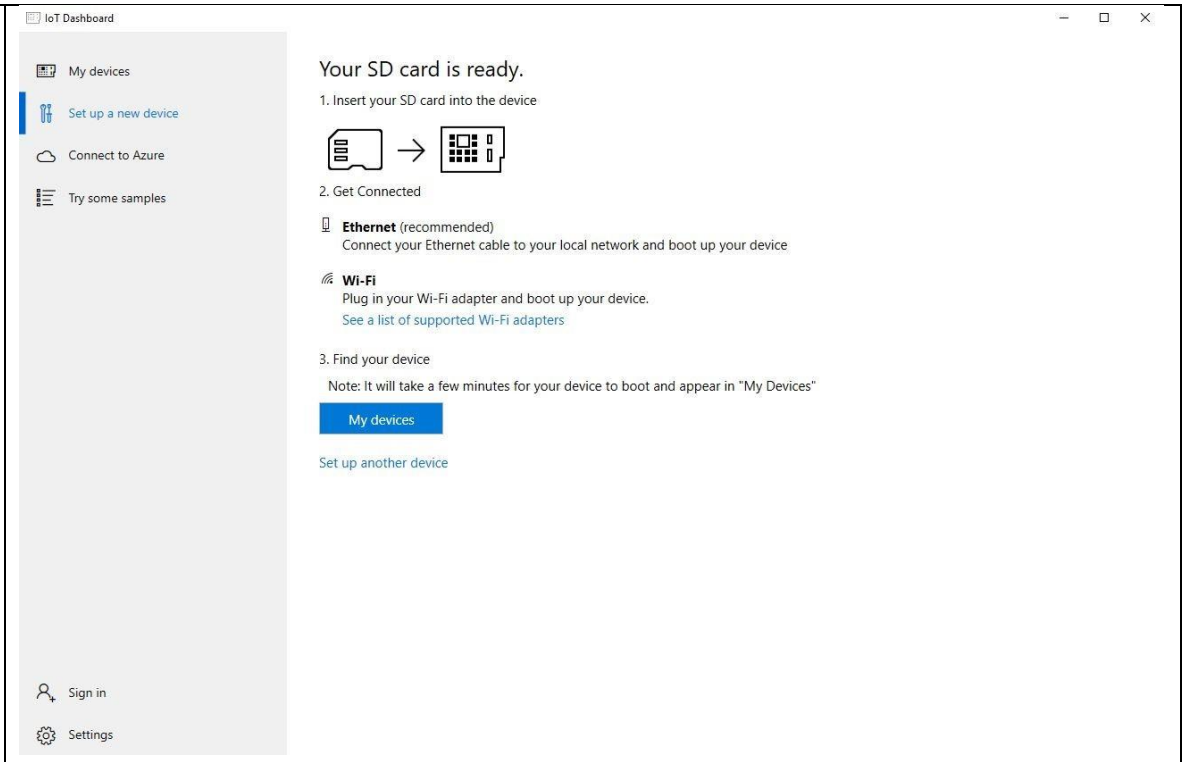
Select the options as shown in the image below. Make sure you select the correct drive for your microSD card and give your device a name and admin password.



Select the WiFi network connection you want your Raspberry Pi to connect to, if required. Only networks your PC connects to will be shown.

Click **download and install**.

The application will now download the necessary files from Microsoft and flash them to your microSD card. It'll take a little while, but the dashboard will show you the progress.



Once the image has been installed on the microSD card, it's time to eject it from your PC and go over to the Raspberry Pi. First connect up the micro USB cable and power supply, HDMI cable and USB WiFi adapter or Ethernet cable. Connect the HDMI cable to your chosen display, insert the microSD card into the Raspberry Pi and power it up.

Deep learning

Practical No:1

Aim: Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow.

```
import tensorflow as tf print("Matrix
Multiplication Demo")
x=tf.constant([1,2,3,4,5,6],shape=[2,
3]) print(x)
y=tf.constant([7,8,9,10,11,12],shape=[3,
2]) print(y) z=tf.matmul(x,y)
print("Product:",z)
e_matrix_A=tf.random.uniform([2,2],minval=3,maxval=10,dtype=tf.float32,name="matrixA")
print("Matrix A:\n{}\n\n".format(e_matrix_A))
eigen_values_A,eigen_vectors_A=tf.linalg.eigh(e_matrix_A)
print("Eigen Vectors:\n{}\n\nEigen Values:\n{}\n".format(eigen_vectors_A,eigen_values_A))
```

OUTPUT:

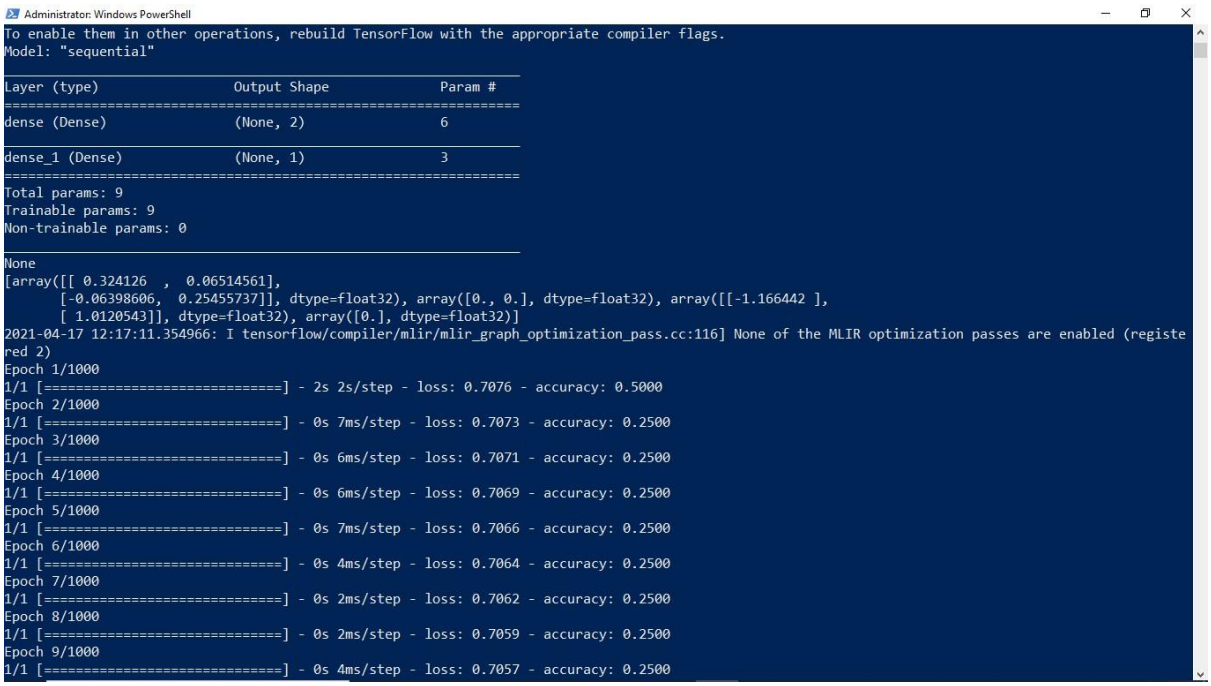
```
tf.Tensor(
[[1 2 3]
 [4 5 6]], shape=(2, 3), dtype=int32)
tf.Tensor(
[[ 7  8]
 [ 9 10]
 [11 12]], shape=(3, 2), dtype=int32)
Product: tf.Tensor(
[[ 58  64]
 [139 154]], shape=(2, 2), dtype=int32)
Matrix A:
[[7.791751  6.3527837]
 [6.8659496 5.229142 ]]


Eigen Vectors:
[[-0.63896394  0.7692366 ]
 [ 0.7692366   0.63896394]]

Eigen Values:
[-0.47403672 13.494929  ]

(venv) PS D:\keras>
```

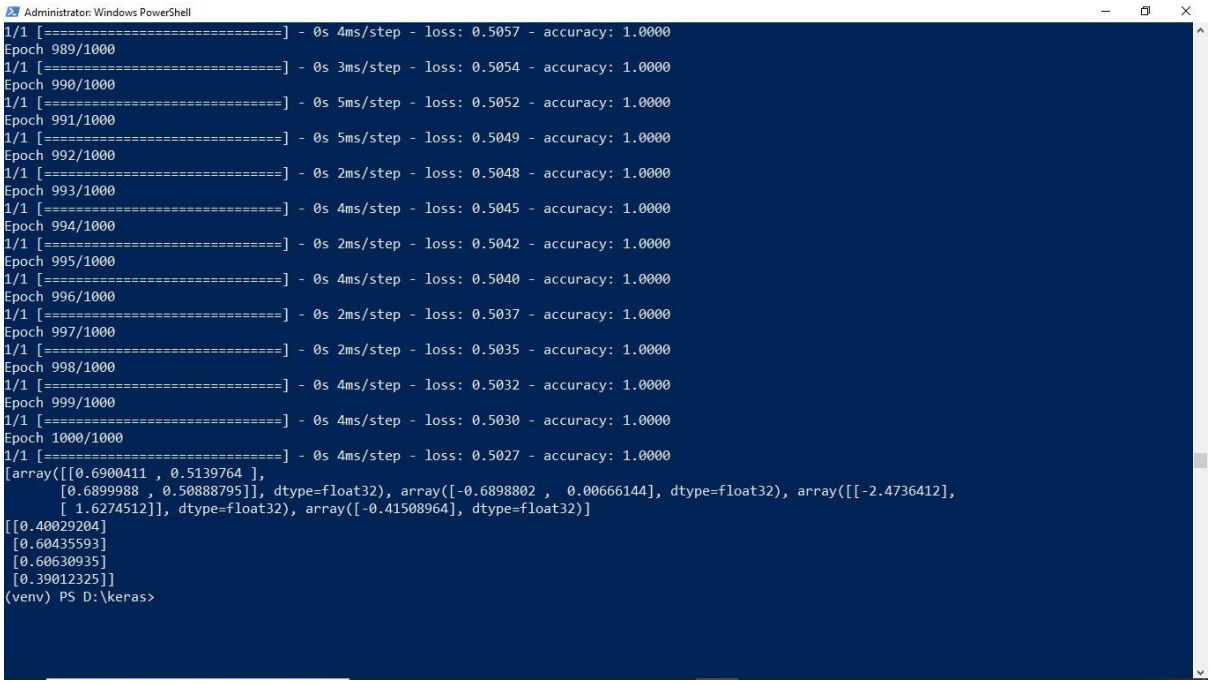
Practical No:2

Aim: Solving XOR problem using deep feed forward network.

```
import numpy as np from

keras.layers import Dense from

keras.models import Sequential

model=Sequential()

model.add(Dense(units=2,activation='relu',input_dim=2))

model.add(Dense(units=1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac

y']) print(model.summary()) print(model.get_weights())

X=np.array([[0.,0.],[0.,1.],[1.,0.],[1.,1.]])

Y=np.array([0.,1.,1.,0.])

model.fit(X,Y,epochs=1000,batch_size=4)

print(model.get_weights())

print(model.predict(X,batch_size=4))
```

OUTPUT:



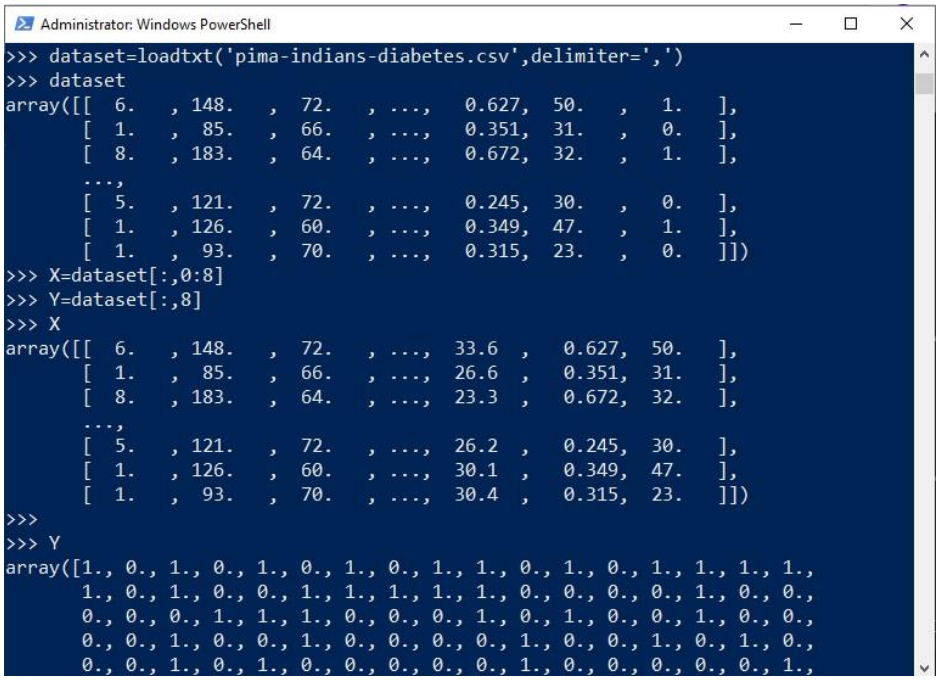

Administrator: Windows PowerShell
1/1 [==============================] - 0s 4ms/step - loss: 0.5057 - accuracy: 1.0000
Epoch 989/1000
1/1 [==============================] - 0s 3ms/step - loss: 0.5054 - accuracy: 1.0000
Epoch 990/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.5052 - accuracy: 1.0000
Epoch 991/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.5049 - accuracy: 1.0000
Epoch 992/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.5048 - accuracy: 1.0000
Epoch 993/1000
1/1 [==============================] - 0s 4ms/step - loss: 0.5045 - accuracy: 1.0000
Epoch 994/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.5042 - accuracy: 1.0000
Epoch 995/1000
1/1 [==============================] - 0s 4ms/step - loss: 0.5040 - accuracy: 1.0000
Epoch 996/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.5037 - accuracy: 1.0000
Epoch 997/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.5035 - accuracy: 1.0000
Epoch 998/1000
1/1 [==============================] - 0s 4ms/step - loss: 0.5032 - accuracy: 1.0000
Epoch 999/1000
1/1 [==============================] - 0s 4ms/step - loss: 0.5030 - accuracy: 1.0000
Epoch 1000/1000
1/1 [==============================] - 0s 4ms/step - loss: 0.5027 - accuracy: 1.0000
[array([[0.6900411 , 0.5139764],
[0.6899988 , 0.50888795]], dtype=float32), array([-0.6898802 , 0.00666144], dtype=float32), array([[-2.4736412],
[1.6274512]], dtype=float32), array([-0.41508964], dtype=float32)]
[[0.40029204]
[0.60435593]
[0.60630935]
[0.39012325]]
(venv) PS D:\keras>

Practical No:3

Aim: Implementing deep neural network for performing classification task.

Problem statement: the given dataset comprises of health information about diabetic women patient. we need to create deep feed forward network that will classify women suffering from diabetes mellitus as 1.

```
>>> from numpy import loadtxt
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>>
```



Creating model:

```
>>> model=Sequential()
```

```
>>> model.add(Dense(12,input_dim=8,activation='relu'))
>>> model.add(Dense(8,activation='relu'))
>>> model.add(Dense(1,activation='sigmoid'))
>>>
```

Compiling and fitting model:

```
>>> model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
>>> model.fit(X,Y,epochs=150,batch_size=10)
```

```
Administrator: Windows PowerShell
>>> model.add(Dense(1,activation='sigmoid'))
>>> model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
>>> model.fit(X,Y,epochs=150,batch_size=10)
2021-04-05 17:40:32.289557: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registe
red 2)
Epoch 1/150
77/77 [==============================] - 2s 2ms/step - loss: 2.6770 - accuracy: 0.4399
Epoch 2/150
77/77 [==============================] - 0s 1ms/step - loss: 1.1332 - accuracy: 0.5064
Epoch 3/150
77/77 [==============================] - 0s 2ms/step - loss: 0.8624 - accuracy: 0.5592
Epoch 4/150
77/77 [==============================] - 0s 2ms/step - loss: 0.8135 - accuracy: 0.5700
Epoch 5/150
77/77 [==============================] - 0s 2ms/step - loss: 0.7369 - accuracy: 0.6089
Epoch 6/150
77/77 [==============================] - 0s 1ms/step - loss: 0.7405 - accuracy: 0.6269
Epoch 7/150
77/77 [==============================] - 0s 2ms/step - loss: 0.7157 - accuracy: 0.6060
Epoch 8/150
77/77 [==============================] - 0s 1ms/step - loss: 0.6852 - accuracy: 0.6354
Epoch 9/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6585 - accuracy: 0.6398
Epoch 10/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6524 - accuracy: 0.6330
Epoch 11/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6671 - accuracy: 0.6584
Epoch 12/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6216 - accuracy: 0.6857
Epoch 13/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6656 - accuracy: 0.6469
Epoch 14/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6304 - accuracy: 0.6870
Epoch 15/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6290 - accuracy: 0.6594
Epoch 16/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6033 - accuracy: 0.6722
```

Evaluating the accuracy:

```
>>> _,accuracy=model.evaluate(X,Y)
24/24 [==============================] - 0s 1ms/step - loss: 0.4849 - accuracy: 0
.7591
>>> print('Accuracy of model is',(accuracy*100))
Accuracy of model is 75.91145634651184
>>>
```

Using model for prediction class:

```
>>> prediction=model.predict_classes(X)
```

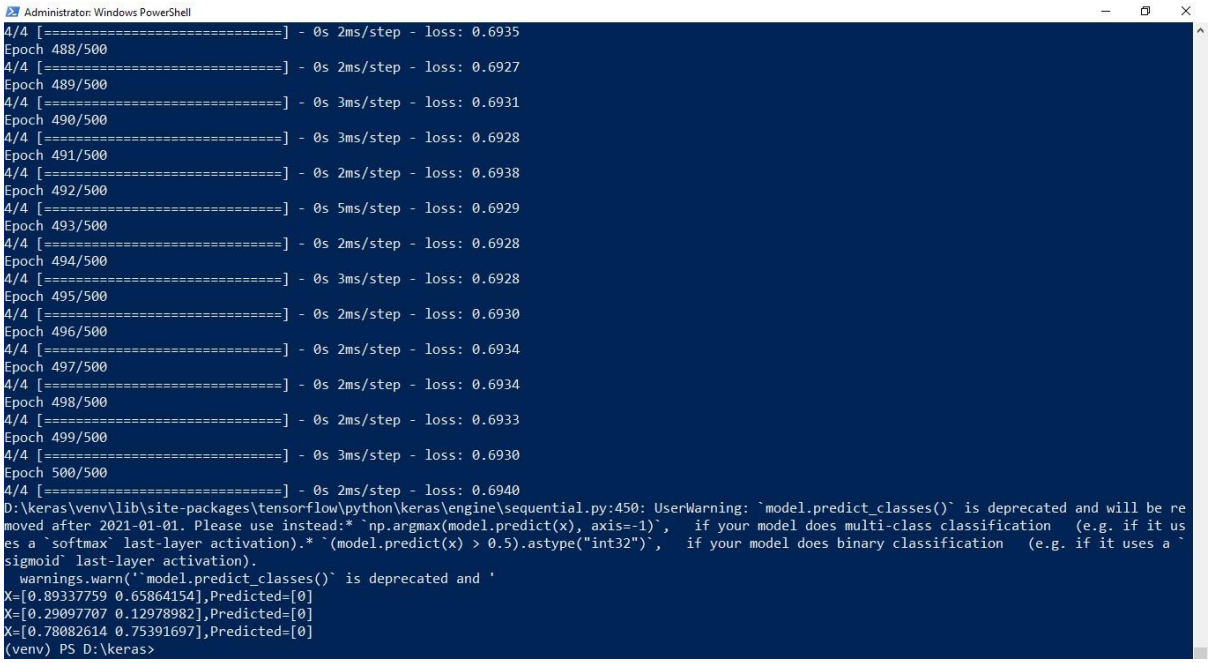
```
>>> exec("for i in range(5):print(X[i].tolist(),prediction[i],Y[i])")
[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0] [1] 1.0
[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0] [0] 0.0
[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0] [1] 1.0
[1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0] [0] 0.0
[0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0] [1] 1.0
>>>
```

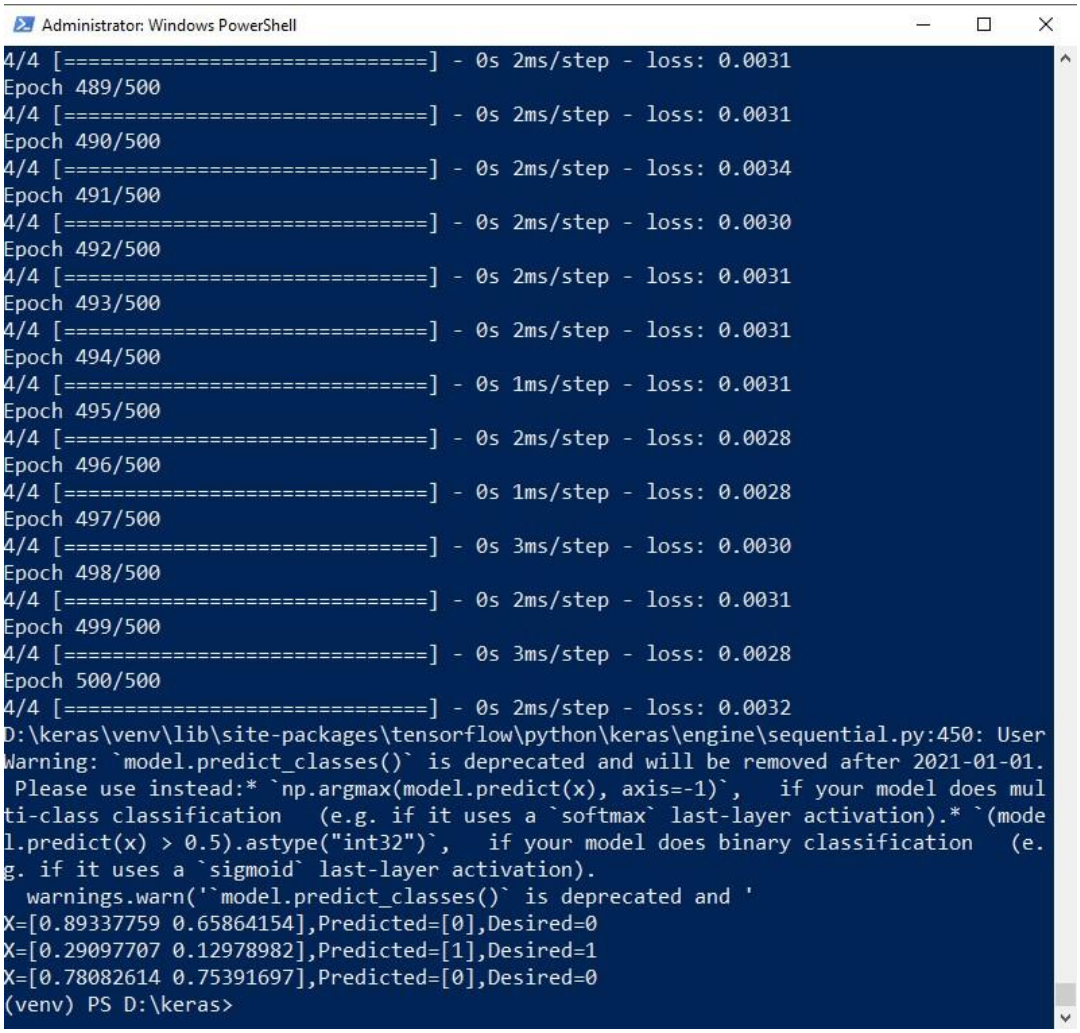
Practical No:4

d) Aim: Using deep feed forward network with two hidden layers for performing classification and predicting the class.

```
From keras.models import Sequential from
keras.layers import Dense from
sklearn.datasets import make_blobs from
sklearn.preprocessing import MinMaxScaler

X,Y=make_blobs(n_samples=100,centers=2,n_features=2,random_state
=1) scalar=MinMaxScaler() scalar.fit(X)
X=scalar.transform(X)

model=Sequential()
model.add(Dense(4,input_dim=2,activation='relu'))
model.add(Dense(4,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam')
model.fit(X,Y,epochs=500)

Xnew,Yreal=make_blobs(n_samples=3,centers=2,n_features=2,random_state=1)
Xnew=scalar.transform(Xnew)

Ynew=model.predict_classes(Xnew)
for I in range(len(Xnew)):
    print("X=%s,Predicted=%s,Desired=%s"%(Xnew[i],Ynew[i],Yreal[i]))
```

OUTPUT:

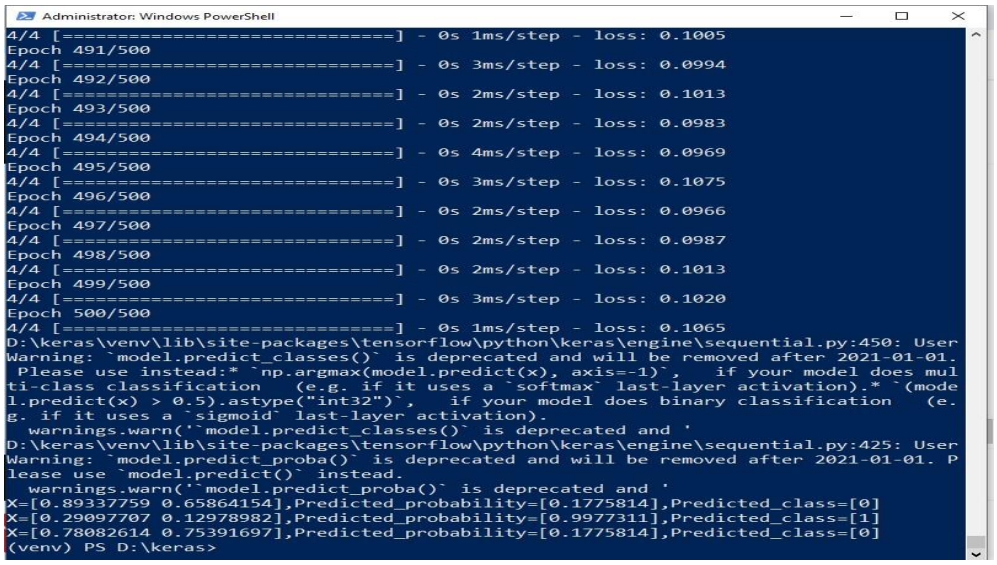

Administrator: Windows PowerShell
4/4 [==========] - 0s 2ms/step - loss: 0.6935
Epoch 488/500
4/4 [==========] - 0s 2ms/step - loss: 0.6927
Epoch 489/500
4/4 [==========] - 0s 3ms/step - loss: 0.6931
Epoch 490/500
4/4 [==========] - 0s 3ms/step - loss: 0.6928
Epoch 491/500
4/4 [==========] - 0s 2ms/step - loss: 0.6938
Epoch 492/500
4/4 [==========] - 0s 5ms/step - loss: 0.6929
Epoch 493/500
4/4 [==========] - 0s 2ms/step - loss: 0.6928
Epoch 494/500
4/4 [==========] - 0s 3ms/step - loss: 0.6928
Epoch 495/500
4/4 [==========] - 0s 2ms/step - loss: 0.6930
Epoch 496/500
4/4 [==========] - 0s 2ms/step - loss: 0.6934
Epoch 497/500
4/4 [==========] - 0s 2ms/step - loss: 0.6934
Epoch 498/500
4/4 [==========] - 0s 2ms/step - loss: 0.6933
Epoch 499/500
4/4 [==========] - 0s 3ms/step - loss: 0.6930
Epoch 500/500
4/4 [==========] - 0s 2ms/step - loss: 0.6940
D:\keras\venv\lib\site-packages\tensorflow\python\keras\engine\sequential.py:450: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`, if your model does multi-class classification (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`, if your model does binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn('`model.predict_classes()` is deprecated and '
X=[0.89337759 0.65864154],Predicted=[0]
X=[0.29097707 0.12978982],Predicted=[0]
X=[0.78082614 0.75391697],Predicted=[0]
(venv) PS D:\keras>


Administrator: Windows PowerShell
4/4 [==========] - 0s 2ms/step - loss: 0.0031
Epoch 489/500
4/4 [==========] - 0s 2ms/step - loss: 0.0031
Epoch 490/500
4/4 [==========] - 0s 2ms/step - loss: 0.0034
Epoch 491/500
4/4 [==========] - 0s 2ms/step - loss: 0.0030
Epoch 492/500
4/4 [==========] - 0s 2ms/step - loss: 0.0031
Epoch 493/500
4/4 [==========] - 0s 2ms/step - loss: 0.0031
Epoch 494/500
4/4 [==========] - 0s 1ms/step - loss: 0.0031
Epoch 495/500
4/4 [==========] - 0s 2ms/step - loss: 0.0028
Epoch 496/500
4/4 [==========] - 0s 1ms/step - loss: 0.0028
Epoch 497/500
4/4 [==========] - 0s 3ms/step - loss: 0.0030
Epoch 498/500
4/4 [==========] - 0s 2ms/step - loss: 0.0031
Epoch 499/500
4/4 [==========] - 0s 3ms/step - loss: 0.0028
Epoch 500/500
4/4 [==========] - 0s 2ms/step - loss: 0.0032
D:\keras\venv\lib\site-packages\tensorflow\python\keras\engine\sequential.py:450: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`, if your model does multi-class classification (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`, if your model does binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn('`model.predict_classes()` is deprecated and '
X=[0.89337759 0.65864154],Predicted=[0],Desired=0
X=[0.29097707 0.12978982],Predicted=[1],Desired=1
X=[0.78082614 0.75391697],Predicted=[0],Desired=0
(venv) PS D:\keras>

b) Aim: Using a deep field forward network with two hidden layers for performing classification and predicting the probability of class.

```
From keras.models import Sequential from

keras.layers import Dense from

sklearn.datasets import make_blobs from

sklearn.preprocessing import MinMaxScaler

X,Y=make_blobs(n_samples=100,centers=2,n_features=2,random_state

=1) scalar=MinMaxScaler() scalar.fit(X)

X=scalar.transform(X)

model=Sequential()

model.add(Dense(4,input_dim=2,activation='relu'))

model.add(Dense(4,activation='relu'))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam')

model.fit(X,Y,epochs=500)

Xnew,Yreal=make_blobs(n_samples=3,centers=2,n_features=2,random_state=1)

Xnew=scalar.transform(Xnew)

Yclass=model.predict_classes(Xne

w)

Ynew=model.predict_proba(Xnew)

for I in range(len(Xnew)):

    print("X=%s,Predicted_probability=%s,Predicted_class=%s"%(Xnew[i],Ynew[i],Yclass[i]))
```

OUTPUT:



e) Aim: Using a deep field forward network with two hidden layers for performing linear regression and predicting values.

```
From keras.models import Sequential from

keras.layers import Dense from

sklearn.datasets import make_regression from

sklearn.preprocessing import MinMaxScaler


X,Y=make_regression(n_samples=100,n_features=2,noise=0.1,random_state=1)

calar,scalarY=MinMaxScaler(),MinMaxScaler()


calar.fit(X)

scalarY.fit(Y.reshape(100,1))

X=calar.transform(X)

Y=scalarY.transform(Y.reshape(100,1))


model=Sequential()

model.add(Dense(4,input_dim=2,activation='relu'))

model.add(Dense(4,activation='relu'))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='mse',optimizer='adam')

model.fit(X,Y,epochs=1000,verbose=0)


Xnew,a=make_regression(n_samples=3,n_features=2,noise=0.1,random_state=1)

Xnew=calar.transform(Xnew)


Ynew=model.predict(Xnew)


for I in range(len(Xnew)):

    print("X=%s,Predicted=%s"%(Xnew[i],Ynew[i]))
```

OUTPUT:

```
X=[0.29466096 0.30317302],Predicted=[0.18255734]
X=[0.39445118 0.79390858],Predicted=[0.7581165]
X=[0.02884127 0.6208843 ],Predicted=[0.3932857]
(venv) PS D:\keras>
```

Practical No:5(a)

Aim: Evaluating feed forward deep network for regression using KFold cross validation.

```
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import
KerasRegressor from sklearn.model_selection import
cross_val_score from sklearn.model_selection import
KFold from sklearn.preprocessing import
StandardScaler from sklearn.pipeline import Pipeline
dataframe=pd.read_csv("housing.csv",delim_whitespace=True,header=None)
dataset=dataframe.values
X=dataset[:,0:13
] Y=dataset[:,13]
def
wider_model():
    model=Sequential()

model.add(Dense(15,input_dim=13,kernel_initializer='normal',activation='relu'))
model.add(Dense(13,kernel_initializer='normal',activation='relu'))
model.add(Dense(1,kernel_initializer='normal'))
model.compile(loss='mean_squared_error',optimizer='adam')        return model
estimators=[]
estimators.append(('standardize',StandardScaler()))
estimators.append(('mlp',KerasRegressor(build_fn=wider_model,epochs=100,batch_size=5
))) pipeline=Pipeline(estimators) kfold=KFold(n_splits=10)
results=cross_val_score(pipeline,X,Y,cv=kfold)
print("Wider: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

OUTPUT:

```
Wider: -20.88 (24.29) MSE
(venv) PS D:\keras>
```

(After changing neuron)

model.add(Dense(20, input_dim=13,kernel_initializer='normal',activation='relu'))

```
Wider: -22.17 (24.38) MSE
(venv) PS D:\keras>
```

Practical No:5b

Aim: Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.

```
#loading libraries
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import
cross_val_score from sklearn.model_selection
import KFold from sklearn.preprocessing import
LabelEncoder
#loading dataset
df=pandas.read_csv('Flower.csv',header=None)
print(df)
#splitting dataset into input and output
variables X = df.iloc[:,0:4].astype(float)
y=df.iloc[:,4] #print(X)
#print(y)
#encoding string output into numeric
output encoder=LabelEncoder()
encoder.fit(y)
encoded_y=encoder.transform(y)
print(encoded_y)
```

dummy_Y=np_utils.to_categorical(encoded

_y) print(dummy_Y) def baseline_model():

create model

model = Sequential()

model.add(Dense(8, input_dim=4, activation='relu'))

model.add(Dense(3, activation='softmax'))

Compile model

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

estimator=baseline_model()

estimator.fit(X,dummy_Y,epochs=100,shuffle=True)

action=estimator.predict(X)

for i in range(25):

print(dummy_Y[i])

print('^^^^^^^^^^^^^^^^^^^^^

^') for i in range(25):

print(action[i])

OUTPUT:

```
       0    1    2    3               4
0    5.1  3.5  1.4  0.2     Iris-setosa
1    4.9  3.0  1.4  0.2     Iris-setosa
2    4.7  3.2  1.3  0.2     Iris-setosa
3    4.6  3.1  1.5  0.2     Iris-setosa
4    5.0  3.6  1.4  0.2     Iris-setosa
..   ...  ...  ...  ...             ...
145  6.7  3.0  5.2  2.3  Iris-virginica
146  6.3  2.5  5.0  1.9  Iris-virginica
147  6.5  3.0  5.2  2.0  Iris-virginica
148  6.2  3.4  5.4  2.3  Iris-virginica
149  5.9  3.0  5.1  1.8  Iris-virginica

[150 rows x 5 columns]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
```

```
Epoch 98/100
5/5 [==============================] - 0s 0s/step - loss: 0.3899 - accuracy: 0.9313
Epoch 99/100
5/5 [==============================] - 0s 0s/step - loss: 0.3896 - accuracy: 0.9230
Epoch 100/100
5/5 [==============================] - 0s 0s/step - loss: 0.3682 - accuracy: 0.9361
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
```

```
^^^^^^^^^^^^^^^^^^^^^^
[0.9145307  0.08423453 0.00123477]
[0.88751584 0.1100563  0.00242792]
[0.8999843  0.09803853 0.00197715]
[0.858188   0.13759544 0.00421653]
[0.9138275  0.08489472 0.00127787]
[0.8994011  0.09916449 0.0014343 ]
[0.8872866  0.11023647 0.00247695]
[0.89339536 0.10458492 0.00201967]
[0.8545533  0.14064151 0.00480518]
[0.87742513 0.11963753 0.00293737]
[0.9203753  0.07866727 0.00095734]
[0.8665611  0.1300417  0.00339716]
[0.88403696 0.11323617 0.0027269 ]
[0.9008803  0.09682965 0.00229002]
[9.5539063e-01 4.4350266e-02 2.5906262e-04]
[9.4327897e-01 5.6333560e-02 3.8754733e-04]
[9.3672138e-01 6.2714875e-02 5.6370755e-04]
[0.91191673 0.08680107 0.00128225]
[0.9100969  0.08882014 0.00108295]
[0.91078293 0.08794734 0.00126965]
[0.8827079  0.11510085 0.00219123]
[0.9060573  0.09255142 0.00139134]
[9.3434143e-01 6.4821333e-02 8.3730859e-04]
[0.85551745 0.14102885 0.00345369]
[0.80272377 0.1895675  0.00770868]
```

Code 2:

```
import
pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import
cross_val_score from sklearn.model_selection
import KFold from sklearn.preprocessing import
LabelEncoder

dataset=pandas.read_csv("Flower.csv",header=None) dataset1=dataset.values
X=dataset1[:,0:4].astype(flo
at) Y=dataset1[:,4] print(Y)
encoder=LabelEncoder()
encoder.fit(Y)
encoder_Y=encoder.transform(Y)
print(encoder_Y)
dummy_Y=np_utils.to_categorical(encoder_
Y) print(dummy_Y) def baseline_model():
        model=Sequential()
        model.add(Dense(8,input_dim=4,activation='relu'))
model.add(Dense(3,activation='softmax'))
        model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
return model
estimator=KerasClassifier(build_fn=baseline_model,epochs=100,batch_size=5)
kfold = KFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X, dummy_Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
3/3 [==============================] - 0s 2ms/step - loss: 0.2491 - accuracy: 0.9333
Baseline: 96.00% (4.42%)
```

(Changing neuron)

model.add(Dense(10,input_dim=4,activation='relu'))

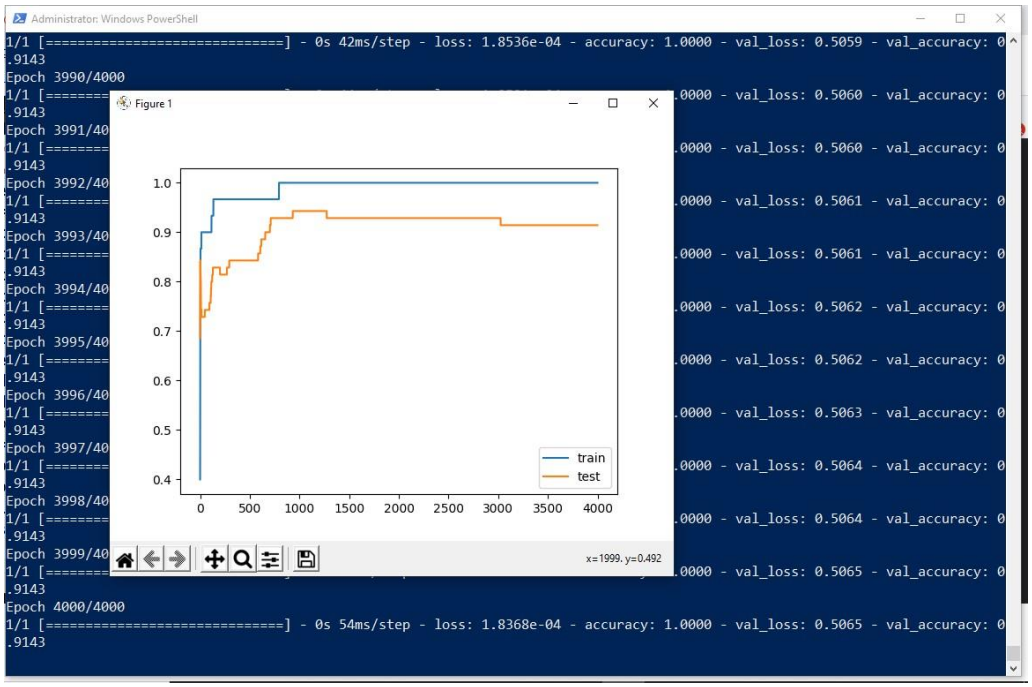
```
3/3 [==============================] - 0s 999us/step - loss: 0.1436 - accuracy: 1.0000
Baseline: 98.67% (2.67%)
```

Practical No :6

Aim: implementing regularization to avoid overfitting in binary classification.

```
from matplotlib import pyplot from
sklearn.datasets import make_moons
from keras.models import Sequential
from keras.layers import Dense
X,Y=make_moons(n_samples=100,noise=0.2,random_state=1)
n_train=30
trainX,testX=X[:n_train,:],X[n_train:]
trainY,testY=Y[:n_train],Y[n_train:]
#print(trainX)
#print(trainY)
#print(testX)
#print(testY)
model=Sequential()
model.add(Dense(500,input_dim=2,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
history=model.fit(trainX,trainY,validation_data=(testX,testY),epochs=4000)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test') pyplot.legend()
pyplot.show()
```

OUTPUT:



The above code and resultant graph demonstrate overfitting with accuracy of testing data less than accuracy of training data also the accuracy of testing data increases once and then start decreases gradually.to solve this problem we can use regularization

Hence, we will add two lines in the above code as highlighted below to implement l2 regularization with alpha=0.001

```
from matplotlib import pyplot from
sklearn.datasets import make_moons
from keras.models import Sequential
from keras.layers import Dense from
keras.regularizers import l2
X,Y=make_moons(n_samples=100,noise=0.2,random_state=1)
n_train=30
trainX,testX=X[:n_train,:],X[n_train:]
trainY,testY=Y[:n_train],Y[n_train:]
#print(trainX)
#print(trainY)
#print(testX)
#print(testY)
model=Sequential()
```

model.add(Dense(500,input_dim=2,activation='relu',**kernel_regularizer=l2(0.001)))**

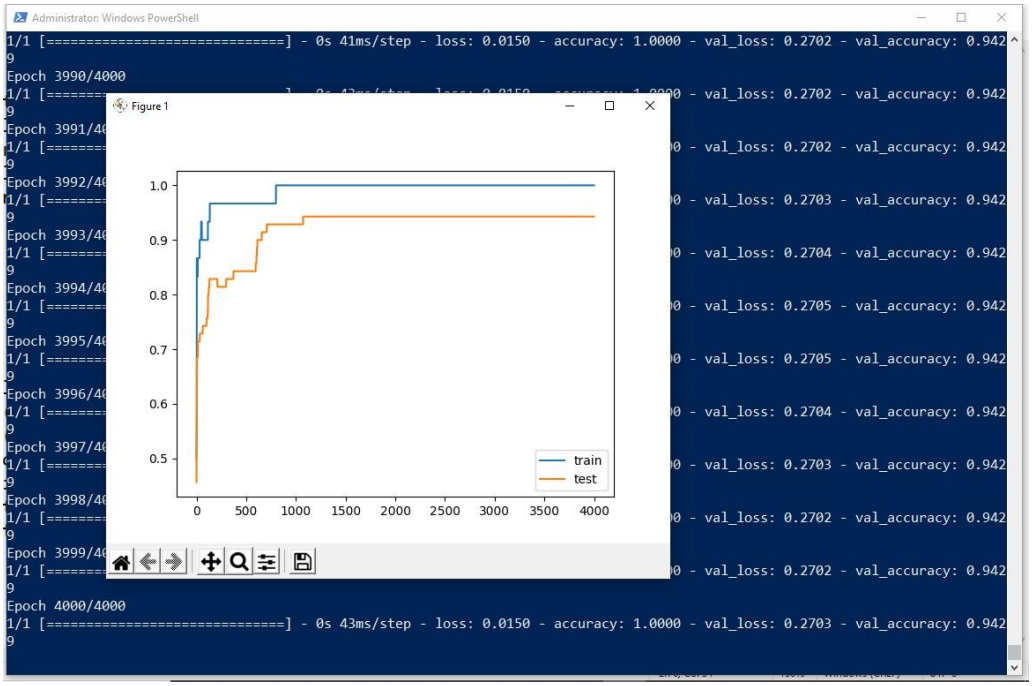
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

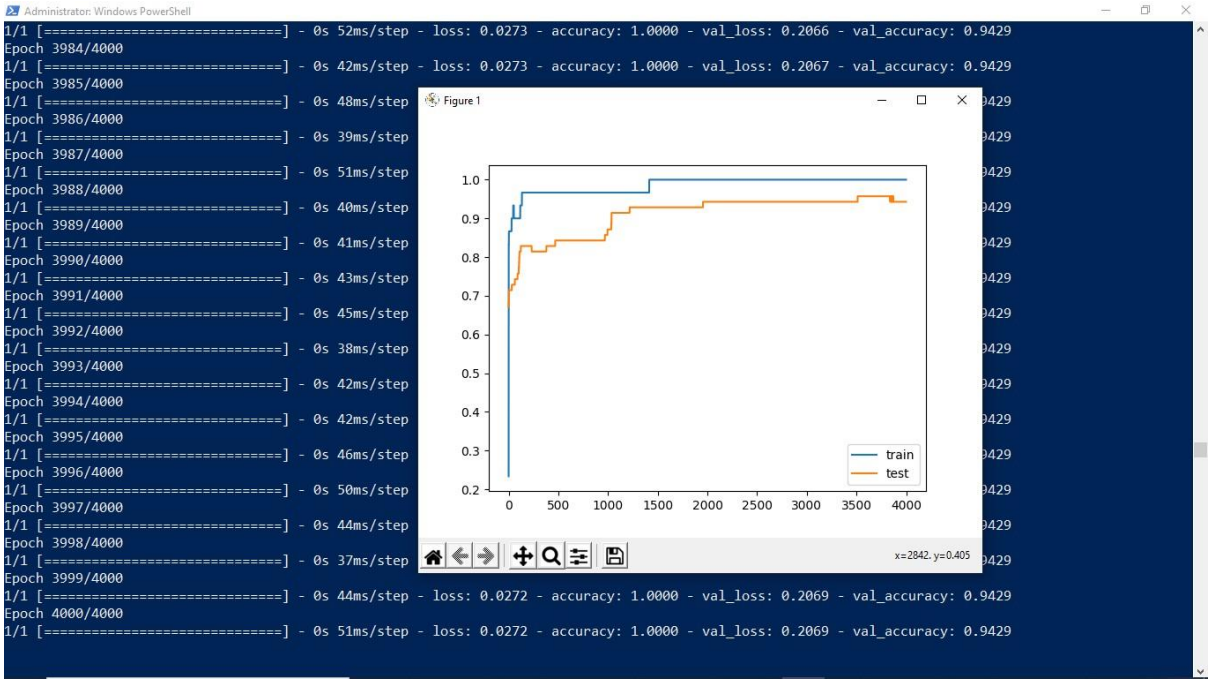
history=model.fit(trainX,trainY,validation_data=(testX,testY),epochs=4000)

pyplot.plot(history.history['accuracy'],label='train') pyplot.plot(history.history['val_accuracy'],label='test')

pyplot.legend() pyplot.show()



By replacing l2 regularizer with l1 regularizer at the same learning rate 0.001 we get the following output.



By applying l1 and l2 regularizer we can observe the following changes in accuracy of both trainig and testing data. The changes in code are also highlighted.

```
from matplotlib import pyplot from
sklearn.datasets import make_moons
from keras.models import Sequential
from keras.layers import Dense from
keras.regularizers import l1_l2
X,Y=make_moons(n_samples=100,noise=0.2,random_state=1)
n_train=30
trainX,testX=X[:n_train,:],X[n_train:]
trainY,testY=Y[:n_train],Y[n_train:]
#print(trainX)
#print(trainY)
#print(testX)
#print(testY)
model=Sequential()
model.add(Dense(500,input_dim=2,activation='relu',kernel_regularizer=l1_l2(l1=0.001,l2=0.001)))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
history=model.fit(trainX,trainY,validation_data=(testX,testY),epochs=4000)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test') pyplot.legend()
pyplot.show()
```

OUTPUT:

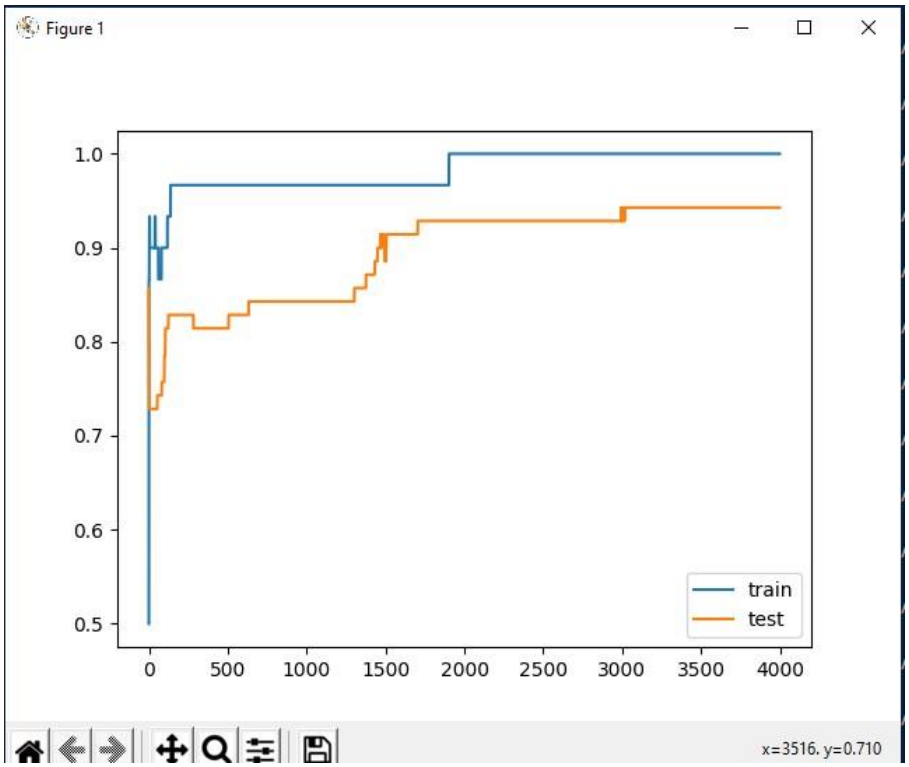


Figure 1
1.0
0.9
0.8
0.7
0.6
0.5
0
500
1000
1500
2000
2500
3000
3500
4000
train
test
x=3516. y=0.710

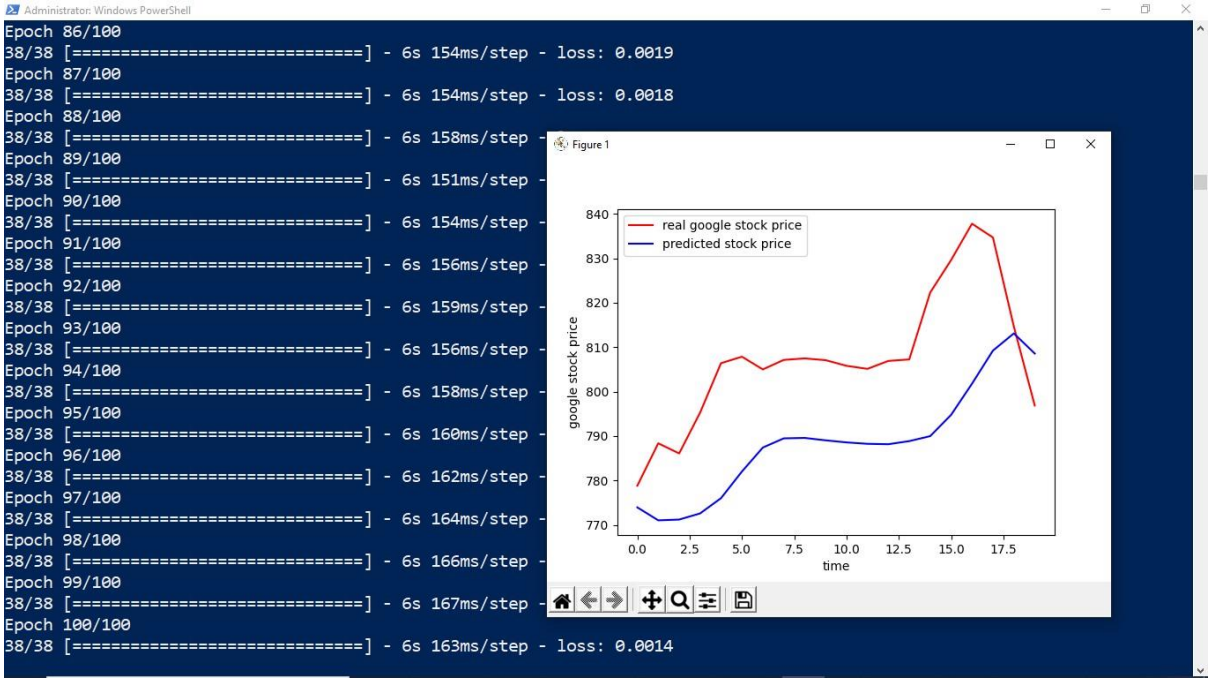
Practical No:7

Aim: Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.

```
import numpy as np import matplotlib.pyplot as plt
import pandas as pd from keras.models import
Sequential from keras.layers import Dense from
keras.layers import LSTM from keras.layers import
Dropout from sklearn.preprocessing import
MinMaxScaler
dataset_train=pd.read_csv('Google_Stock_price_train.cs
v')
#print(dataset_train)
training_set=dataset_train.iloc[:,1:2].values

#print(training_set)
sc=MinMaxScaler(feature_range=(0,1))
training_set_scaled=sc.fit_transform(training_set)
#print(training_set_scaled)
X_train=[]
Y_train=[] for i in
range(60,1258):
    X_train.append(training_set_scaled[i-60:i,0])
    Y_train.append(training_set_scaled[i,0])
X_train,Y_train=np.array(X_train),np.array(Y_train)
print(X_train)
print('**********************************************')
print(Y_train)
X_train=np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
print('**********************************************') print(X_train)
regressor=Sequential()
regressor.add(LSTM(units=50,return_sequences=True,input_shape=(X_train.shape[1],1
))) regressor.add(Dropout(0.2)) regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2)) regressor.add(LSTM(units=50,return_sequences=True))
```

```
regressor.add(Dropout(0.2)) regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2)) regressor.add(Dense(units=1))

regressor.compile(optimizer='adam',loss='mean_squared_error')
regressor.fit(X_train,Y_train,epochs=100,batch_size=32)

dataset_test=pd.read_csv('Google_Stock_price_Test.csv')
real_stock_price=dataset_test.iloc[:,1:2].values
dataset_total=pd.concat((dataset_train['Open'],dataset_test['Open']),axis
=0) inputs=dataset_total[len(dataset_total)-len(dataset_test)-60:].values
inputs=inputs.reshape(-1,1) inputs=sc.transform(inputs) X_test=[] for i
in range(60,80):
    X_test.append(inputs[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
predicted_stock_price=regressor.predict(X_test)

predicted_stock_price=sc.inverse_transform(predicted_stock_price)
plt.plot(real_stock_price,color='red',label='real google stock price')
plt.plot(predicted_stock_price,color='blue',label='predicted stock
price') plt.xlabel('time') plt.ylabel('google stock price') plt.legend()
plt.show()
```

OUTPUT:


Administrator: Windows PowerShell
Epoch 86/100
38/38 [==============================] - 6s 154ms/step - loss: 0.0019
Epoch 87/100
38/38 [==============================] - 6s 154ms/step - loss: 0.0018
Epoch 88/100
38/38 [==============================] - 6s 158ms/step -
Epoch 89/100
38/38 [==============================] - 6s 151ms/step -
Epoch 90/100
38/38 [==============================] - 6s 154ms/step -
Epoch 91/100
38/38 [==============================] - 6s 156ms/step -
Epoch 92/100
38/38 [==============================] - 6s 159ms/step -
Epoch 93/100
38/38 [==============================] - 6s 156ms/step -
Epoch 94/100
38/38 [==============================] - 6s 158ms/step -
Epoch 95/100
38/38 [==============================] - 6s 160ms/step -
Epoch 96/100
38/38 [==============================] - 6s 162ms/step -
Epoch 97/100
38/38 [==============================] - 6s 164ms/step -
Epoch 98/100
38/38 [==============================] - 6s 166ms/step -
Epoch 99/100
38/38 [==============================] - 6s 167ms/step -
Epoch 100/100
38/38 [==============================] - 6s 163ms/step - loss: 0.0014
Figure 1
real google stock price
predicted stock price
google stock price
time
840
830
820
810
800
790
780
770
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Practical No:8

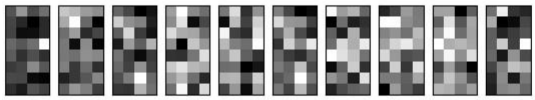
Aim: Performing encoding and decoding of images using deep autoencoder.

```
import keras from keras
import layers from
keras.datasets import mnist
import numpy as np
encoding_dim=32
#this is our input image
input_img=keras.Input(shape=(784,))
#"encoded" is the encoded representation of the input
encoded=layers.Dense(encoding_dim, activation='relu')(input_img)
#"decoded" is the lossy reconstruction of the input
decoded=layers.Dense(784, activation='sigmoid')(encoded)
#creating autoencoder model
autoencoder=keras.Model(input_img,decoded)
#create the encoder model
encoder=keras.Model(input_img,encoded)
encoded_input=keras.Input(shape=(encoding_dim,)) #Retrive the
last layer of the autoencoder model
decoder_layer=autoencoder.layers[-1] #create the decoder model
decoder=keras.Model(encoded_input,decoder_layer(encoded_inpu
t))
autoencoder.compile(optimizer='adam',loss='binary_crossentropy'
)
#scale and make train and test dataset
(X_train,_),(X_test,_)=mnist.load_data()
X_train=X_train.astype('float32')/255.
X_test=X_test.astype('float32')/255.
X_train=X_train.reshape((len(X_train),np.prod(X_train.shape[1:]
)))
X_test=X_test.reshape((len(X_test),np.prod(X_test.shape[1:])))
print(X_train.shape) print(X_test.shape)
```

```
#train autoencoder with training dataset
autoencoder.fit(X_train,X_train,
epochs=50,                batch_size=256,
shuffle=True,
validation_data=(X_test,X_test))
encoded_imgs=encoder.predict(X_test)
decoded_imgs=decoder.predict(encoded_im
gs) import matplotlib.pyplot as plt n = 10 #
How many digits we will display
plt.figure(figsize=(40, 4)) for i in range(10):
# display original     ax = plt.subplot(3, 20, i
+ 1)     plt.imshow(X_test[i].reshape(28,
28))     plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)     # display
encoded image     ax = plt.subplot(3, 20, i +
1 + 20)
plt.imshow(encoded_imgs[i].reshape(8,4))
plt.gray()     ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)     # display
reconstruction     ax = plt.subplot(3, 20,
2*20 +i+ 1)
plt.imshow(decoded_imgs[i].reshape(28,
28))     plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False) plt.show()
```

OUTPUT:


Administrator: Windows PowerShell
235/235 [==============================] - 3s 13ms/step - loss: 0.0929 - val_loss: 0.0918
Epoch 34/50
235/235 [==============================] - 3s 13ms/step - loss: 0.0926 - val_loss: 0.0917
Epoch 35/50
235/235 [==============================] - 3s 13ms/step - loss: 0.0928 - val_loss: 0.0917
Epoch 36/50
Epoch 37/50
Epoch 38/50
Epoch 39/50
Epoch 40/50
Epoch 41/50
Epoch 42/50
Epoch 43/50
Epoch 44/50
Epoch 45/50
Epoch 46/50
Epoch 47/50
Figure 1
Epoch 48/50
235/235 [==============================] - 3s 14ms/step - loss: 0.0927 - val_loss: 0.0915
Epoch 49/50
235/235 [==============================] - 3s 13ms/step - loss: 0.0927 - val_loss: 0.0915
Epoch 50/50
235/235 [==============================] - 3s 15ms/step - loss: 0.0926 - val_loss: 0.0914



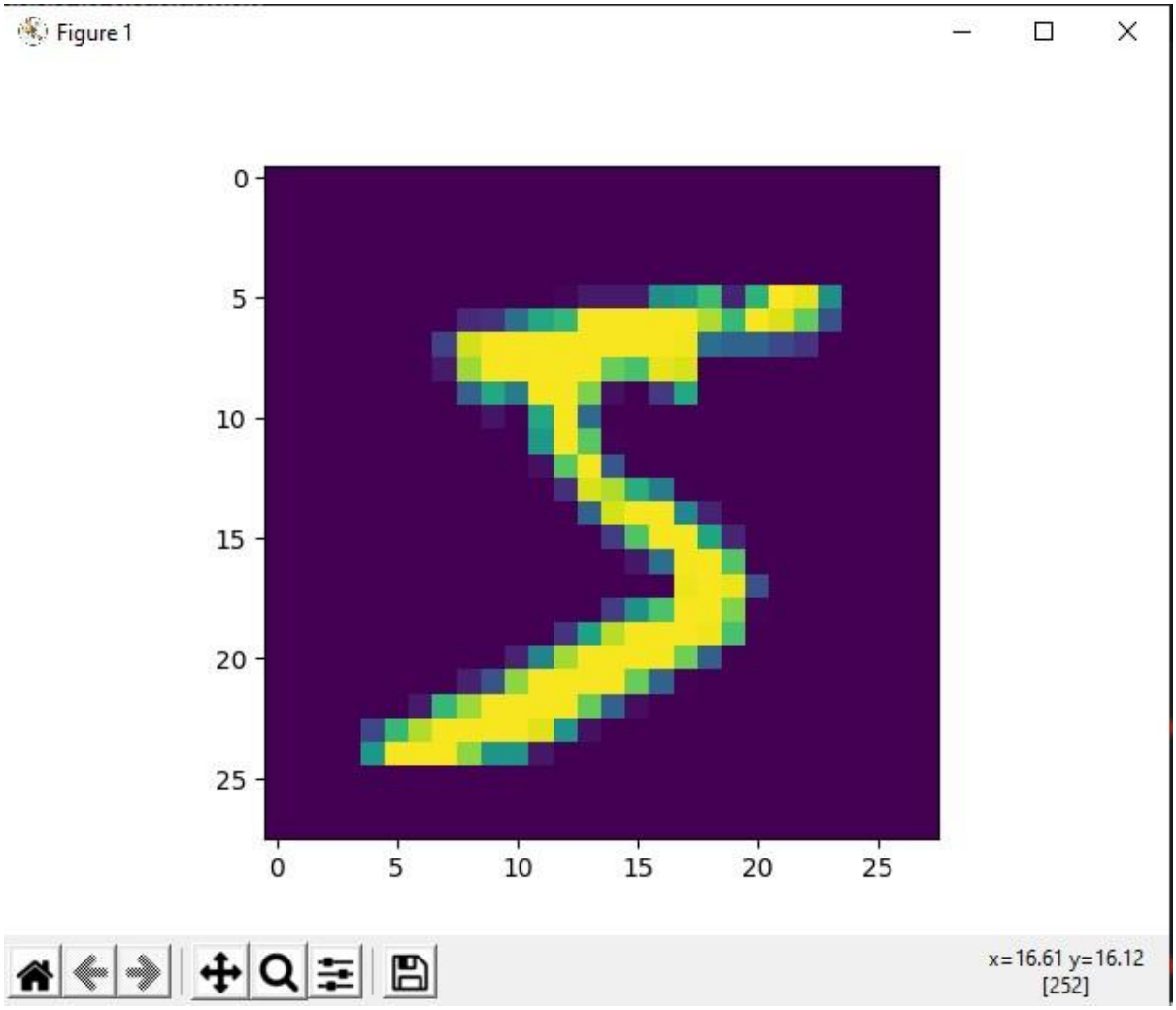
Practical No:9

Aim: Implementation of convolutional neural network to predict numbers from number images

```
from keras.datasets import mnist from
keras.utils import to_categorical from
keras.models import Sequential from
keras.layers import Dense,Conv2D,Flatten
import matplotlib.pyplot as plt

#download mnist data and split into train and test sets
(X_train,Y_train),(X_test,Y_test)=mnist.load_data()

#plot the first image in the
dataset plt.imshow(X_train[0])
plt.show()

print(X_train[0].shape)
X_train=X_train.reshape(60000,28,28,1)
X_test=X_test.reshape(10000,28,28,1)

Y_train=to_categorical(Y_train)
Y_test=to_categorical(Y_t
est) Y_train[0]
print(Y_train[0])

model=Sequential()
#add model layers
#learn image features
model.add(Conv2D(64,kernel_size=3,activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=3,activation='relu'))

model.add(Flatten())
```

```
model.add(Dense(10,activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

#train

model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=3)


print(model.predict(X_test[:4]))

#actual results for 1st 4 images in the test set

print(Y_test[:4])
```

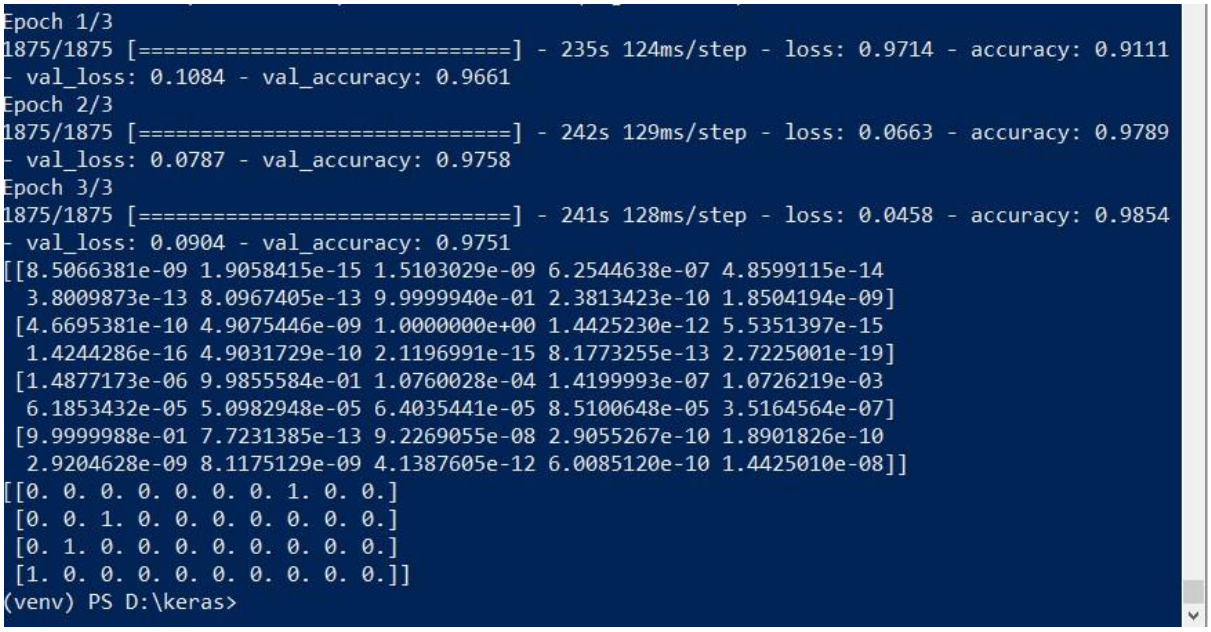
OUTPUT:



(28, 28)

[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]

```
(venv) PS D:\keras> python pract6.py
(28, 28)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```


Epoch 1/3
1875/1875 [==============================] - 235s 124ms/step - loss: 0.9714 - accuracy: 0.9111
- val_loss: 0.1084 - val_accuracy: 0.9661
Epoch 2/3
1875/1875 [==============================] - 242s 129ms/step - loss: 0.0663 - accuracy: 0.9789
- val_loss: 0.0787 - val_accuracy: 0.9758
Epoch 3/3
1875/1875 [==============================] - 241s 128ms/step - loss: 0.0458 - accuracy: 0.9854
- val_loss: 0.0904 - val_accuracy: 0.9751
[[8.5066381e-09 1.9058415e-15 1.5103029e-09 6.2544638e-07 4.8599115e-14
3.8009873e-13 8.0967405e-13 9.9999940e-01 2.3813423e-10 1.8504194e-09]
[4.6695381e-10 4.9075446e-09 1.0000000e+00 1.4425230e-12 5.5351397e-15
1.4244286e-16 4.9031729e-10 2.1196991e-15 8.1773255e-13 2.7225001e-19]
[1.4877173e-06 9.9855584e-01 1.0760028e-04 1.4199993e-07 1.0726219e-03
6.1853432e-05 5.0982948e-05 6.4035441e-05 8.5100648e-05 3.5164564e-07]
[9.9999988e-01 7.7231385e-13 9.2269055e-08 2.9055267e-10 1.8901826e-10
2.9204628e-09 8.1175129e-09 4.1387605e-12 6.0085120e-10 1.4425010e-08]]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
(venv) PS D:\keras>

Practical No:10

Aim: Denoising of images using autoencoder.

```
import keras from
keras.datasets import mnist
from keras import layers
import numpy as np
from keras.callbacks import TensorBoard
import matplotlib.pyplot as plt
(X_train,_),(X_test,_)=mnist.load_data()
X_train=X_train.astype('float32')/255.
X_test=X_test.astype('float32')/255.
X_train=np.reshape(X_train,(len(X_train),28,28,1))
X_test=np.reshape(X_test,(len(X_test),28,28,1))
noise_factor=0.5
X_train_noisy=X_train+noise_factor*np.random.normal(loc=0.0,scale=1.0,size=X_train.shape)
X_test_noisy=X_test+noise_factor*np.random.normal(loc=0.0,scale=1.0,size=X_test.shape)
X_train_noisy=np.clip(X_train_noisy,0.,1.)
X_test_noisy=np.clip(X_test_noisy,0.,1.)
n=10
plt.figure(figsize=(20,2)) for i in
range(1,n+1):     ax=plt.subplot(1,n,i)
plt.imshow(X_test_noisy[i].reshape(28,28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
plt.show()
input_img=keras.Input(shape=(28,28,1))
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(input_img)
x=layers.MaxPooling2D((2,2),padding='same')(x)
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x)
encoded=layers.MaxPooling2D((2,2),padding='same')(x)
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(encoded)
x=layers.UpSampling2D((2,2))(x)
```

```
x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x)
x=layers.UpSampling2D((2,2))(x)
decoded=layers.Conv2D(1,(3,3),activation='sigmoid',padding='same')(x)
autoencoder=keras.Model(input_img,decoded)
autoencoder.compile(optimizer='adam',loss='binary_crossentropy')
autoencoder.fit(X_train_noisy,X_train,
        epochs=3,
batch_size=128,
shuffle=True,
        validation_data=(X_test_noisy,X_test),
callbacks=[TensorBoard(log_dir='/tmo/tb',histogram_freq=0,write_graph=False
)]) predictions=autoencoder.predict(X_test_noisy) m=10
plt.figure(figsize=(20,2)) for i in
range(1,m+1):     ax=plt.subplot(1,m,i)
plt.imshow(predictions[i].reshape(28,28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
plt.show()
```

OUTPUT:

After 3 epochs: