INDEX

	List of Practical:	Page no	Sign
1.	Write the following programs for Blockchain in Python:		
a.	A simple client class that generates the private and public keys by using the built- in Python RSA algorithm and test it.		
b.	A transaction class to send and receive money and test it.		
c.	Create multiple transactions and display them.		
d.	Create a blockchain, a genesis block and execute it.		
e.	Create a mining function and test it.		
f.	Add blocks to the miner and dump the blockchain.		
2.	Implement and demonstrate the use of the following in Solidity:		
a.	Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.		
b.	Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.		
3.	Implement and demonstrate the use of the following in Solidity:		
a.	Withdrawal Pattern, Restricted Access.		
b.	Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.		
c.	Libraries, Assembly, Events, Error handling.		
4.	Install hyperledger fabric and composer. Deploy and execute the application.		
5.	Demonstrate the running of the blockchain node.		
6.	Demonstrate the use of Bitcoin Core API.		
7.	Create your own blockchain and demonstrate its use.		

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:

```
sender: 30819f300d06092a864886f70d010101050003818d00308189028181 00c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b303
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a
-----
value: 15.0
-----
time: 2022-07-30 17:35:06.809777
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d00308189028181 00c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b303
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00a72cfb527dffd9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2
-----
value: 6.0
-----
time: 2022-07-30 17:35:06.811866
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d00308189028181 00d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a814
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d578
-----
value: 2.0
-----
time: 2022-07-30 17:35:06.814247
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d00308189028181 00a72cfb527dffd9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2b268
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a
-----
value: 4.0
-----
time: 2022-07-30 17:35:06.816476
-----
--------------
```

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
```

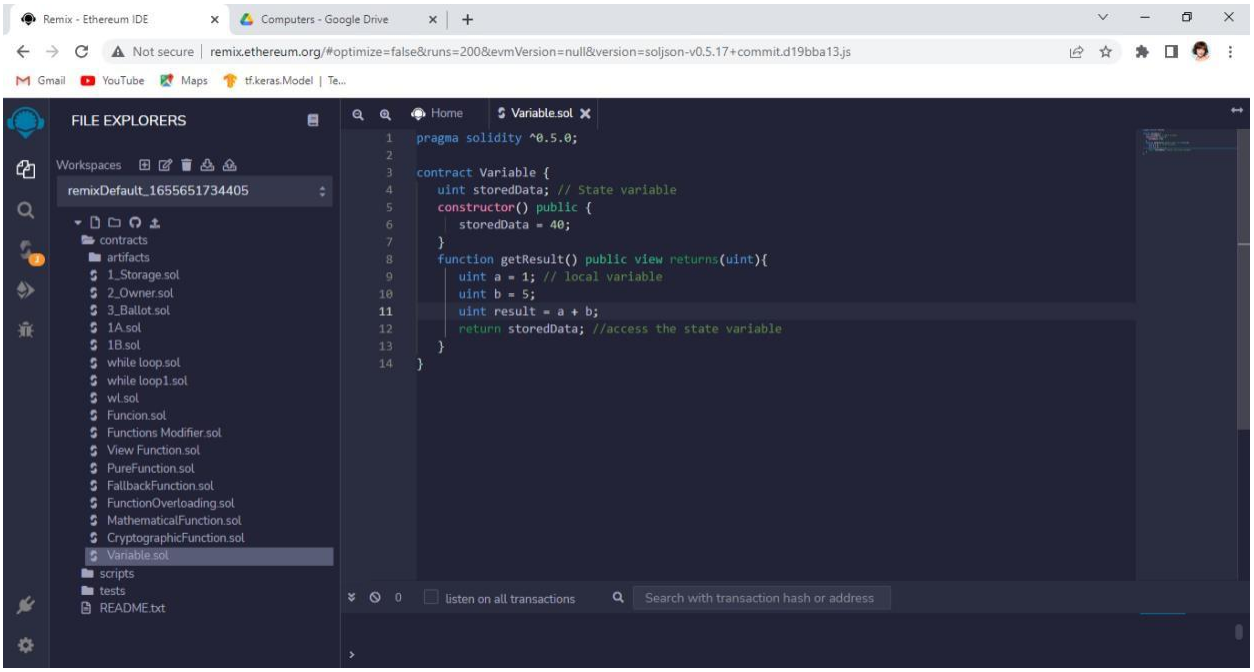
Practical No: 2

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

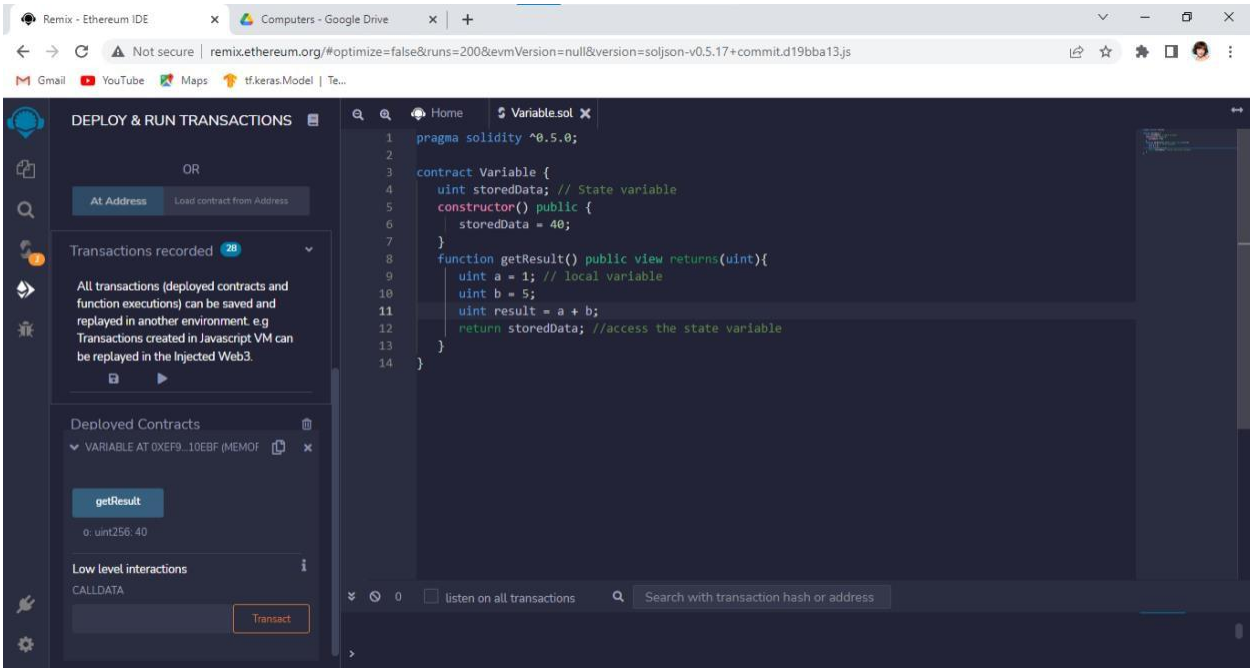
2a) Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. Variable

<u>Variable</u>

Code:



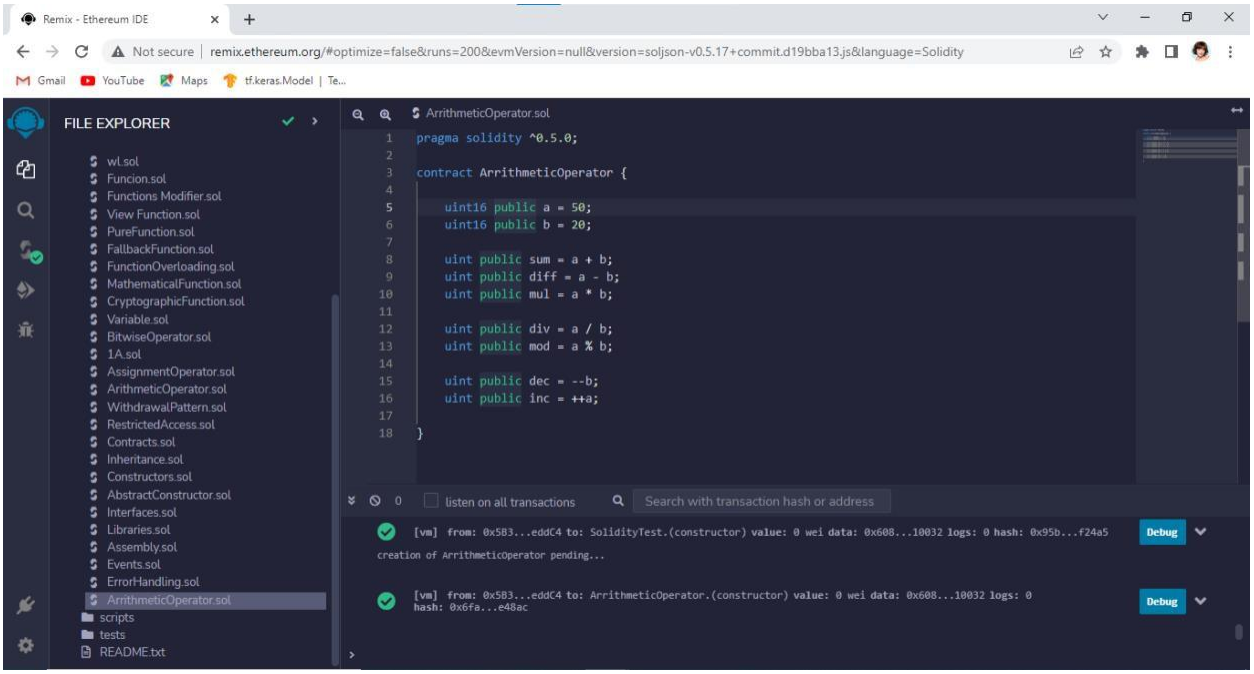
Output:


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 28
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
VARIABLE AT 0XEF9...10EBF (MEMOI
getResult
0: uint256: 40
Low level interactions
CALLDATA
Transact
Home
Variable.sol
pragma solidity ^0.5.0;
contract Variable {
uint storedData; // State variable
constructor() public {
storedData = 40;
}
function getResult() public view returns(uint){
uint a = 1; // local variable
uint b = 5;
uint result = a + b;
return storedData; //access the state variable
}
}
listen on all transactions
Search with transaction hash or address

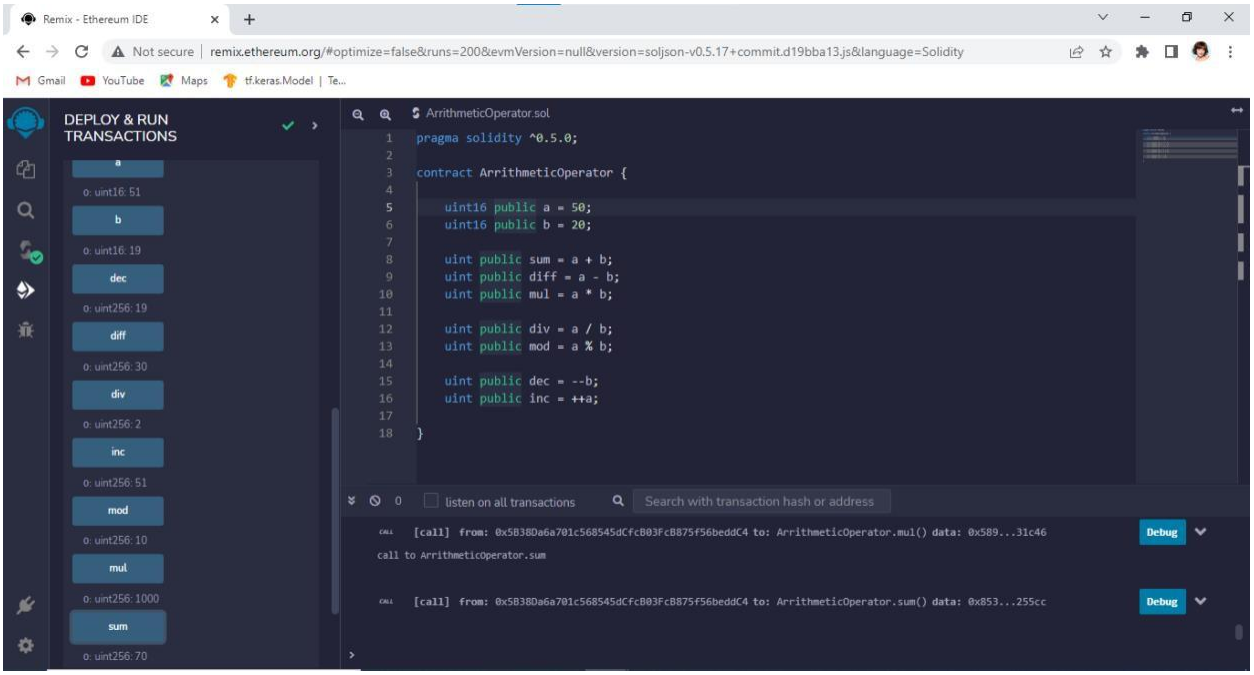
Operators:

Arithmetic Operator

Code:

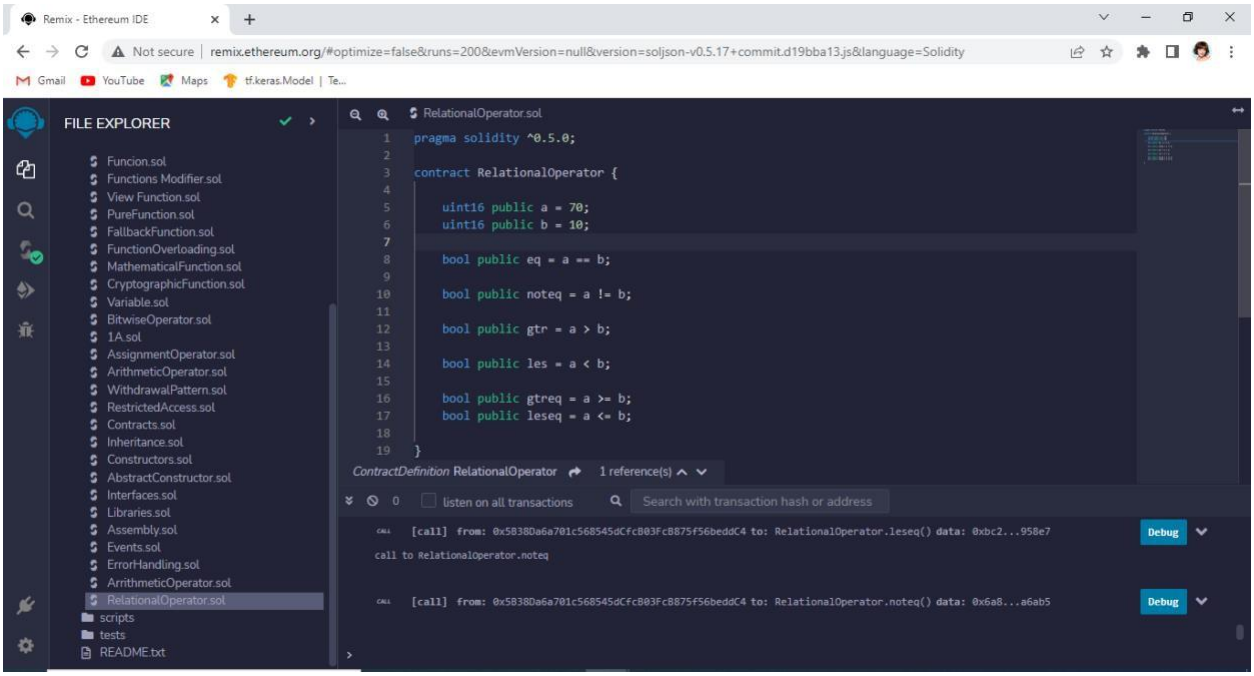


Output:

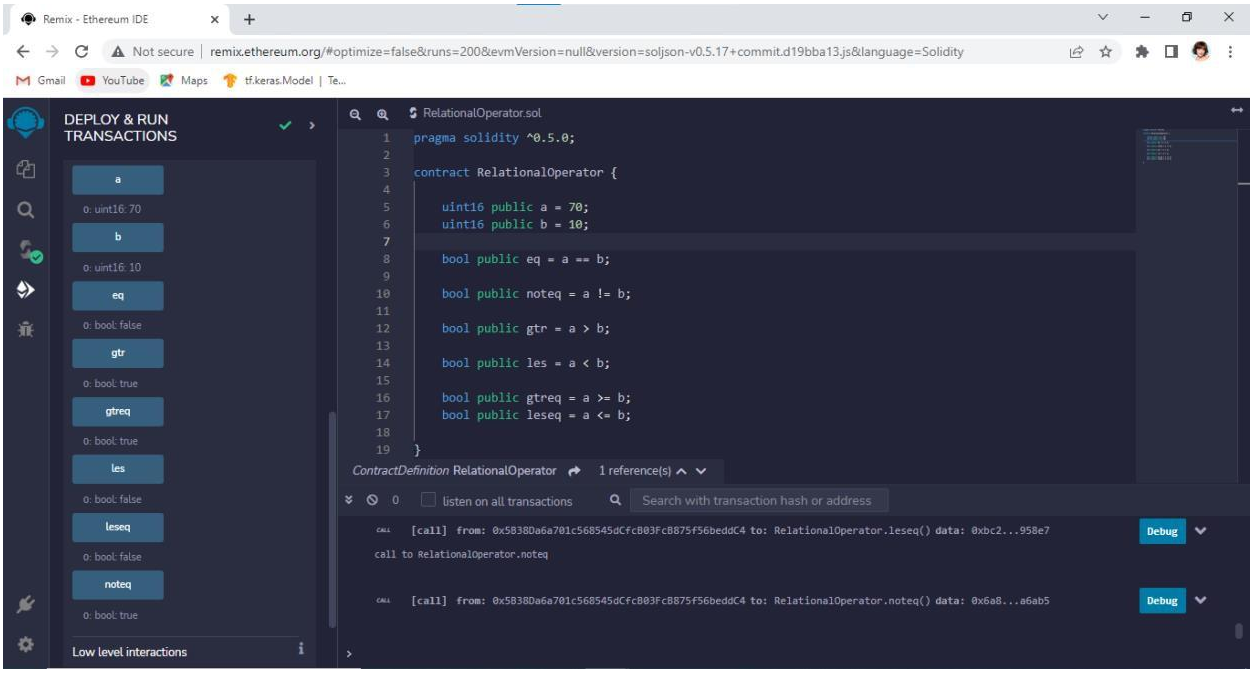


Relational Operator

Code:

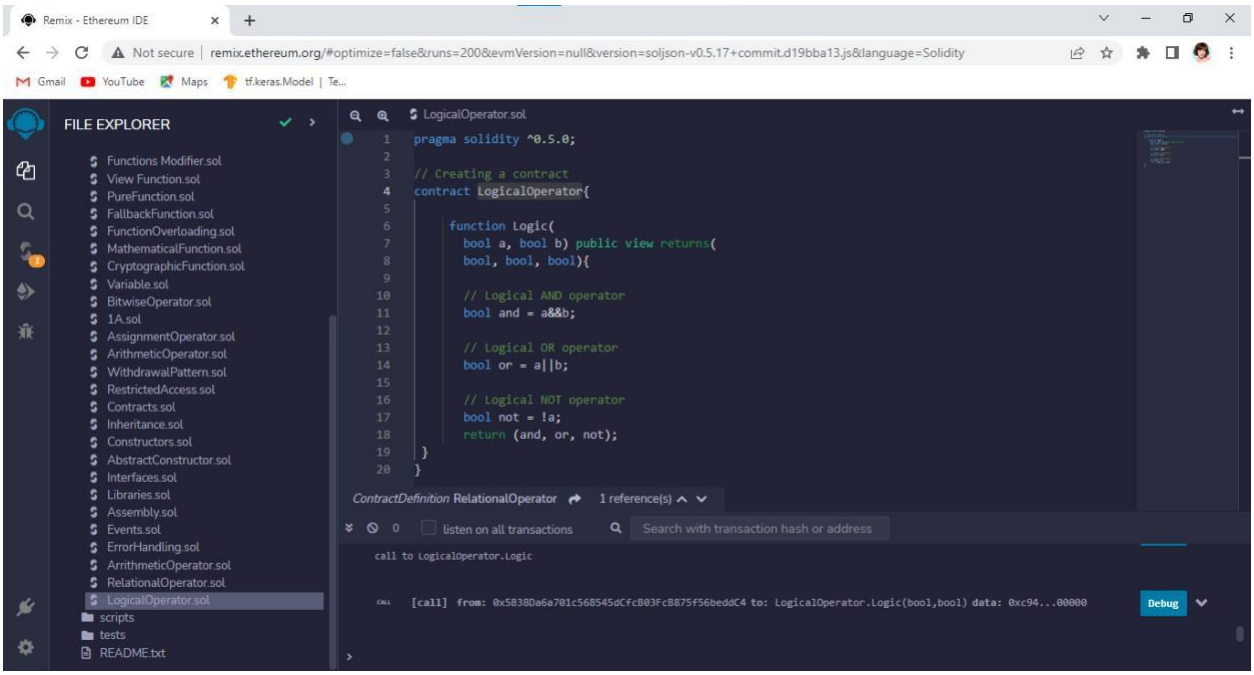


Output:

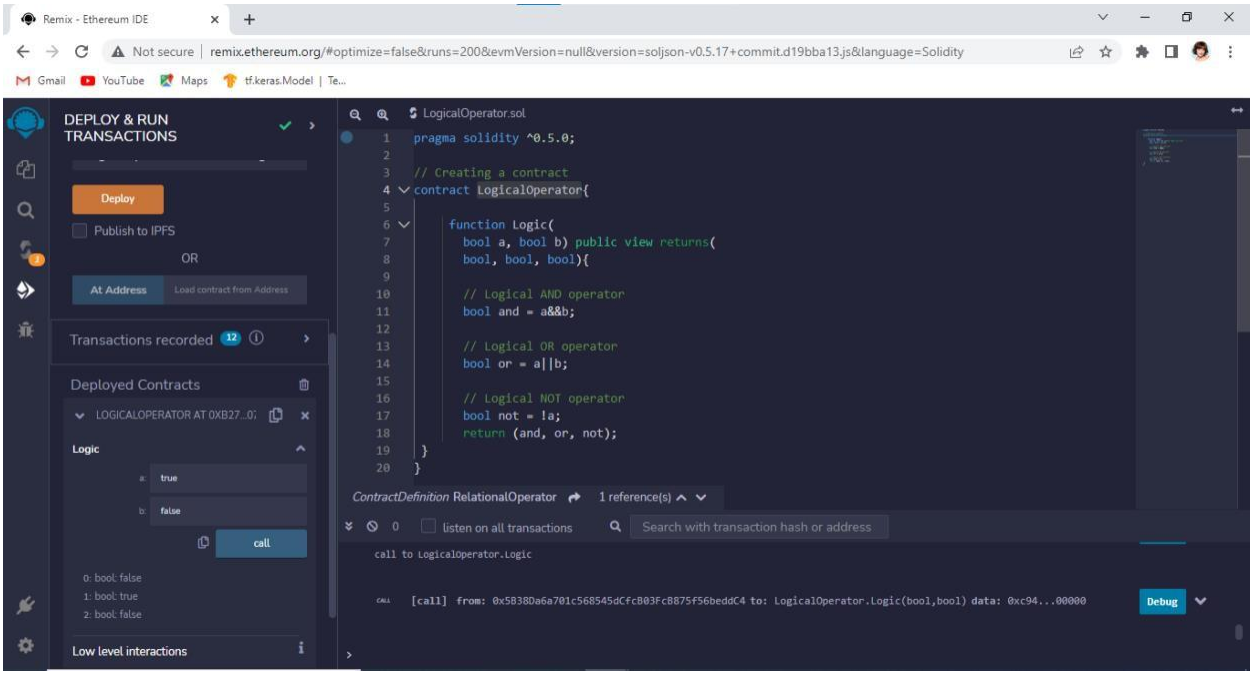


Logical Operator

Code:

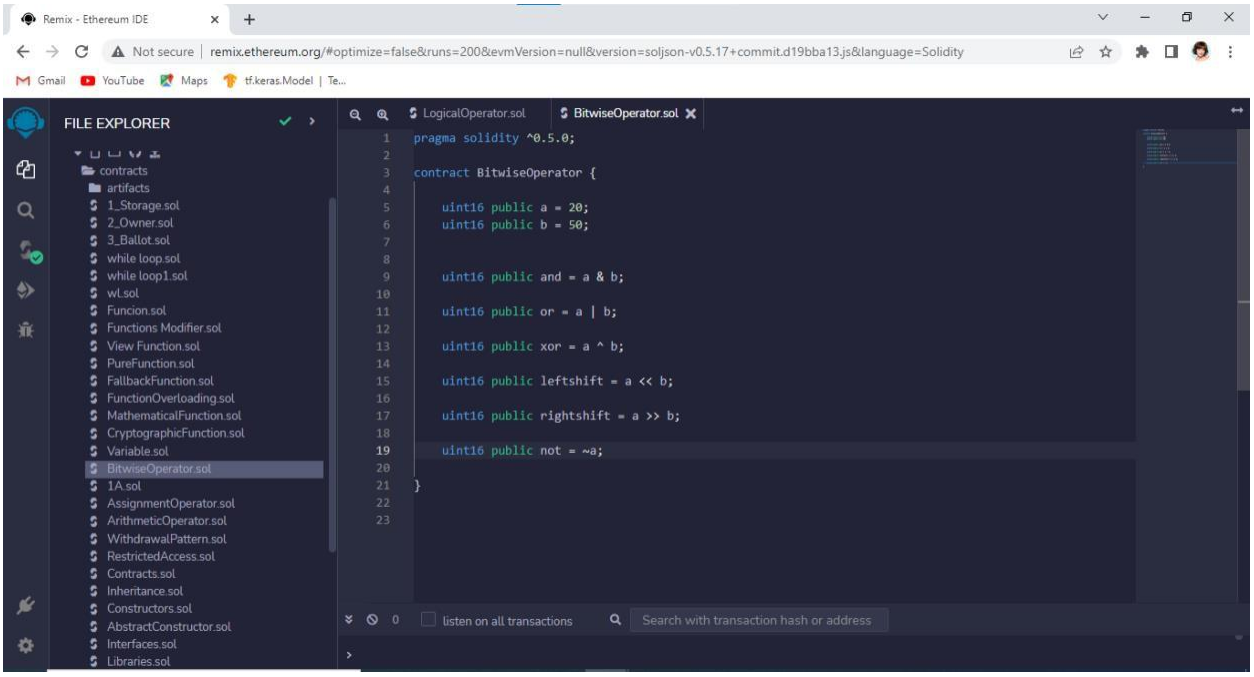


Output:

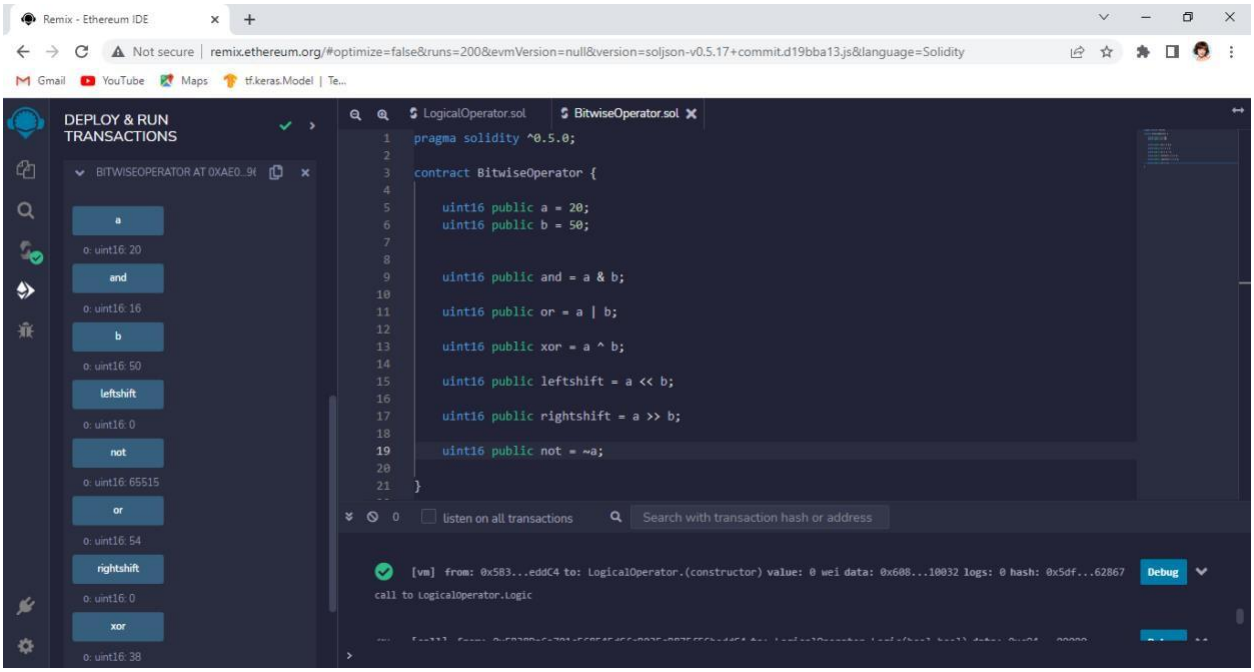


Bitwise Operator

Code:

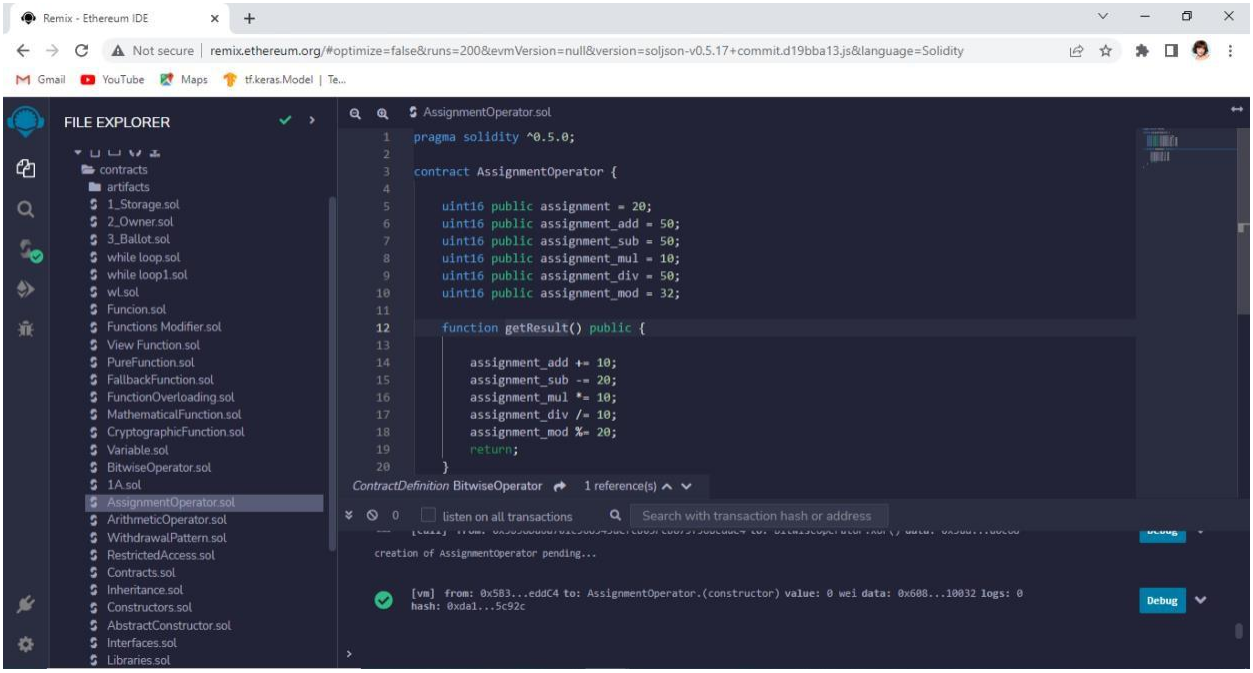


Output:

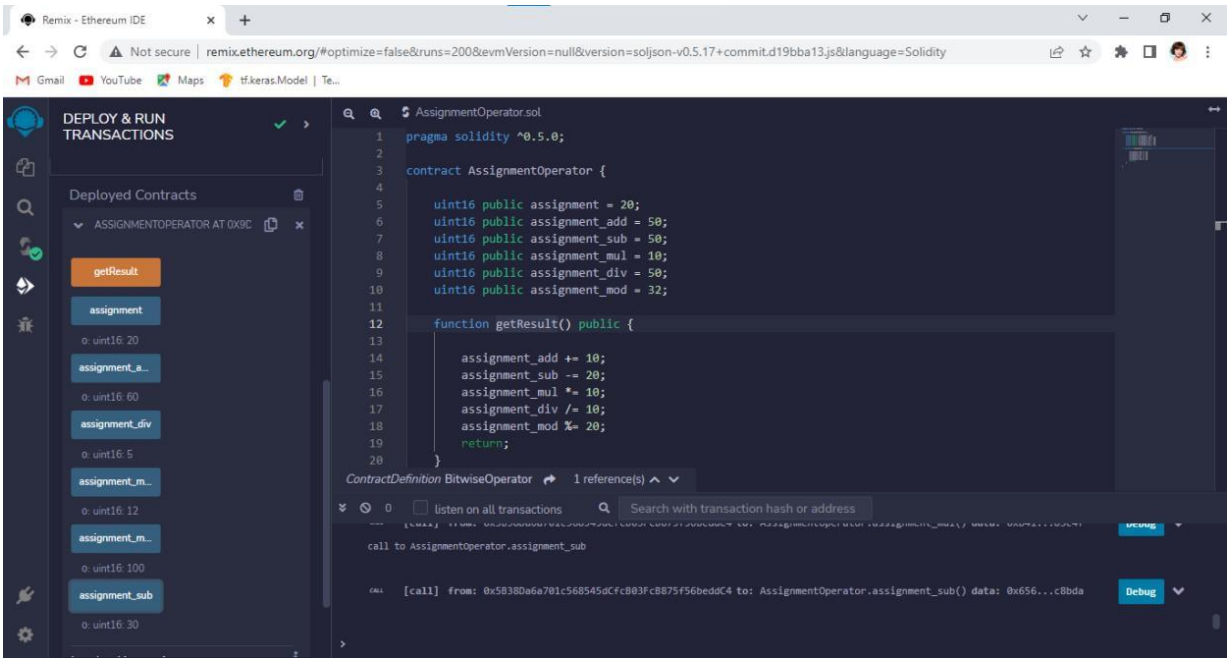


Assignment Operator

Code:



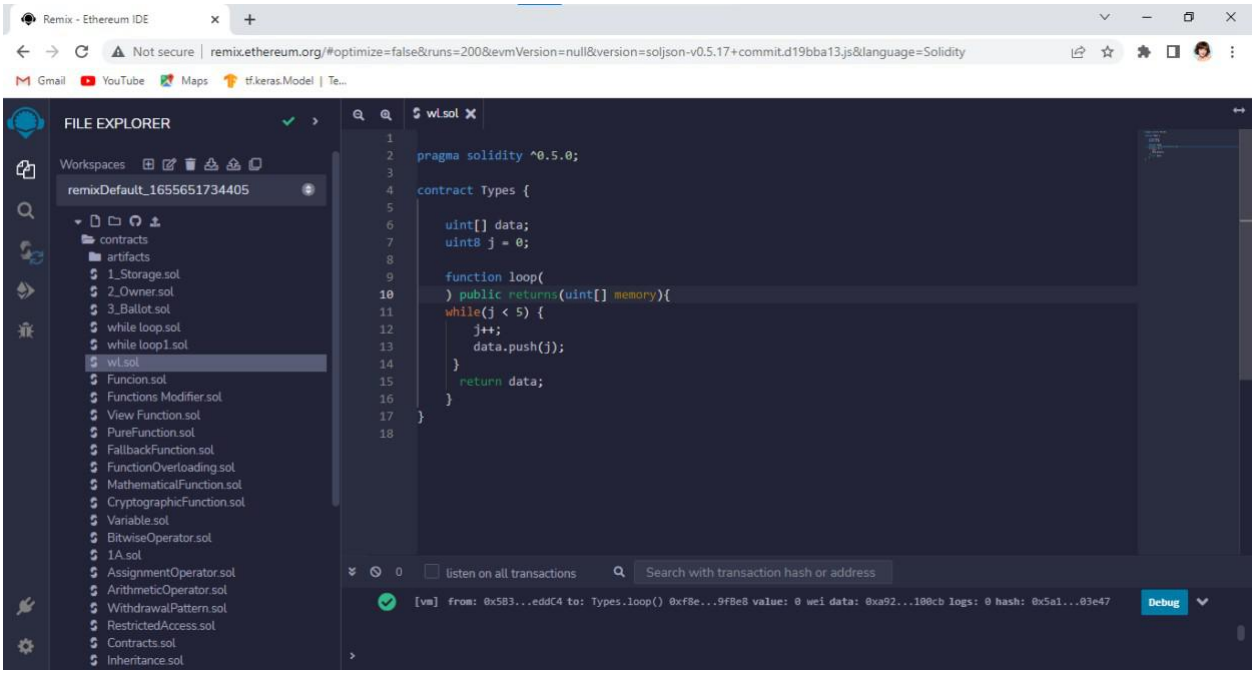
Output:



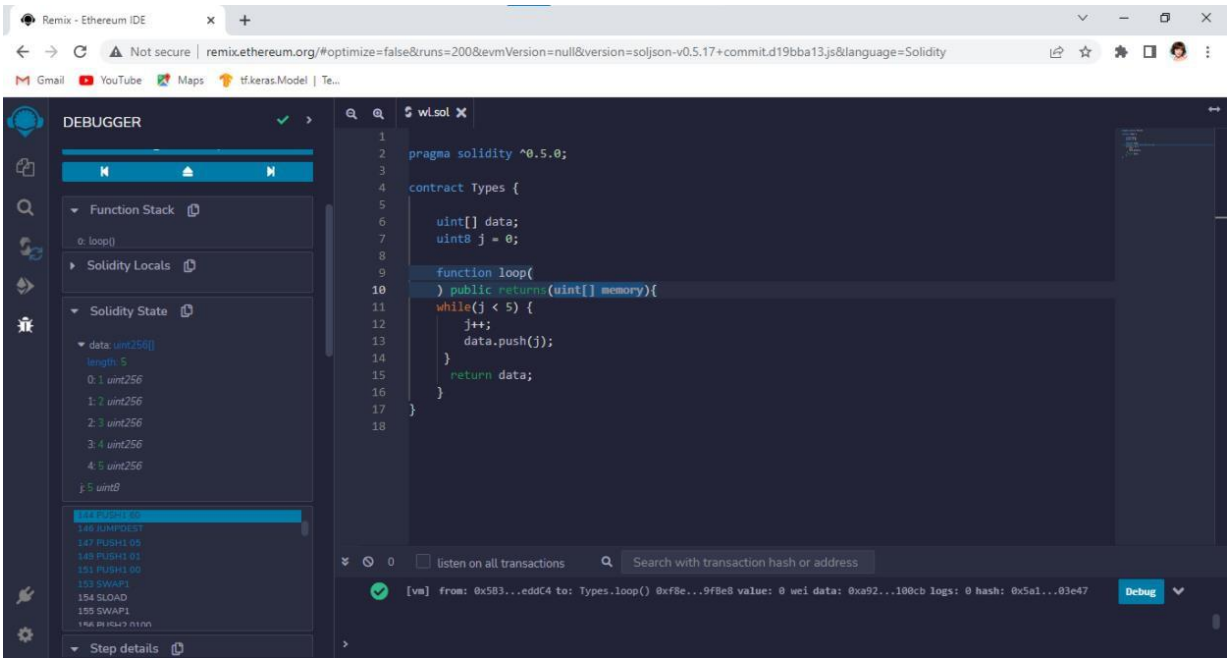
Loops

While Loop

Code:

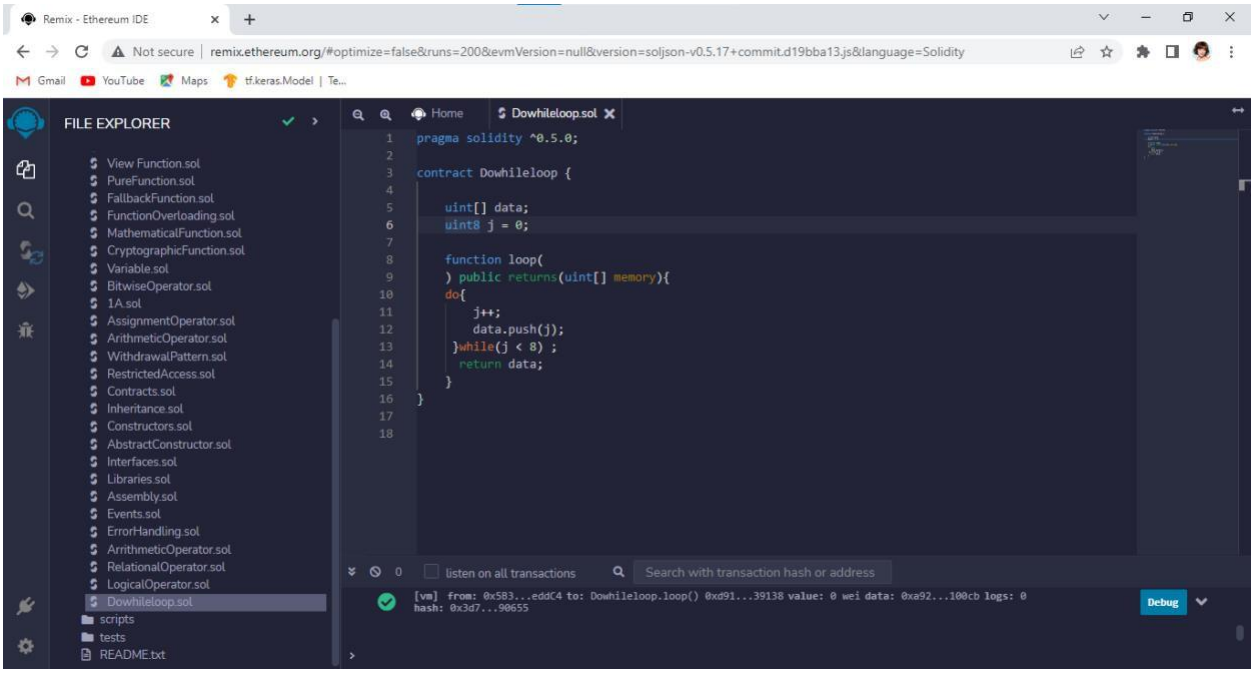


Output:

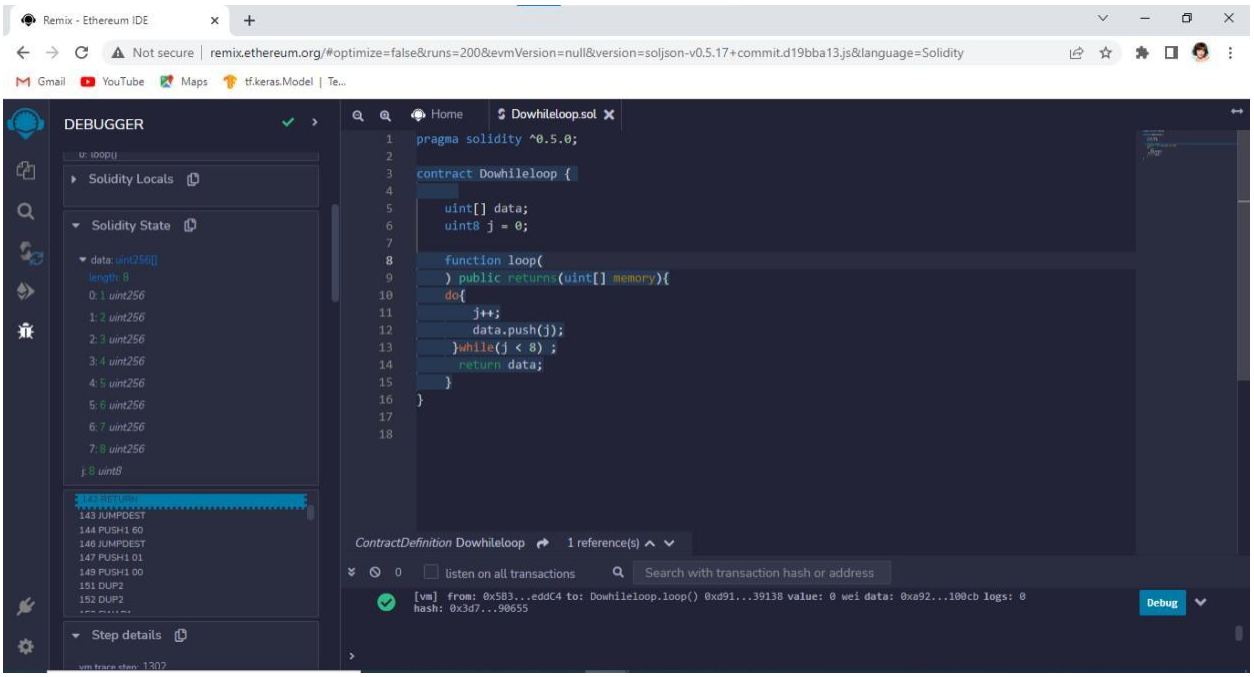


Dowhile loop

Code:

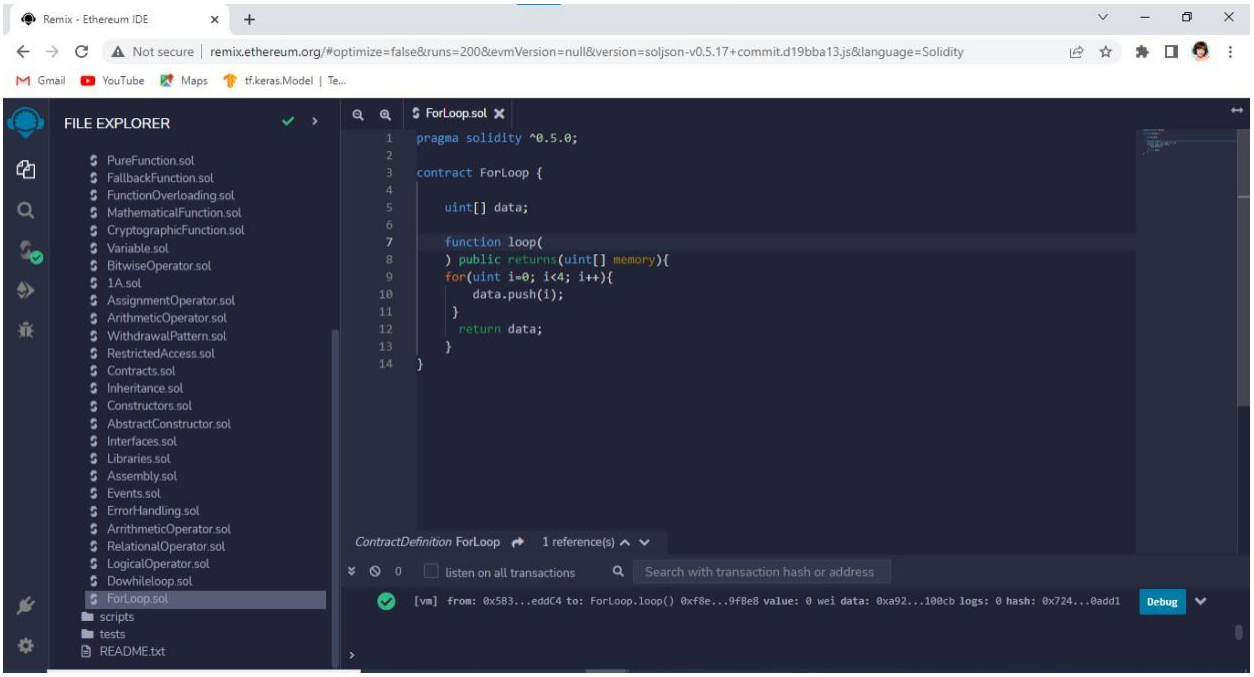


Output:

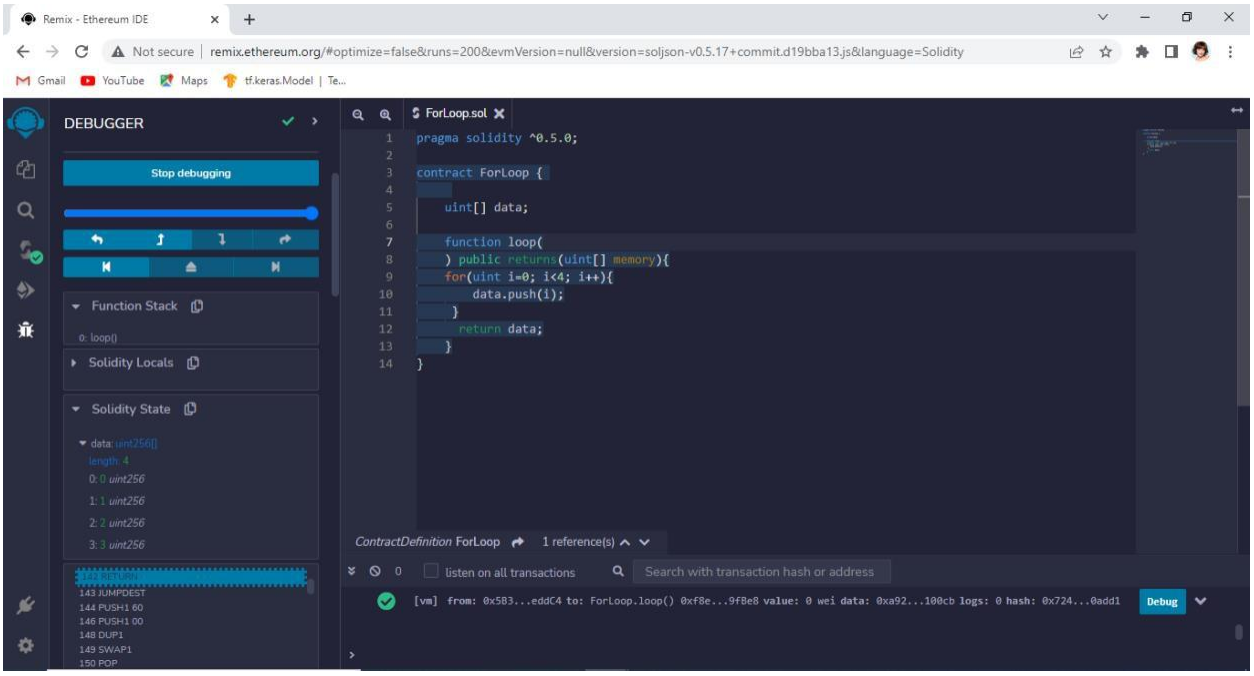


For loop

Code:



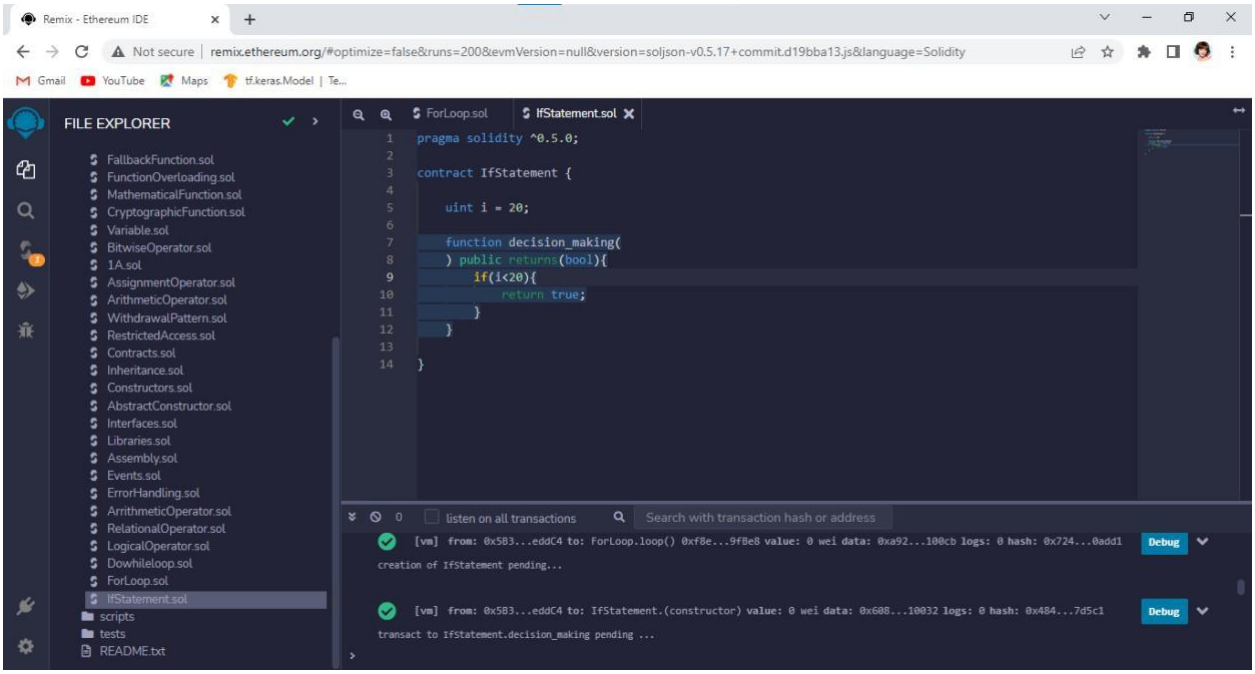
Output:



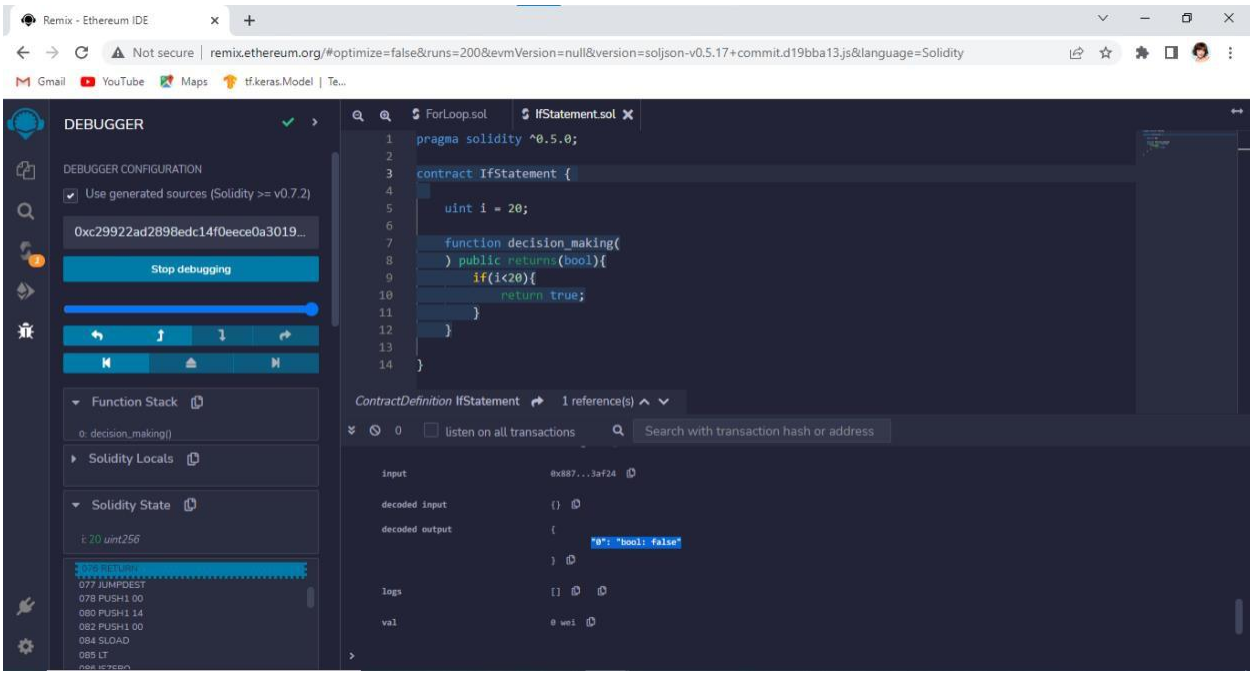
Decision making

If statement

Code:

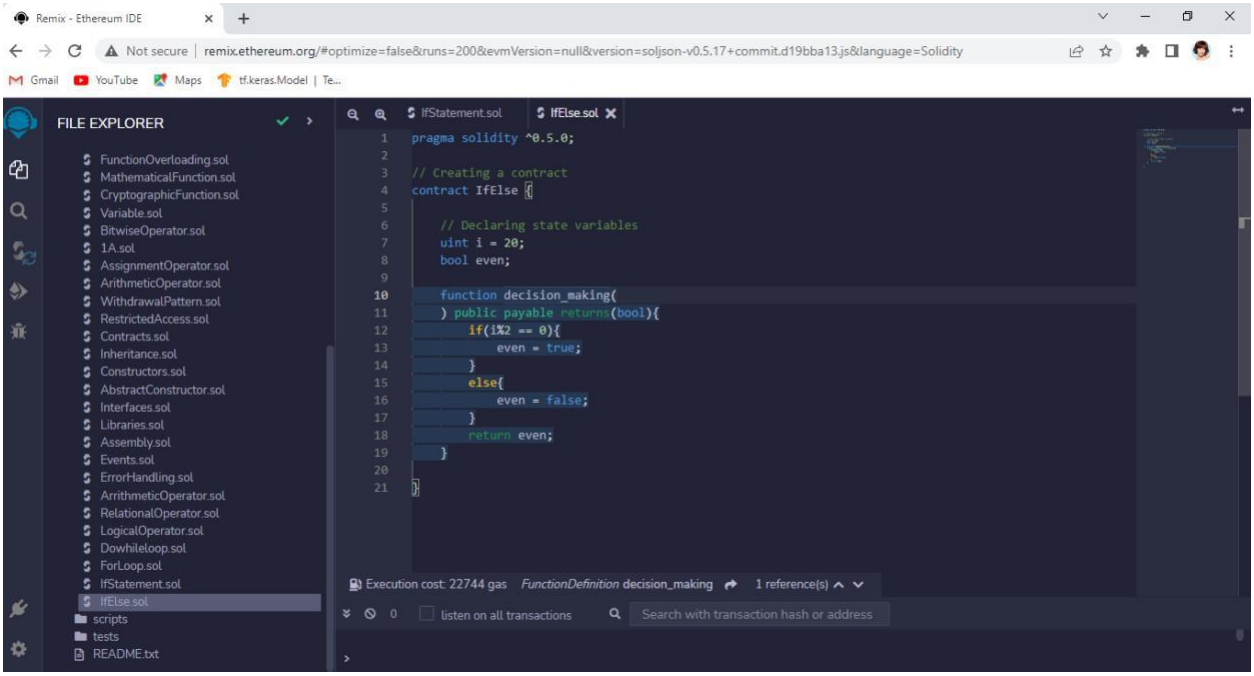


Output:

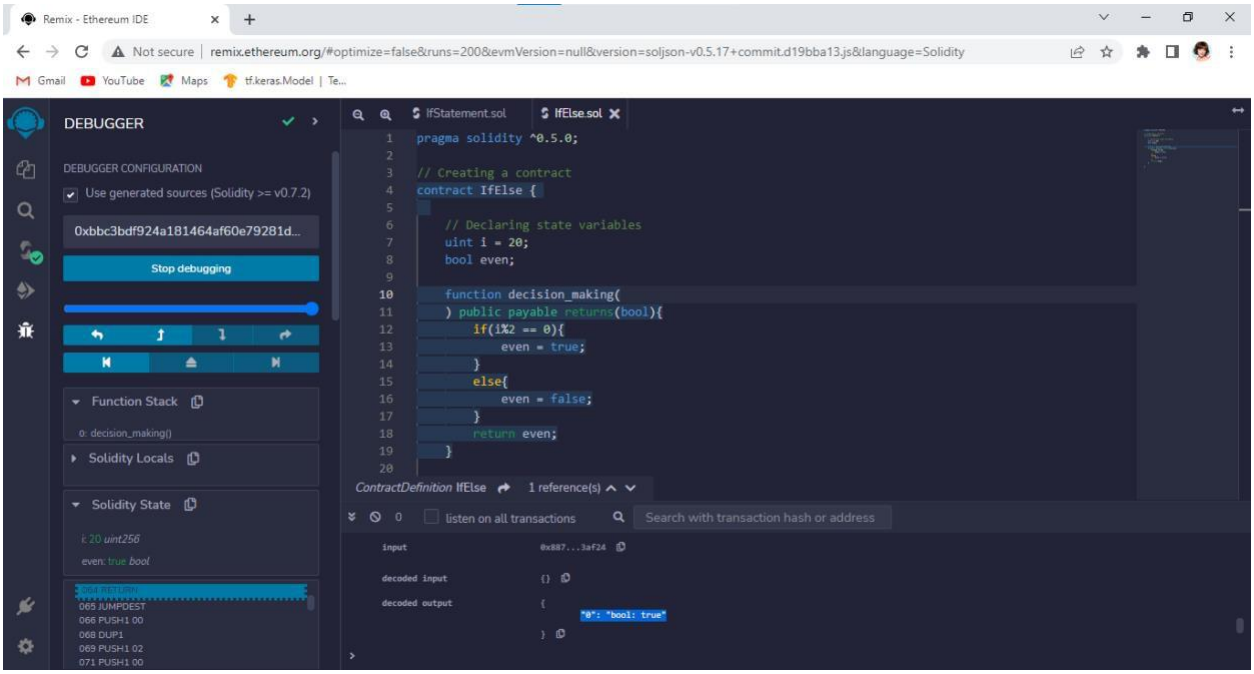


If…else statement

Code:

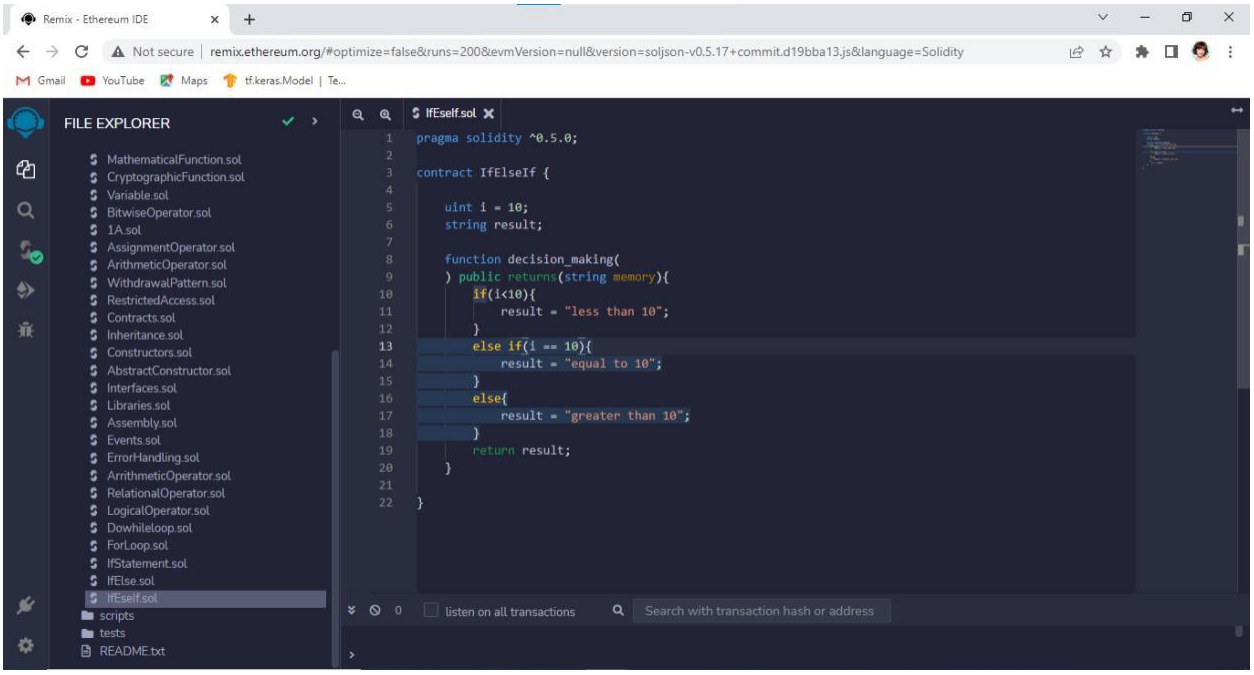


Output:

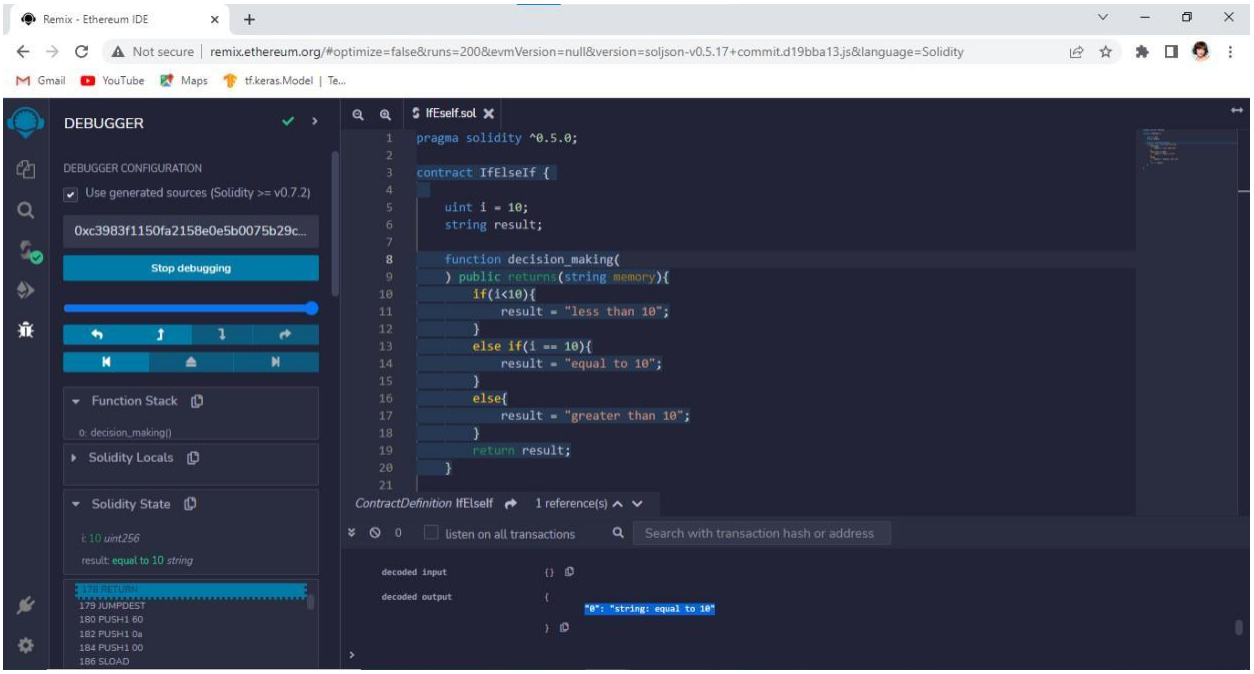


If…else if…else statement

Code:

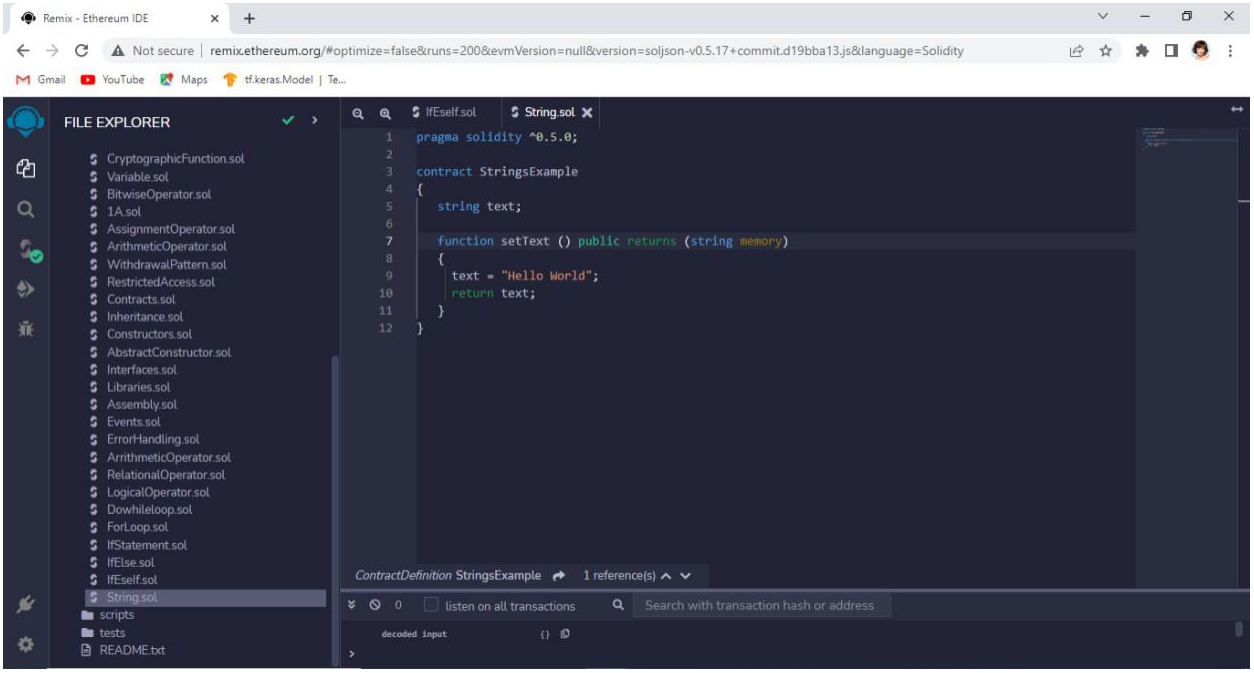


Output:

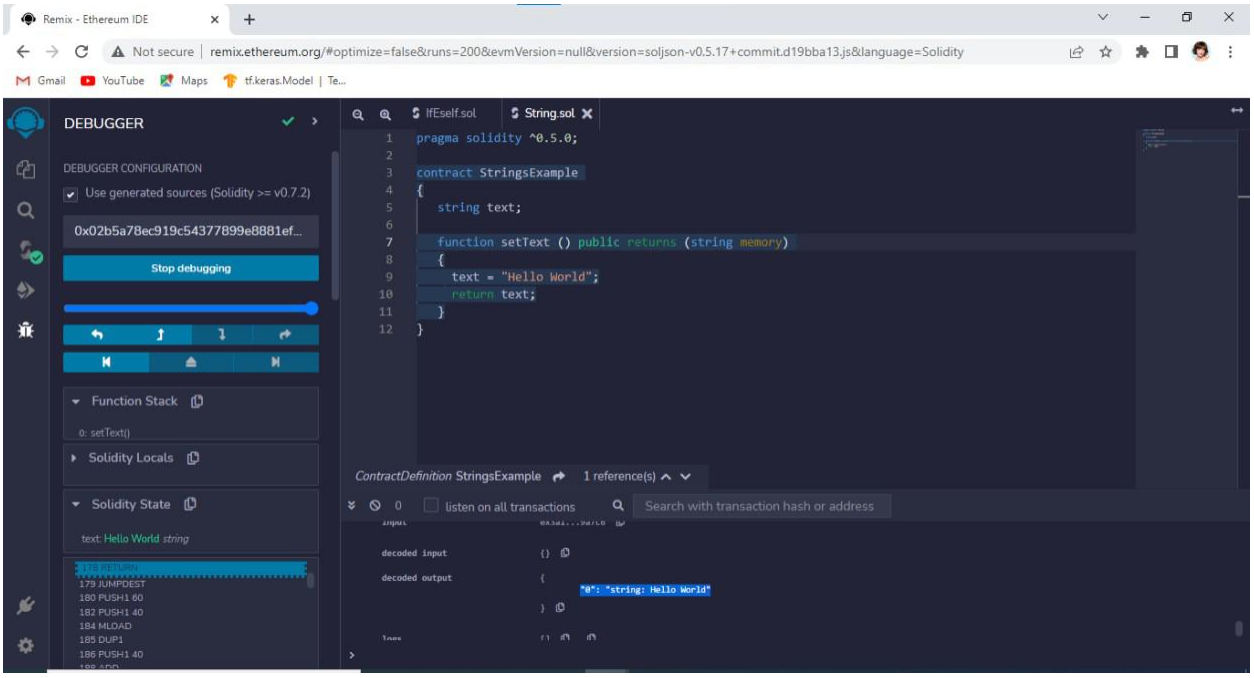


String

Code:

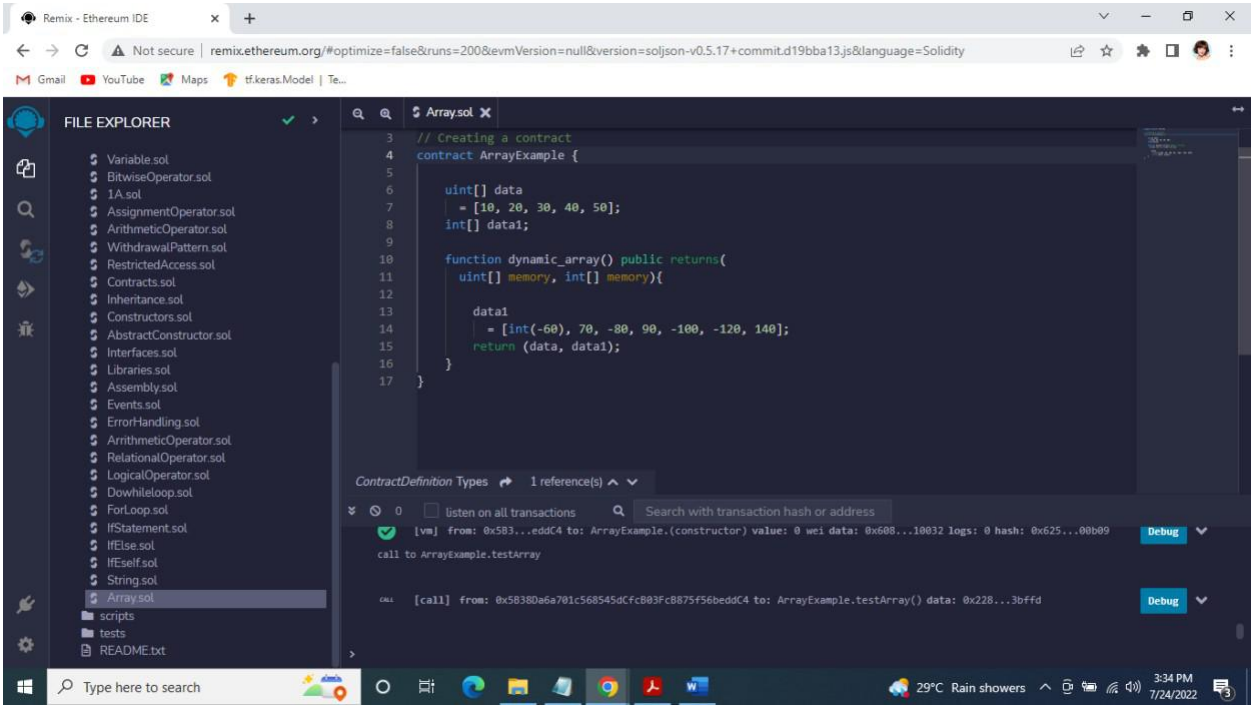


Output:

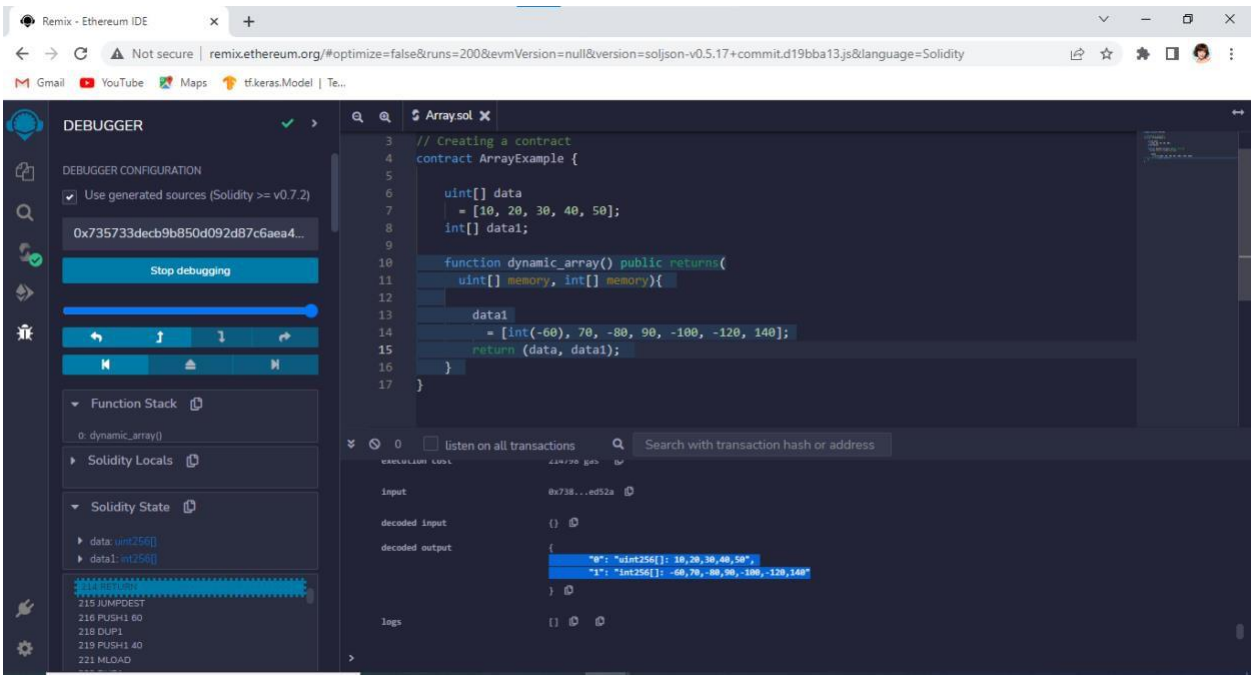


Array

Code:

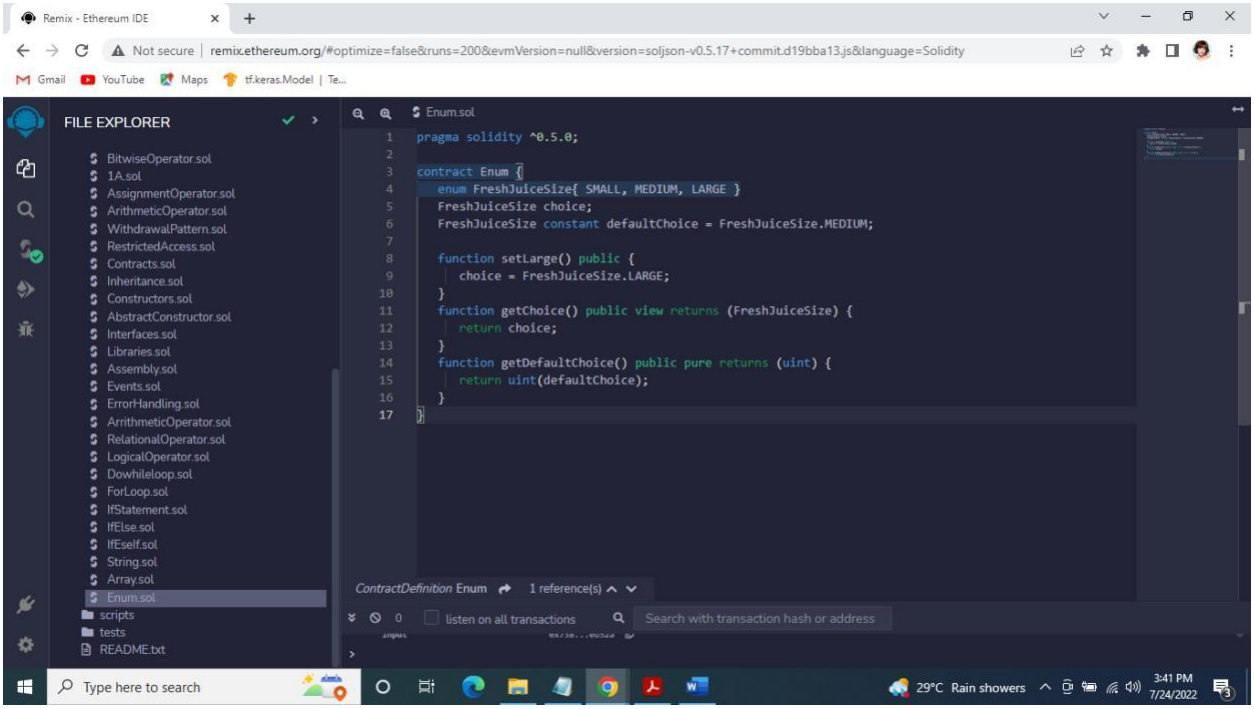


Output:

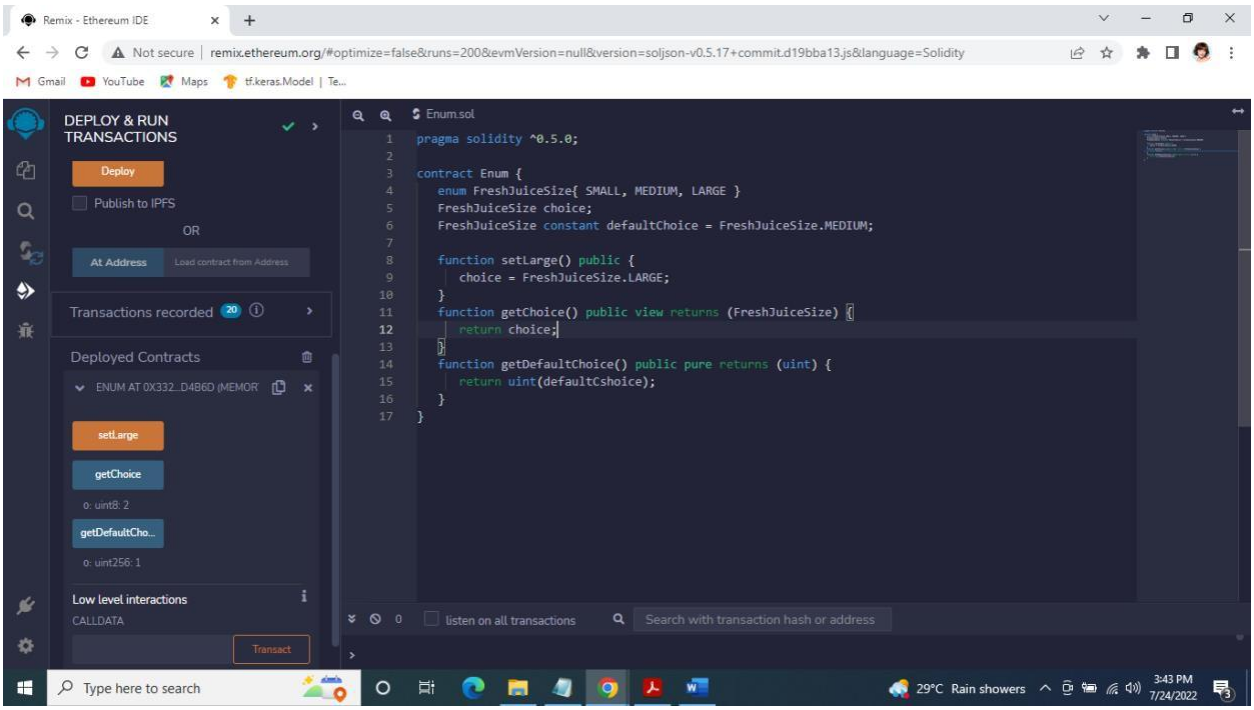


Enums

Code:

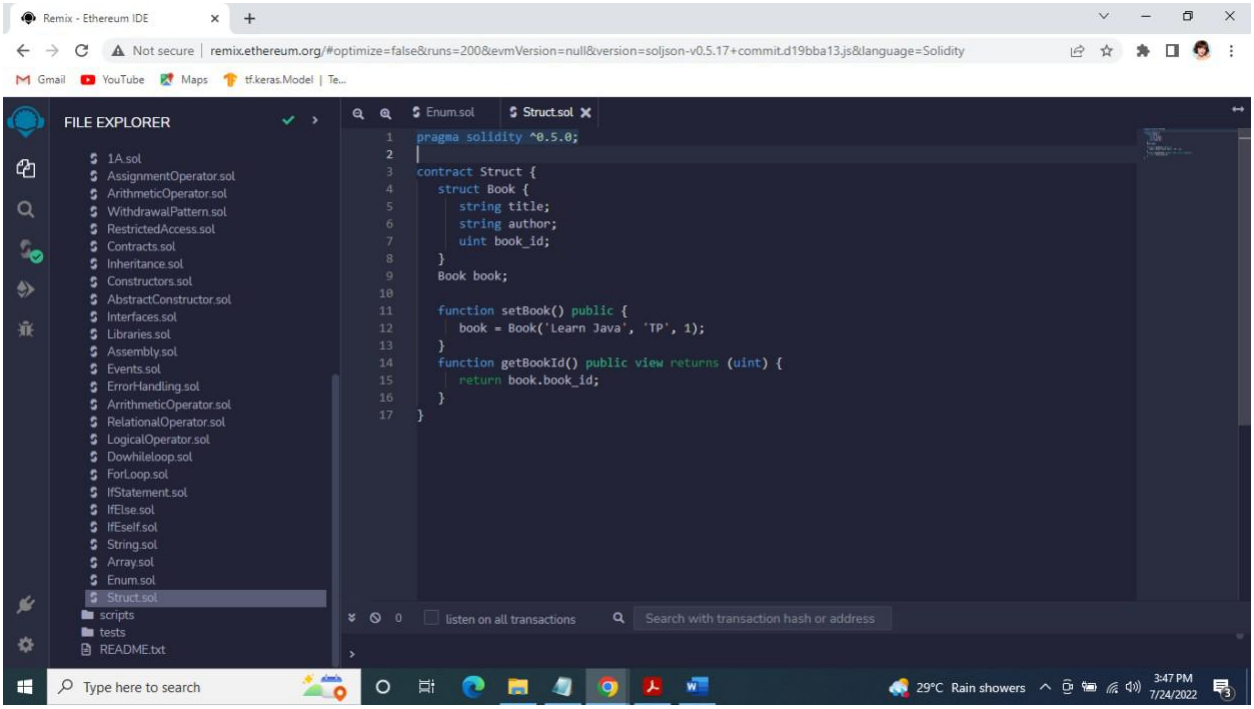


Output:

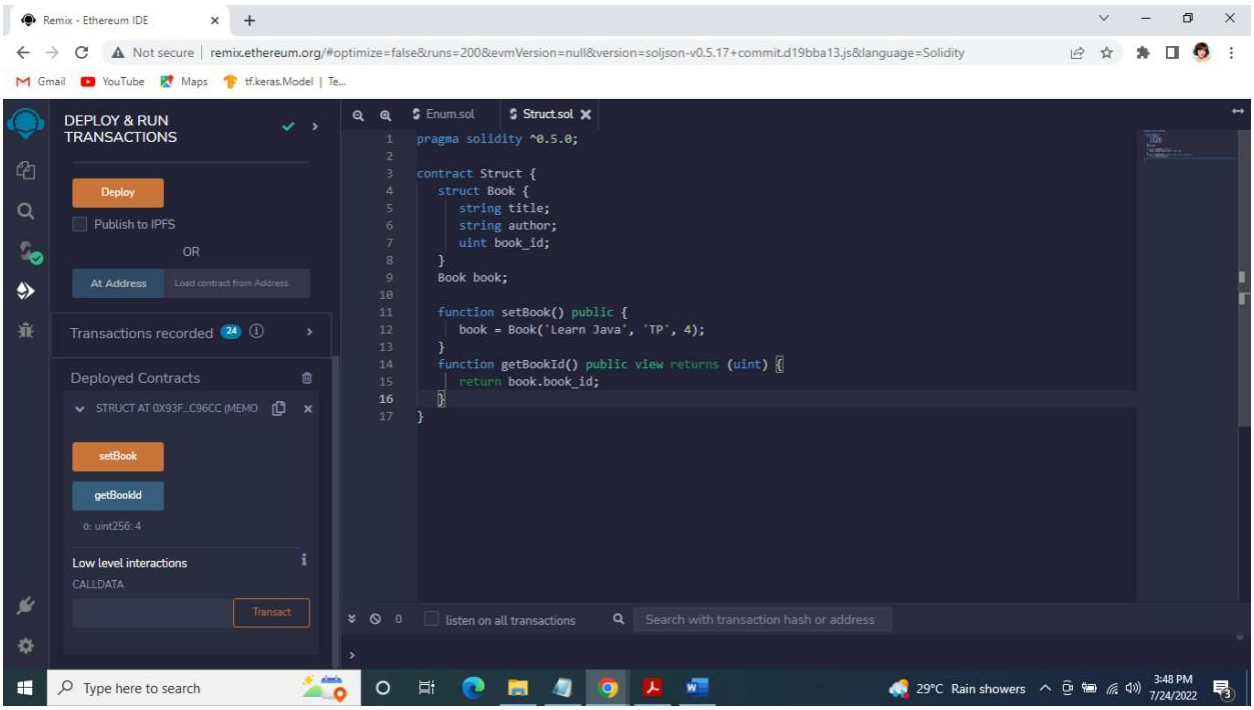


Struct

Code:

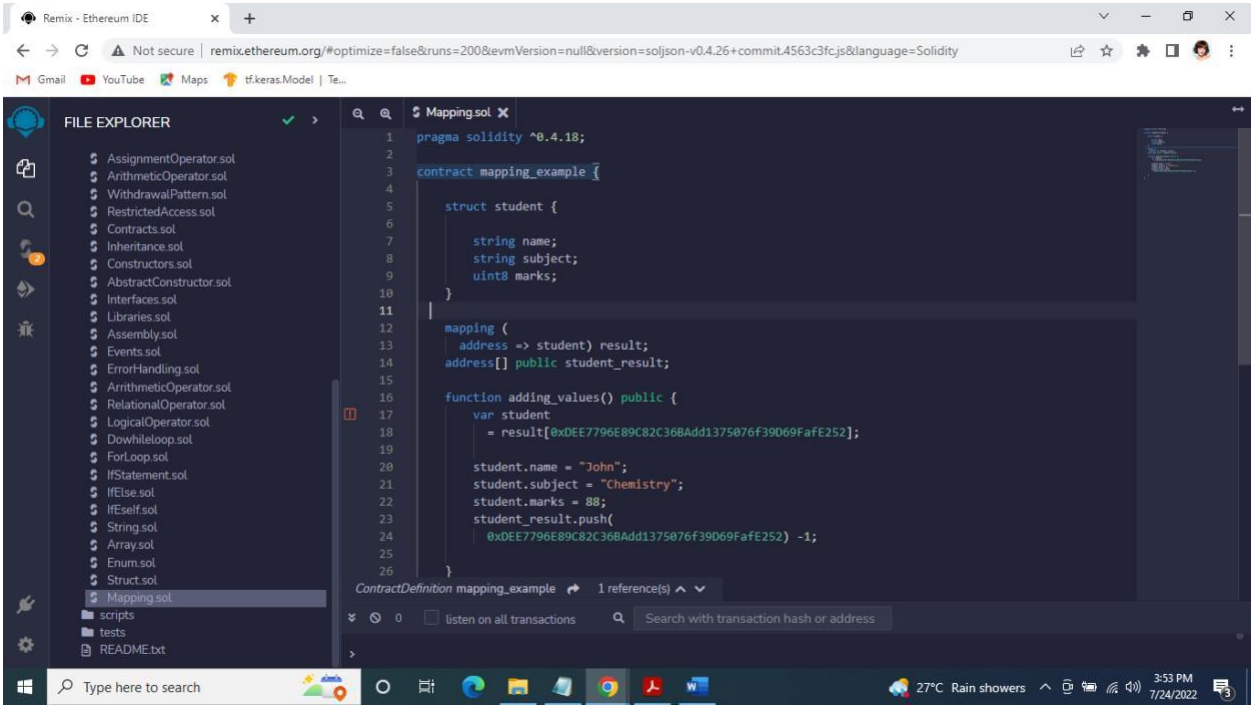


Output:

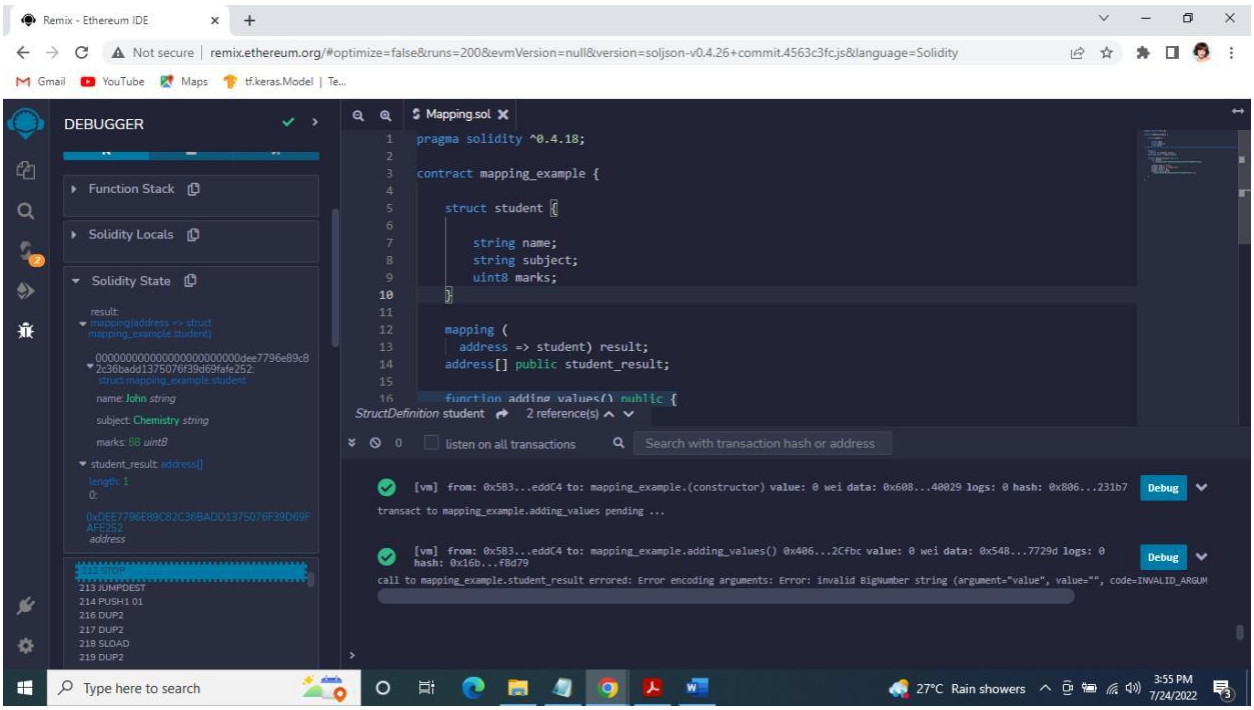


Mapping

Code:

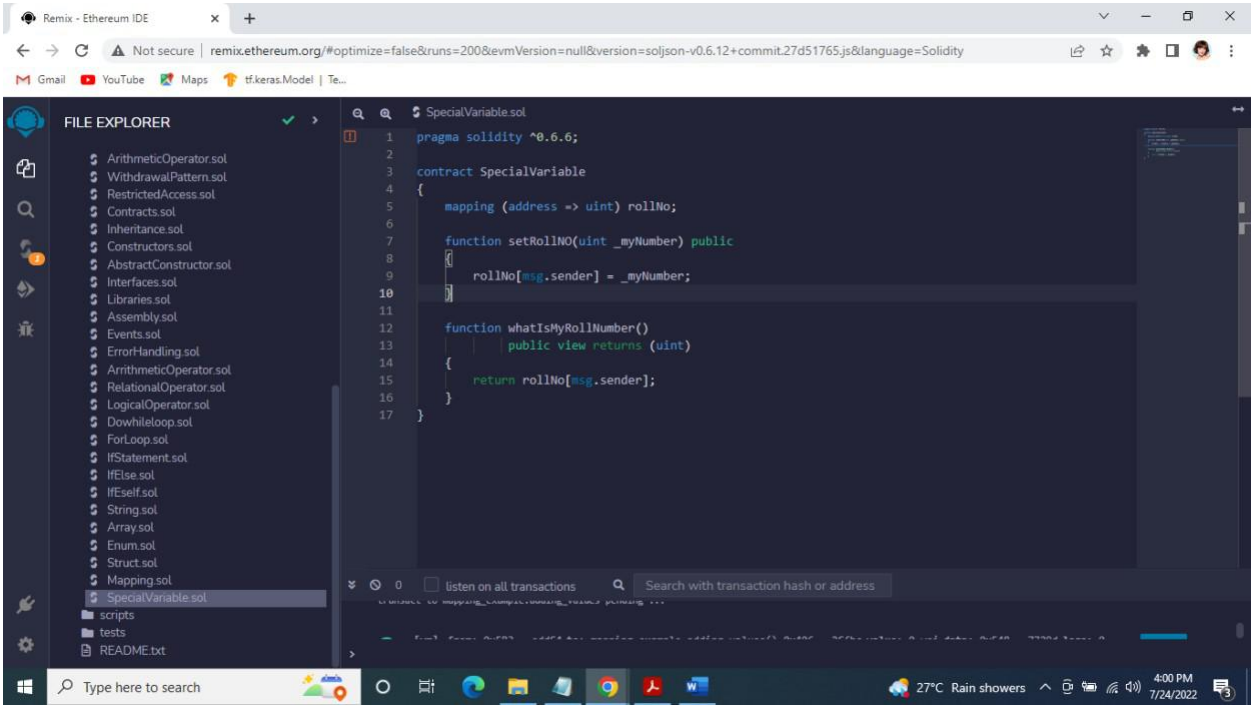


Output:

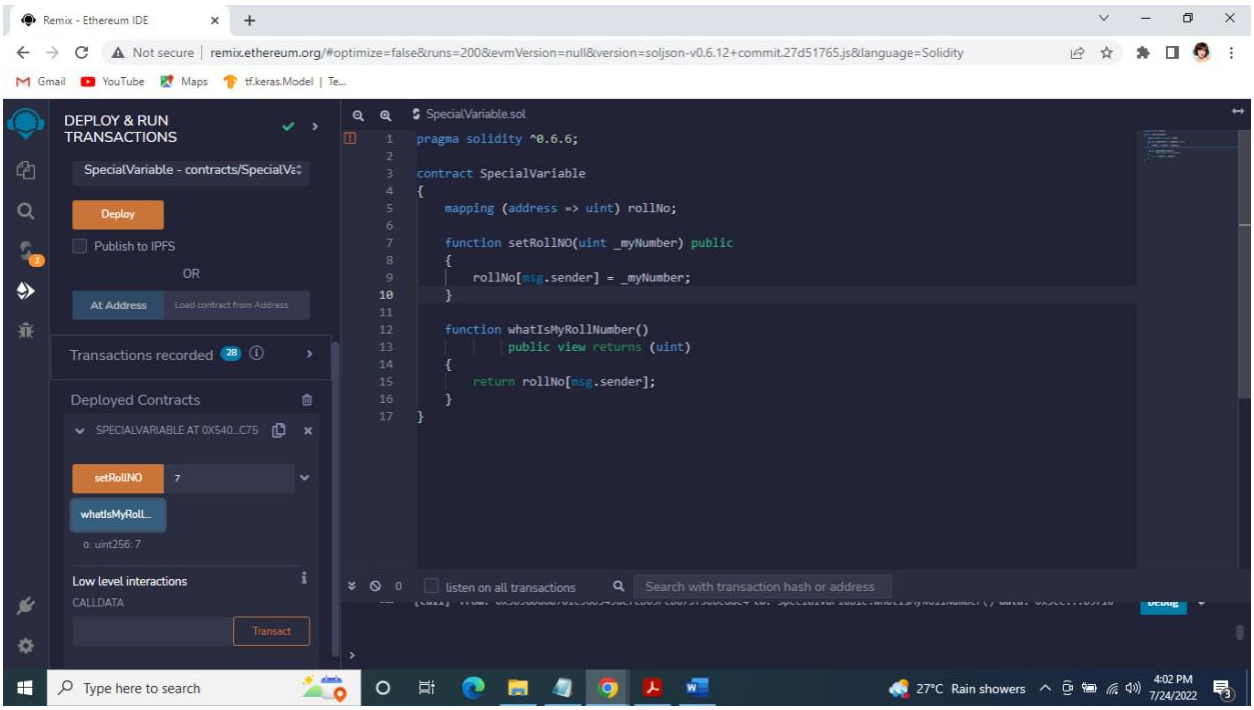


Special Variable

Code:



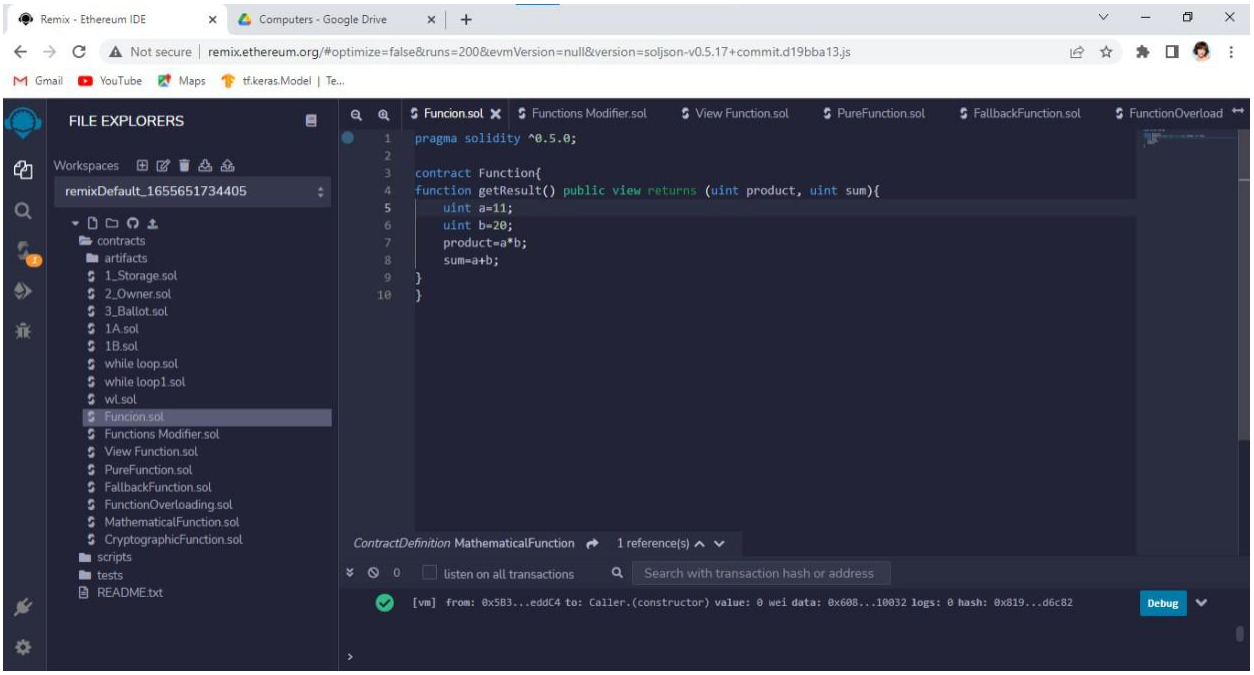
Output:



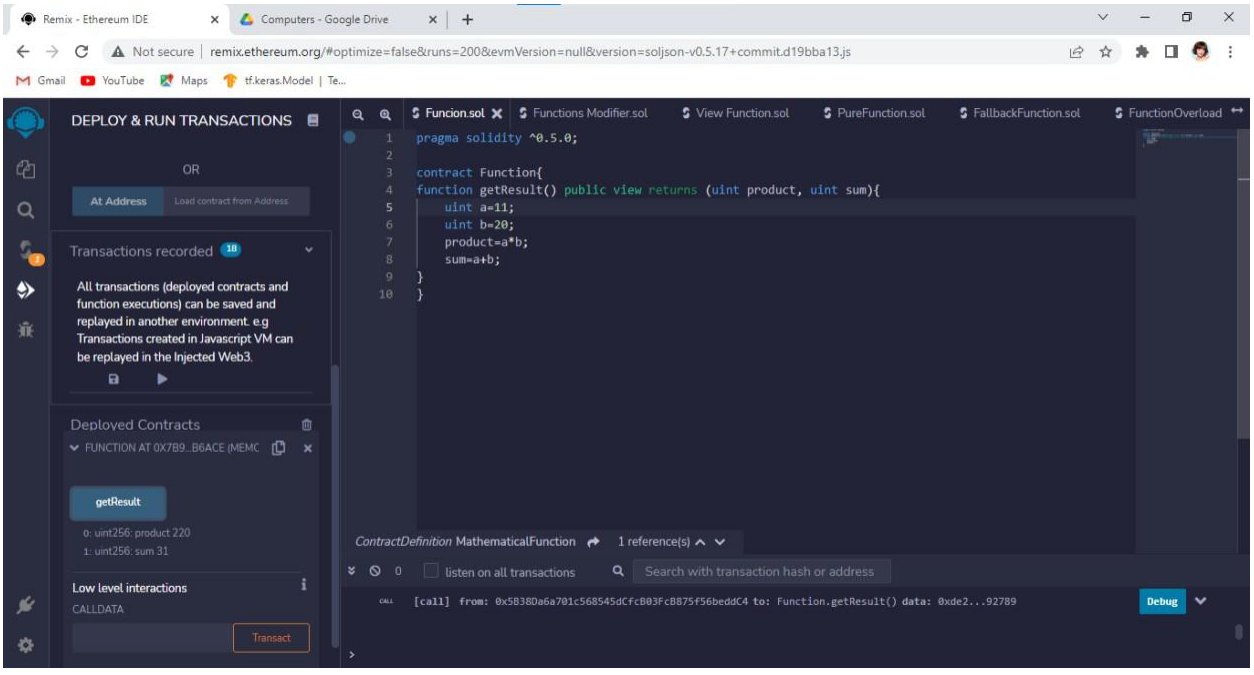
2b) Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

Function:

Code:

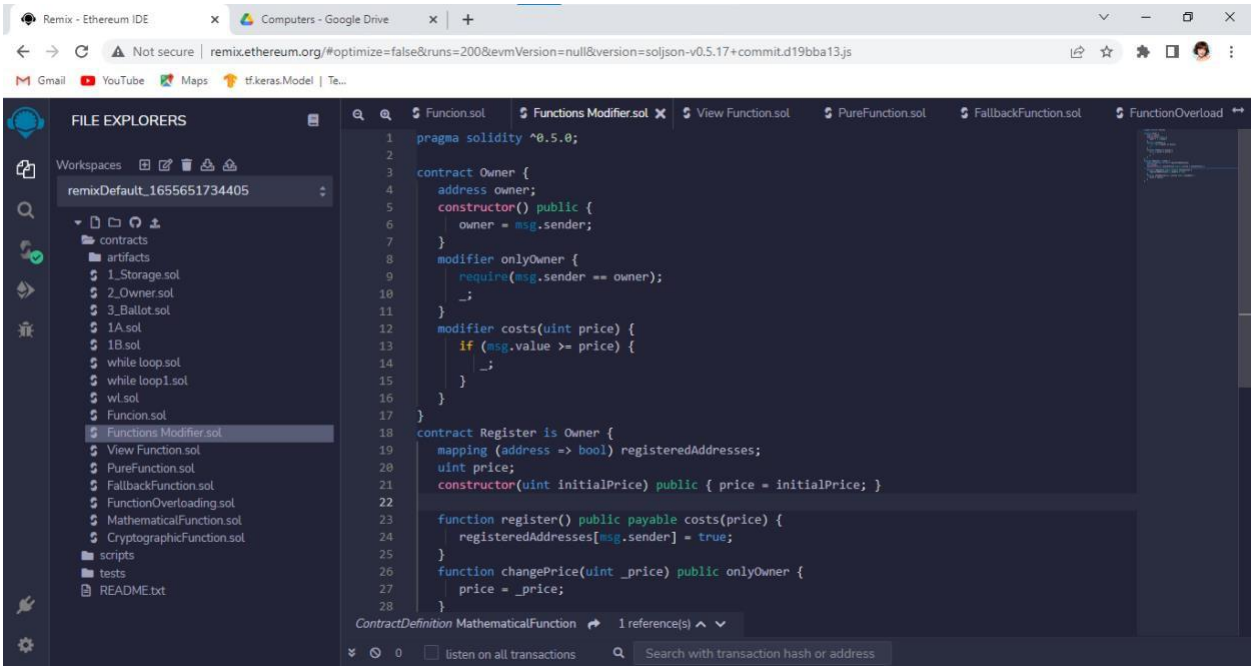


Output:

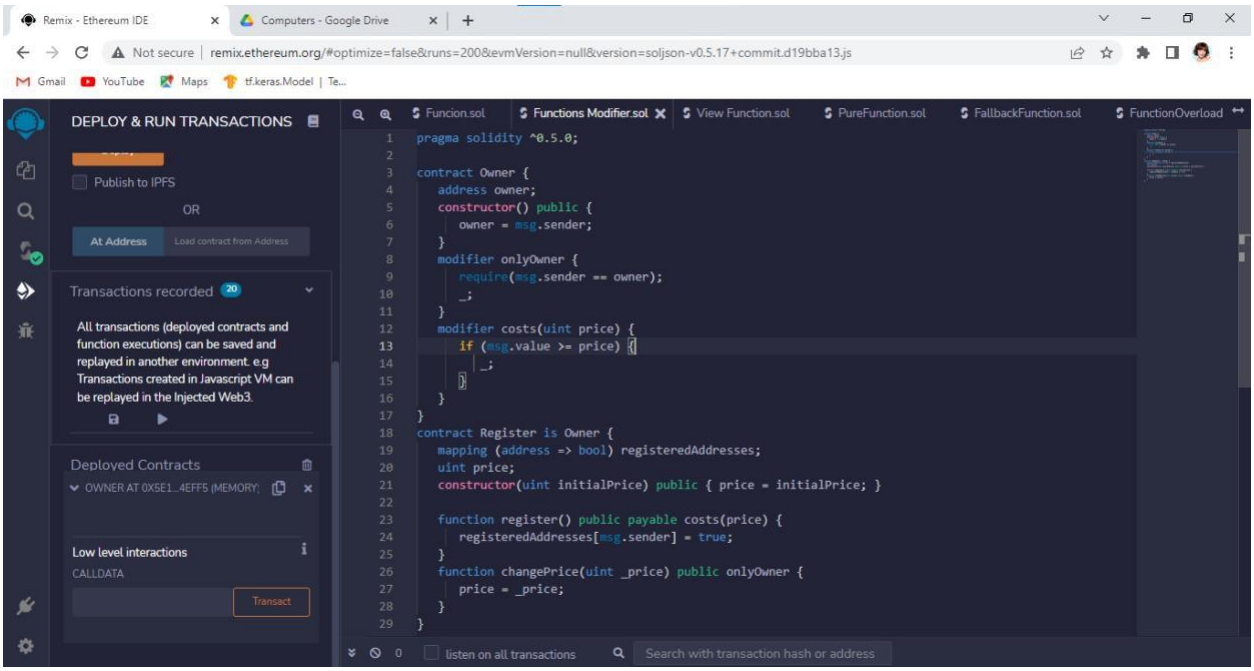


Functions Modifiers

Code:

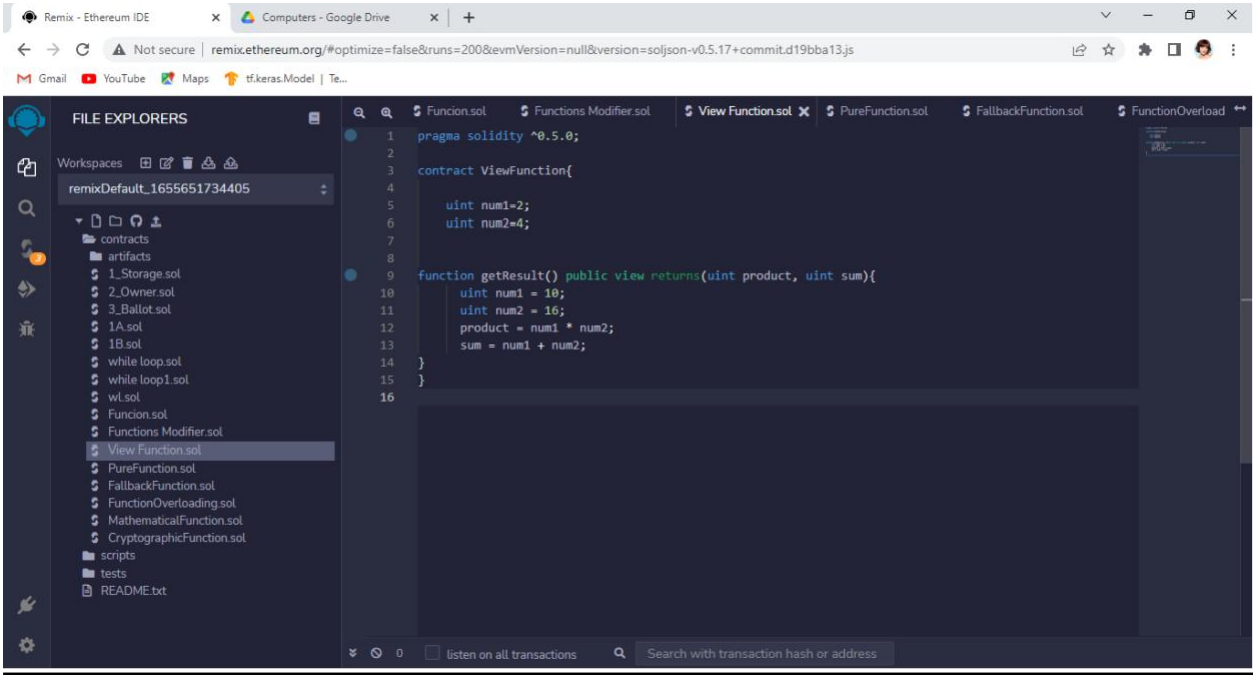


Output:

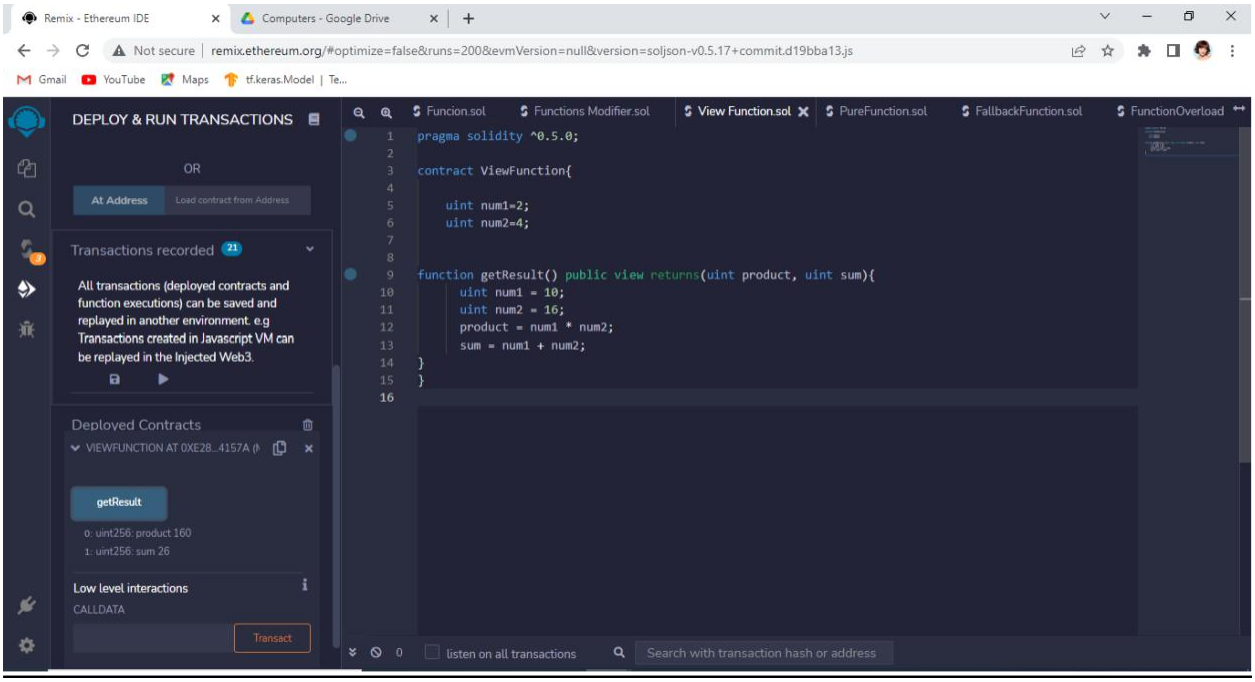


View function

Code:

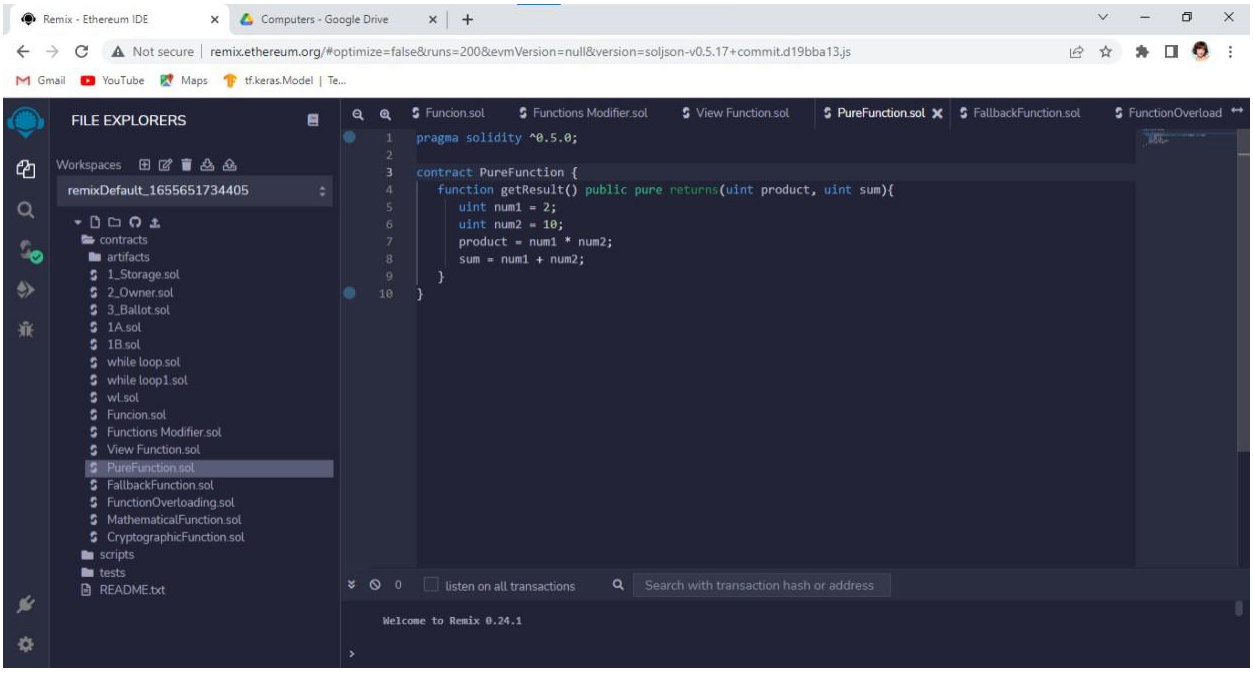


Output:

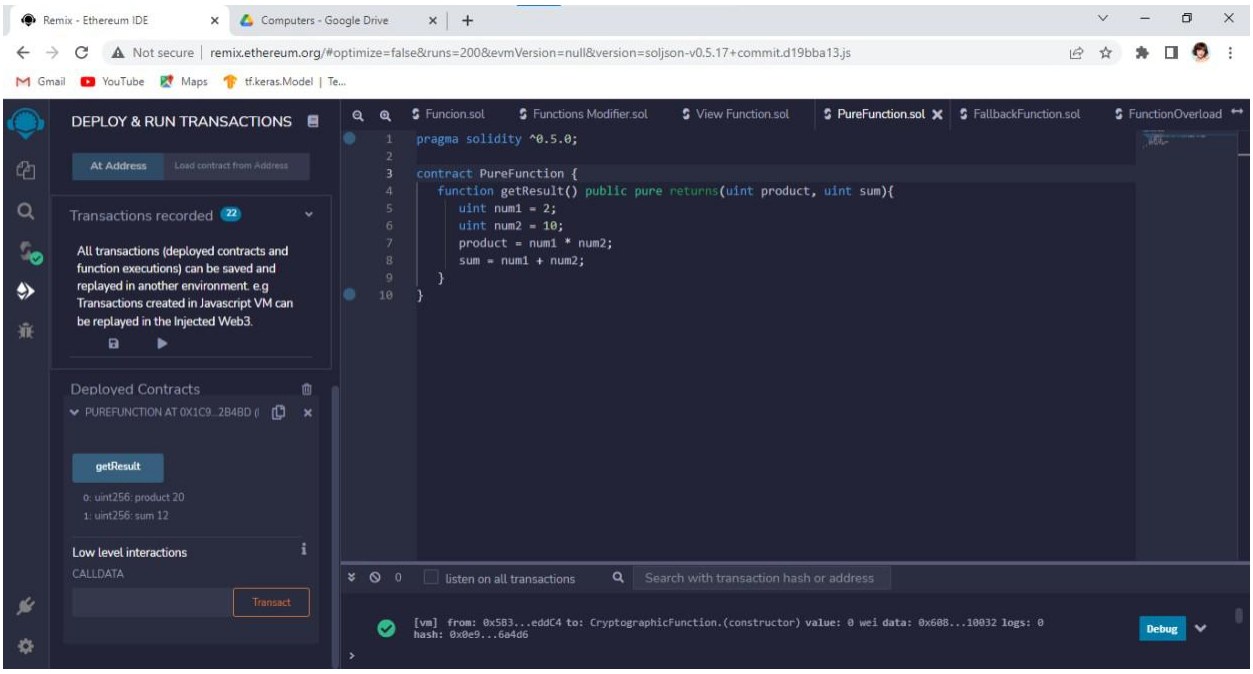


Pure function

Code:

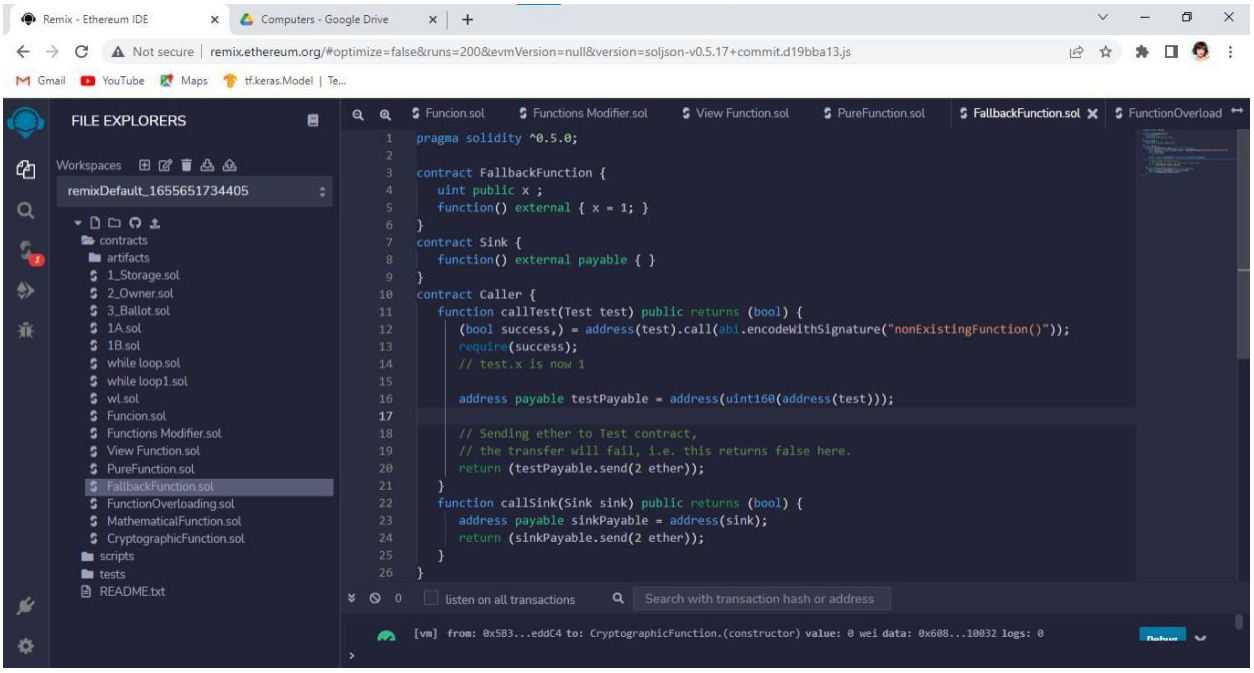


Output:

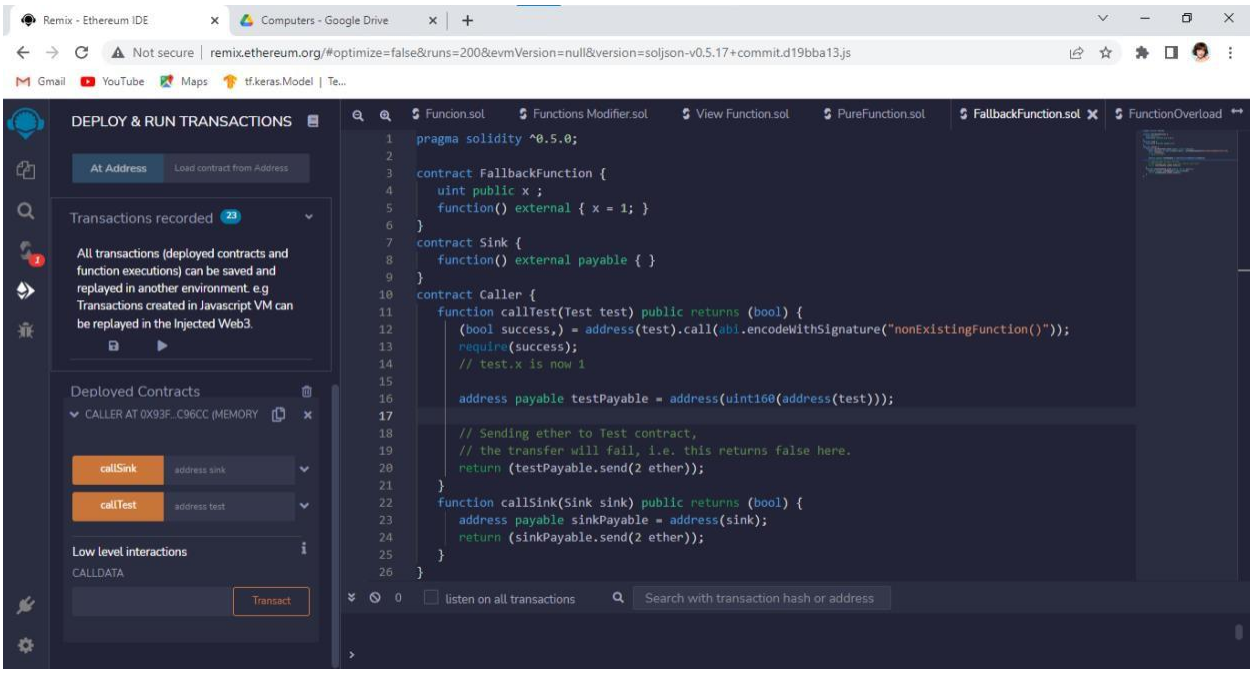


Fallback function

Code:

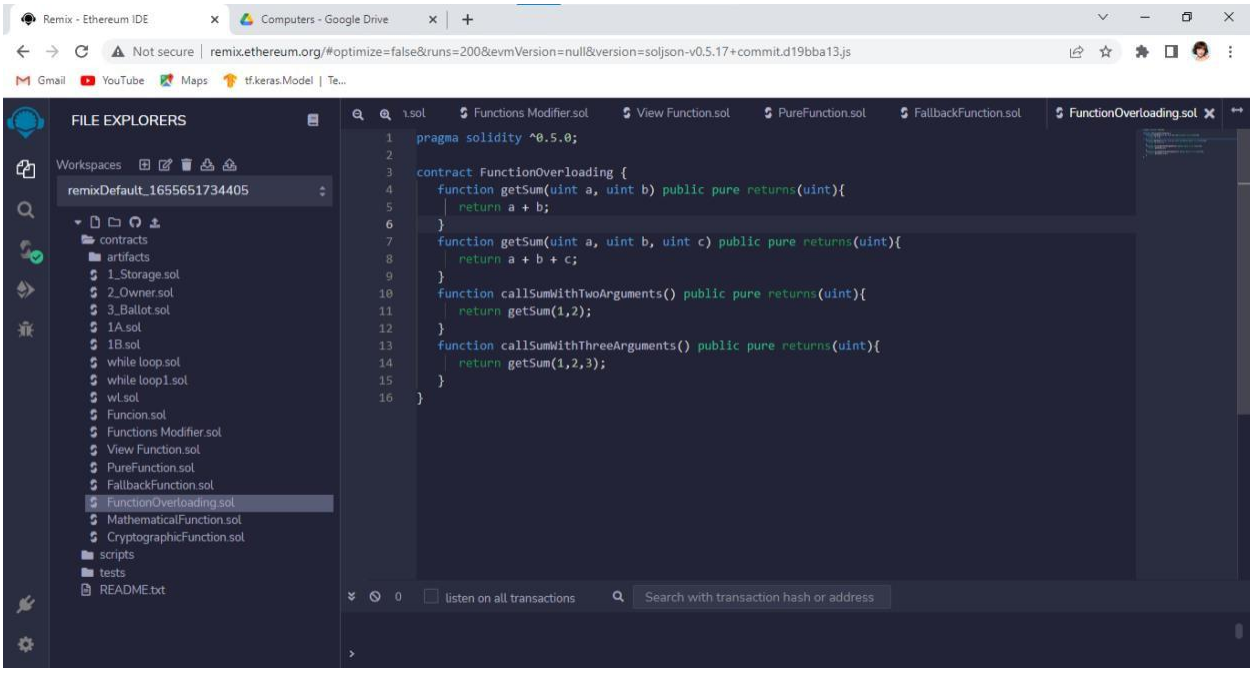


Output:

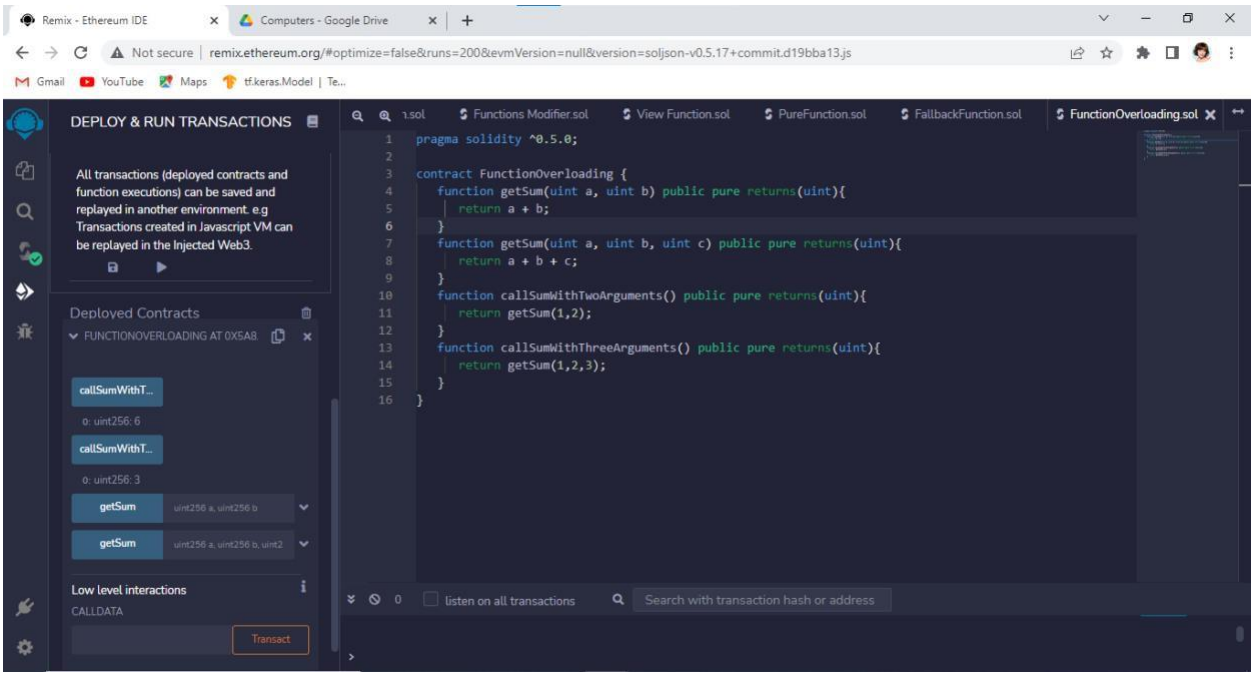


Function Overloading

Code:

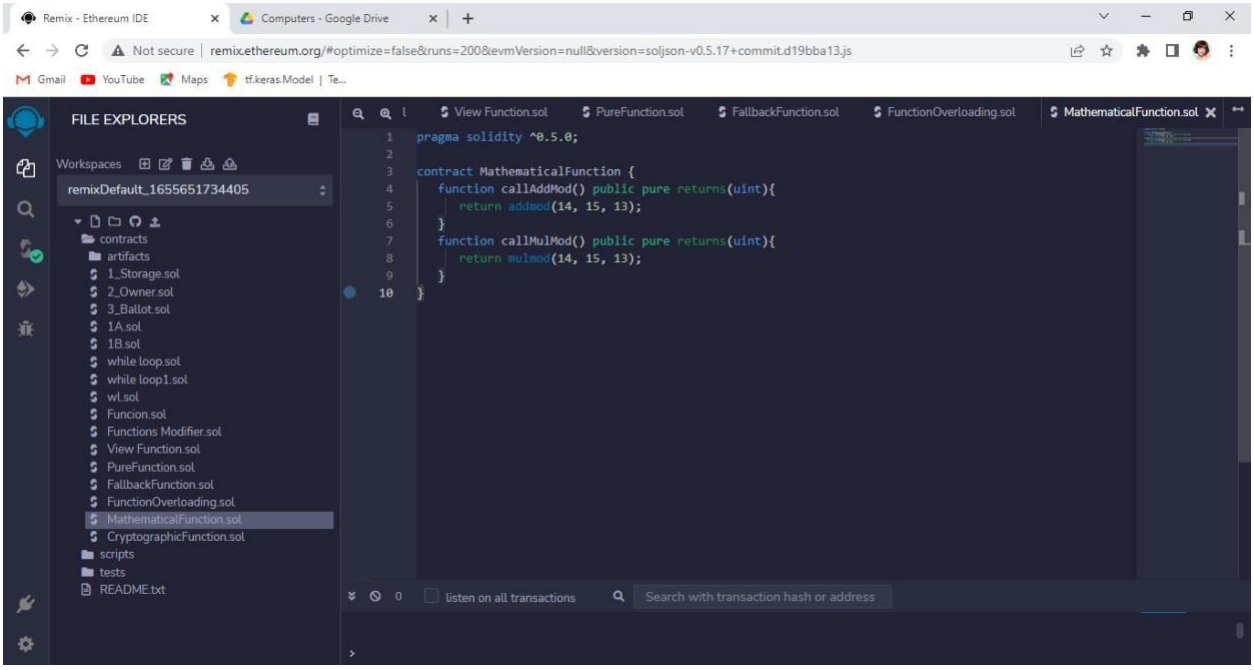


Output:

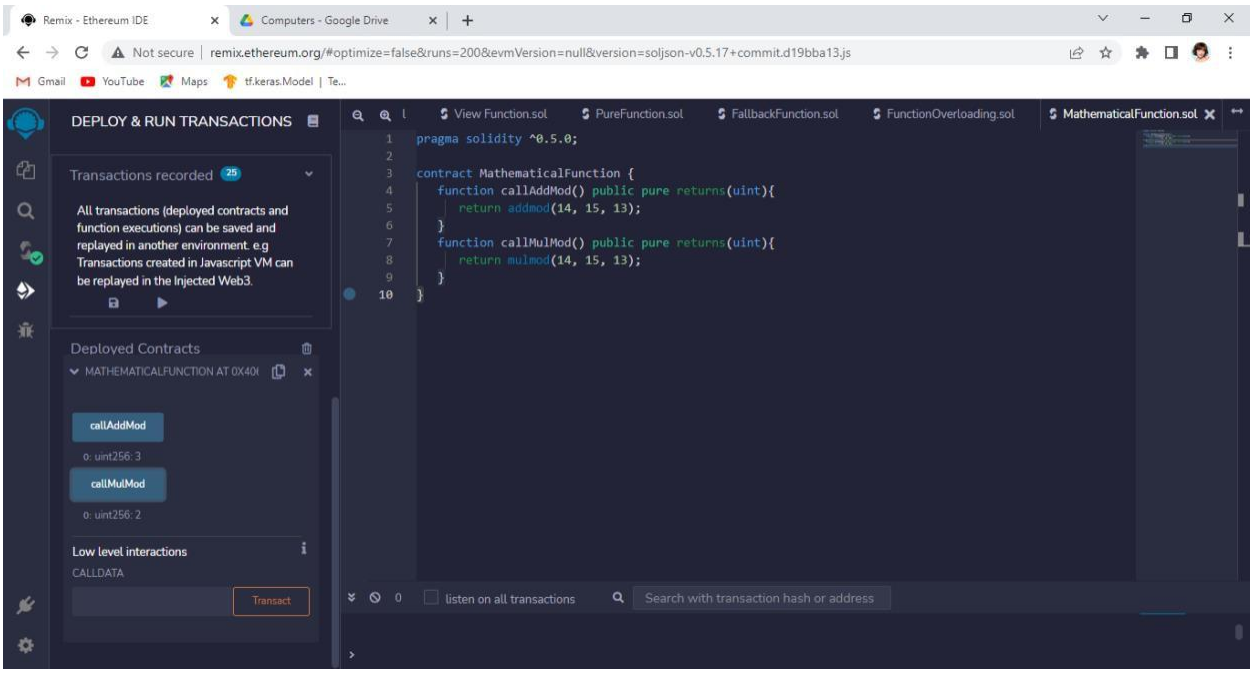


Mathematical Function

Code:

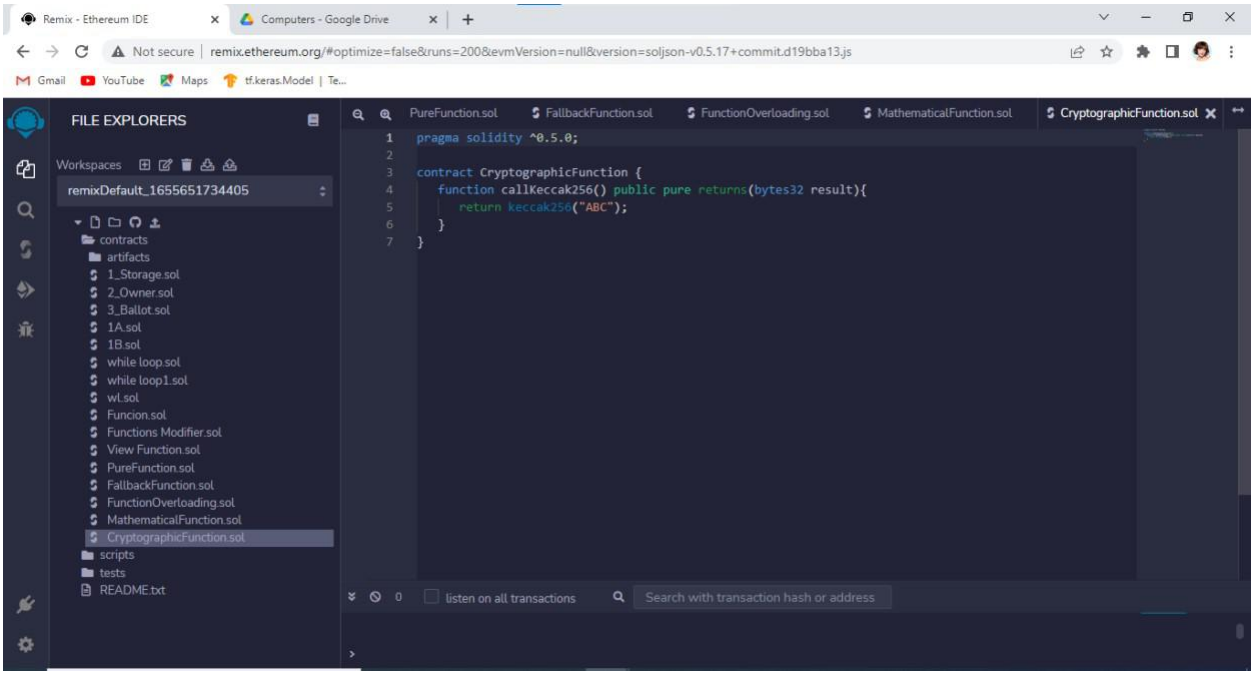


Output:

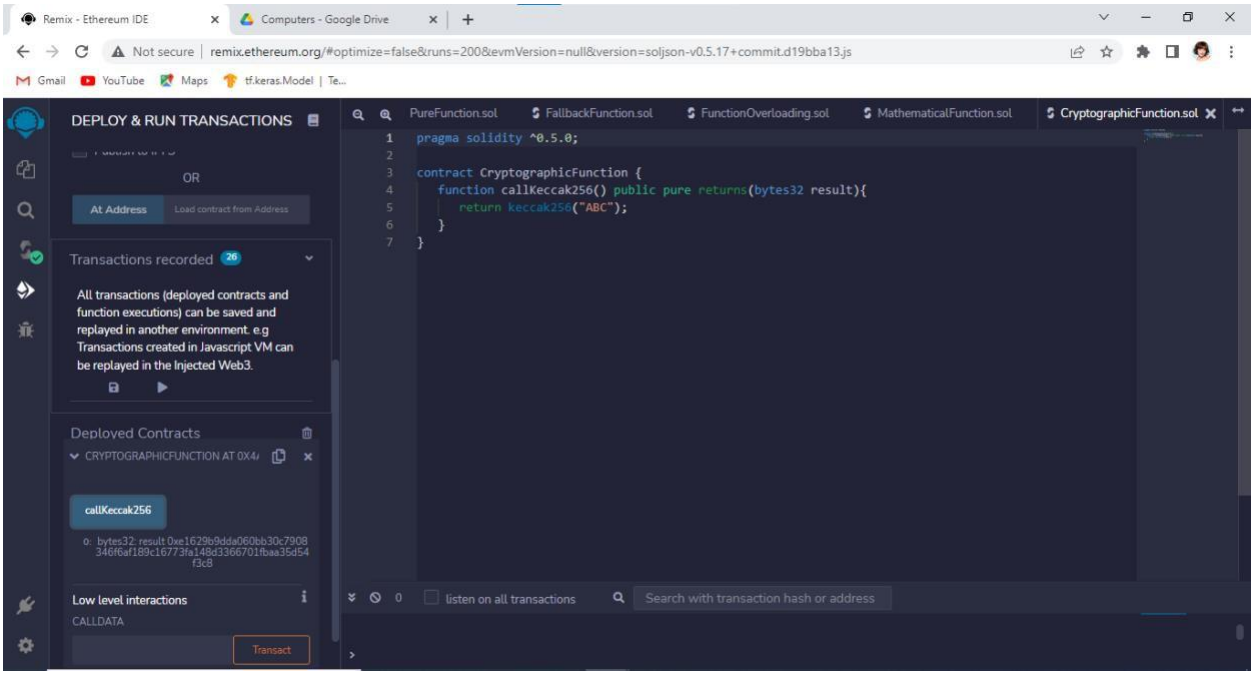


Cryptographic function

Code:



Output:



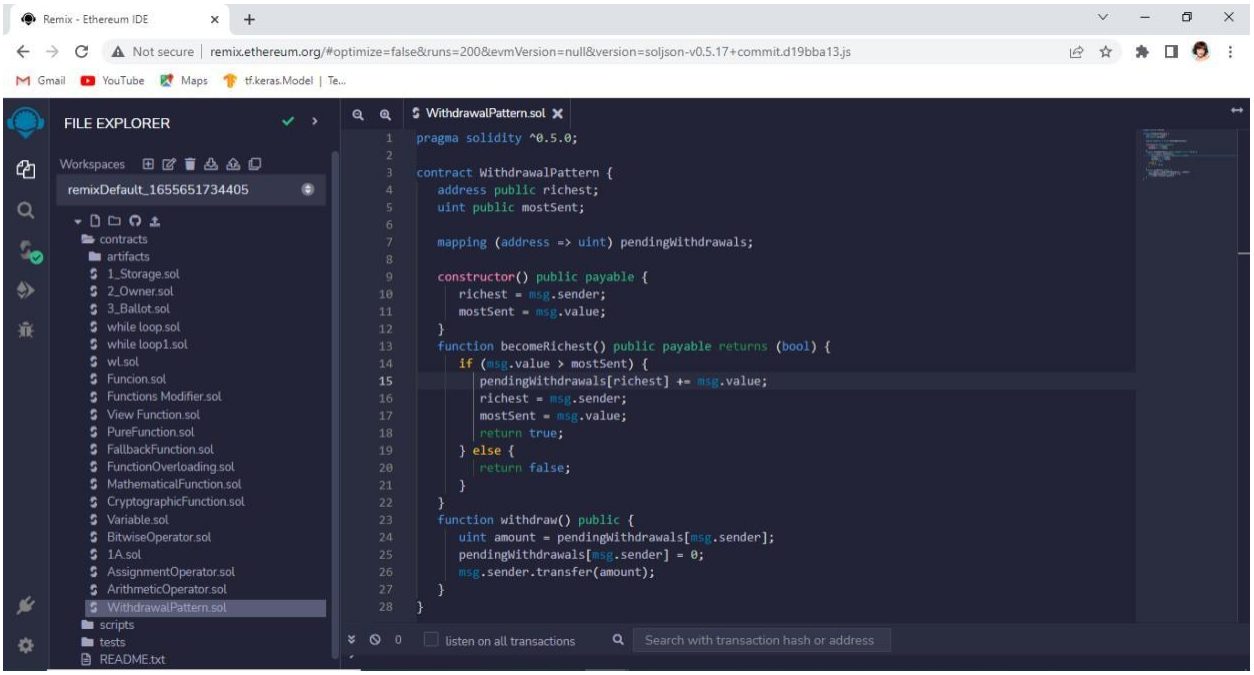
Practical No: 3

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

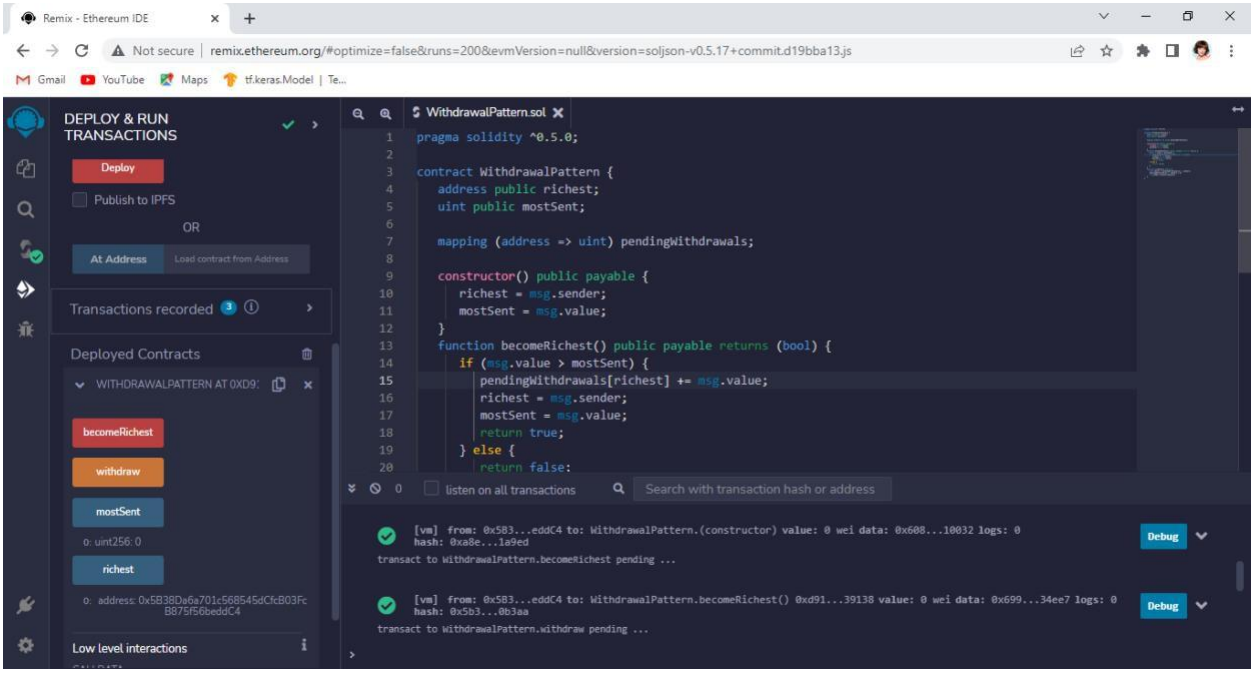
3a) Withdrawal Pattern, Restricted Access.

<u>Withdrawal Pattern</u>

Code:

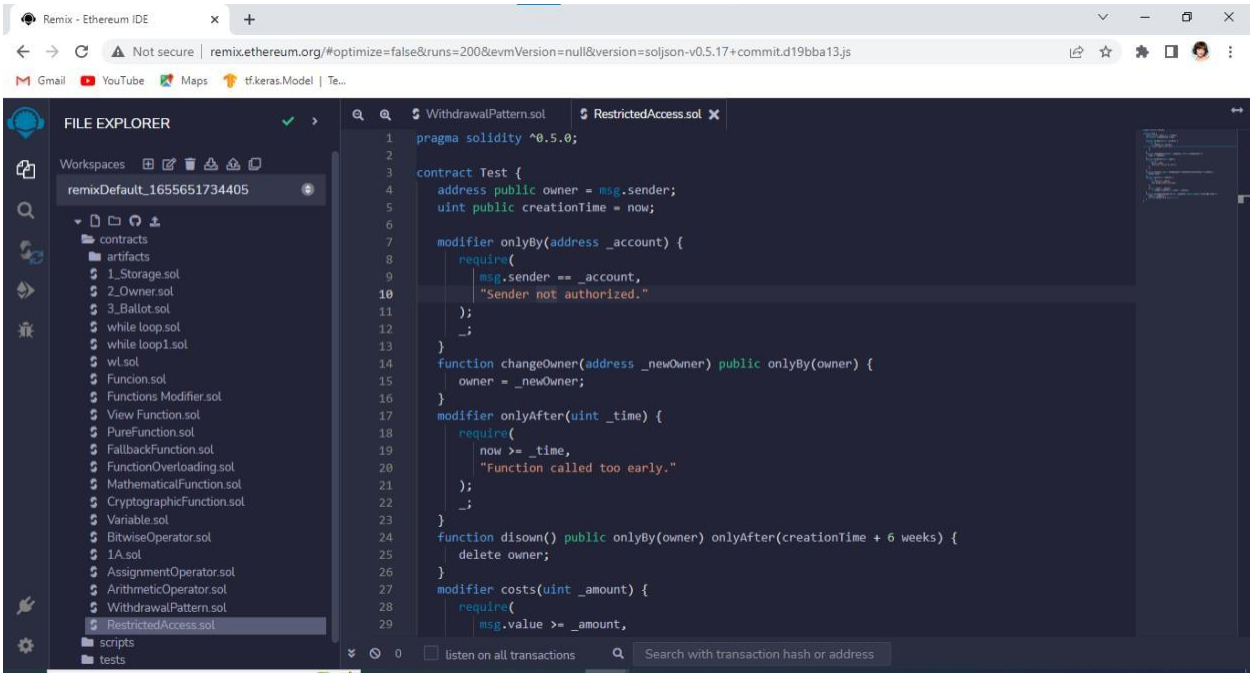


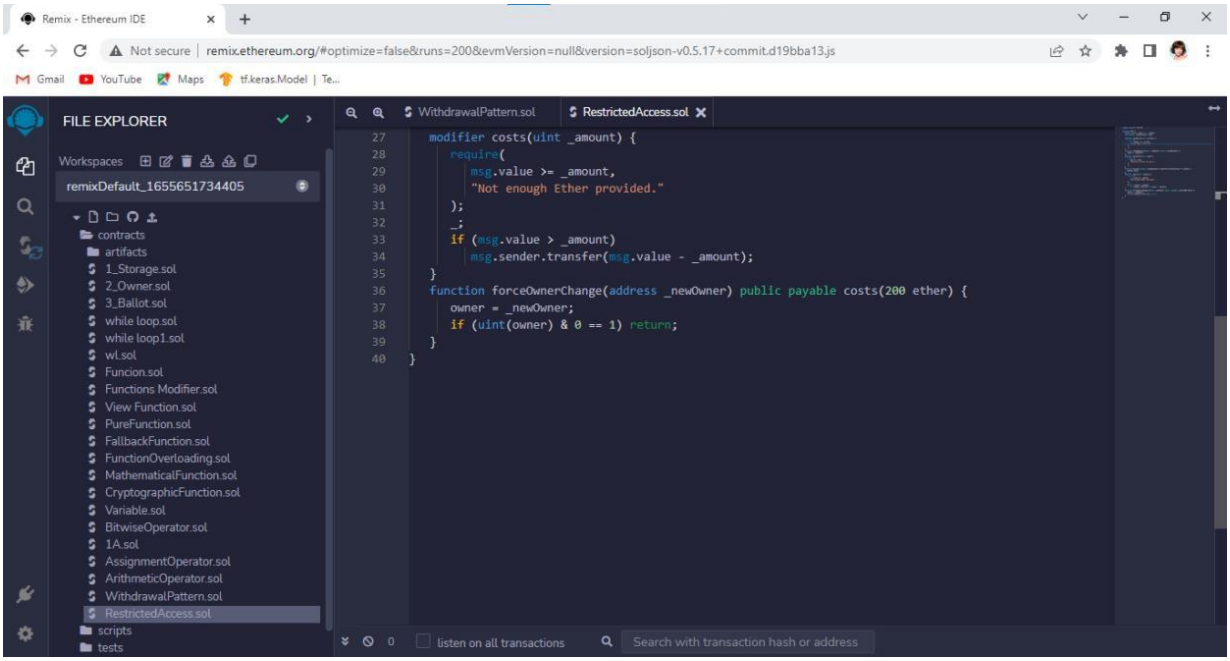
Output:


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
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
WITHDRAWALPATTERN AT 0XD9:
becomeRichest
withdraw
mostSent
0: uint256: 0
richest
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
Low level interactions
WithdrawalPattern.sol
pragma solidity ^0.5.0;
contract WithdrawalPattern {
address public richest;
uint public mostSent;
mapping (address => uint) pendingWithdrawals;
constructor() public payable {
richest = msg.sender;
mostSent = msg.value;
}
function becomeRichest() public payable returns (bool) {
if (msg.value > mostSent) {
pendingWithdrawals[richest] += msg.value;
richest = msg.sender;
mostSent = msg.value;
return true;
} else {
return false;
listen on all transactions
Search with transaction hash or address
[vm] from: 0x5B3...eddC4 to: WithdrawalPattern.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0xa8e...1a9ed
transact to WithdrawalPattern.becomeRichest pending ...
Debug
[vm] from: 0x5B3...eddC4 to: WithdrawalPattern.becomeRichest() 0xd91...39138 value: 0 wei data: 0x699...34ee7 logs: 0
hash: 0x5b3...0b3aa
transact to WithdrawalPattern.withdraw pending ...
Debug

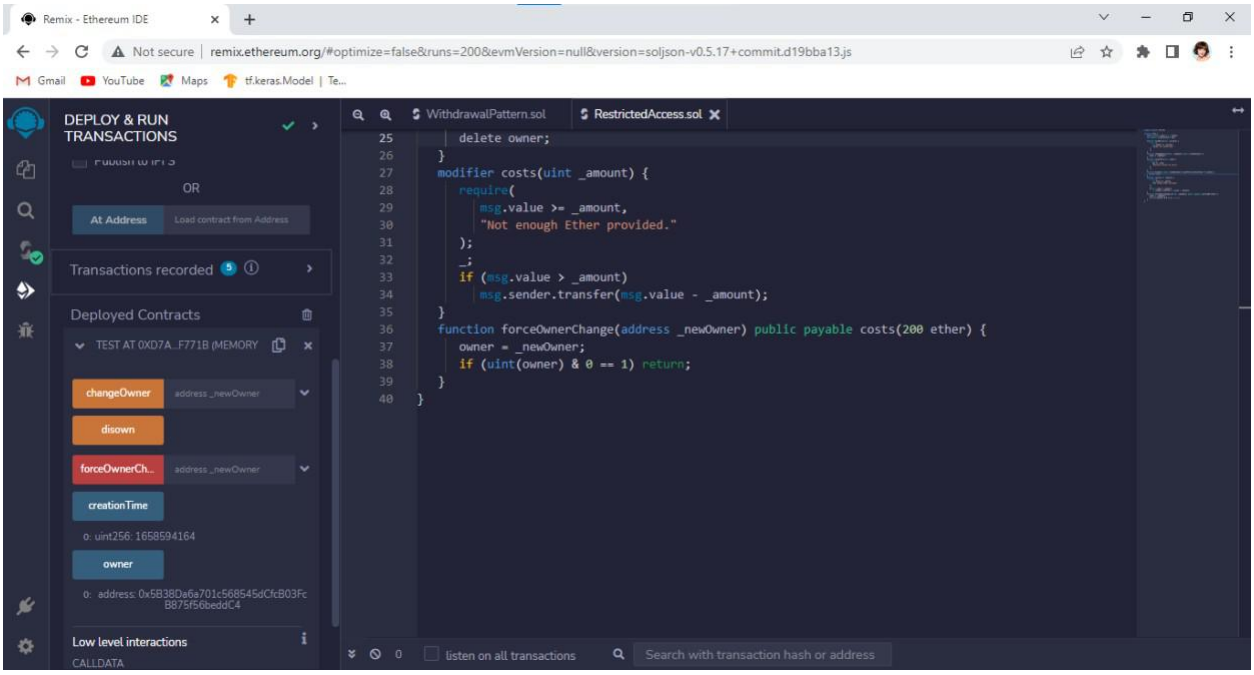
Restricted Access

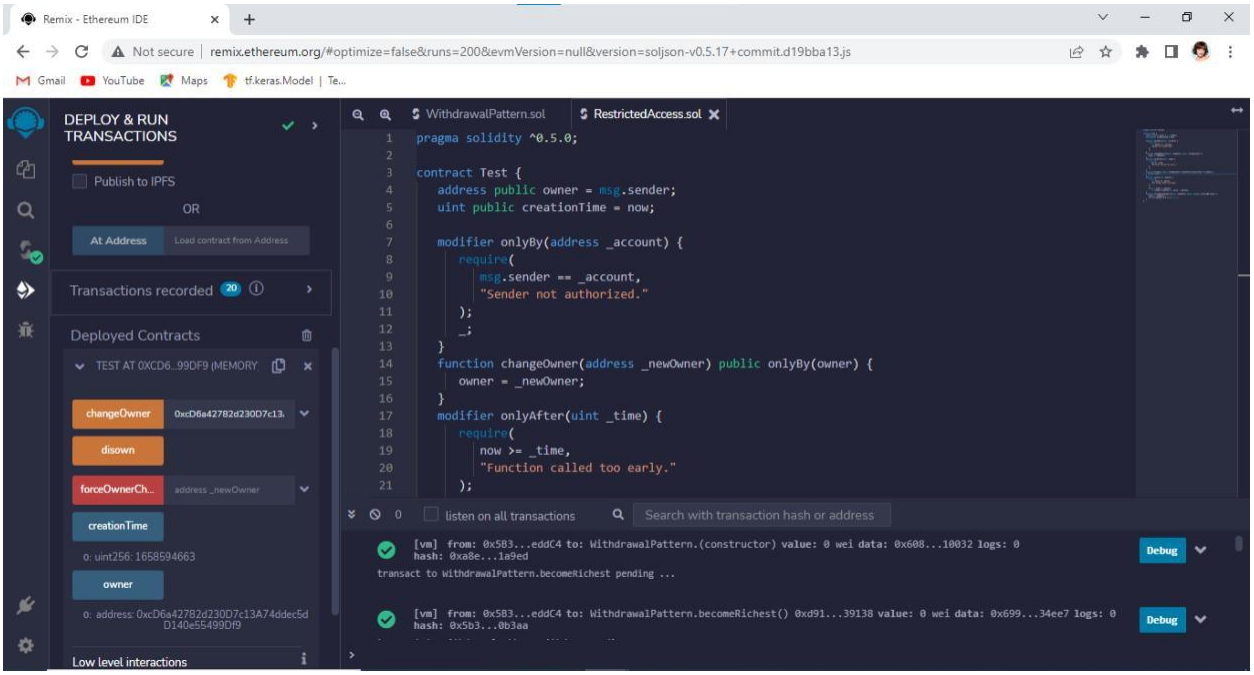
Code:





Output:

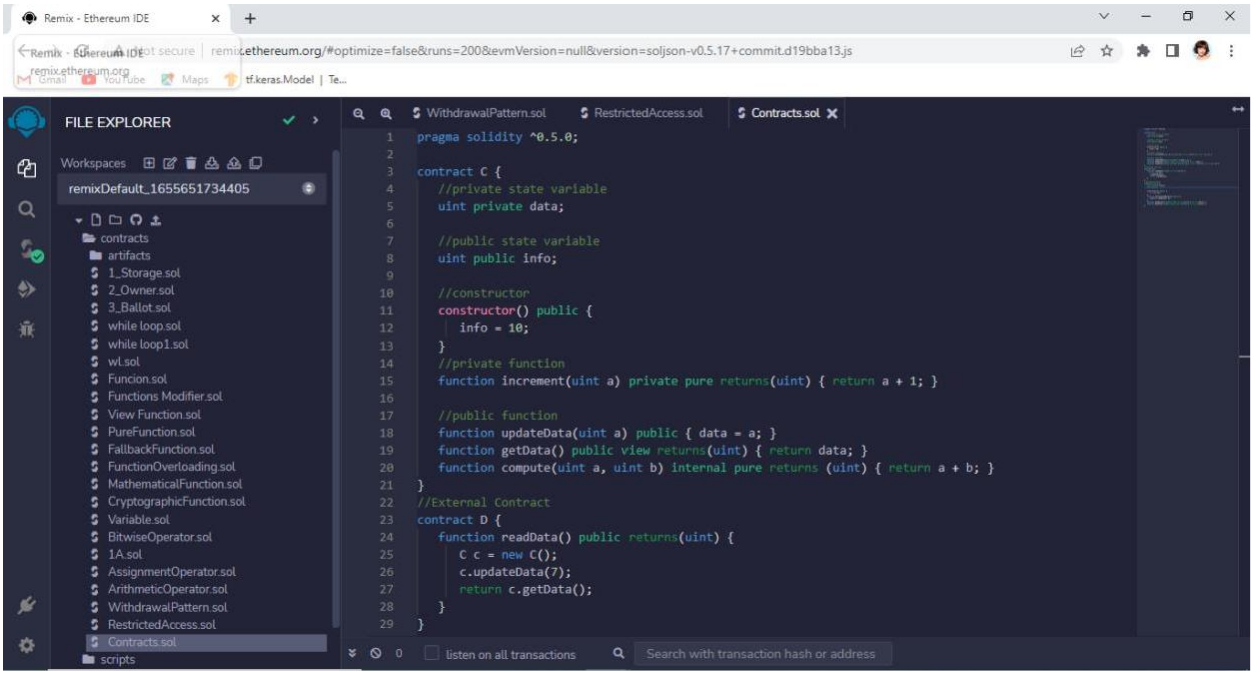

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
DEPLOY & RUN TRANSACTIONS
OR
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
TEST AT 0XD7A...F771B (MEMORY
changeOwner
disown
forceOwnerCh...
creationTime
owner
Low level interactions
CALLDATA
WithdrawalPattern.sol
RestrictedAccess.sol
delete owner;
}
modifier costs(uint _amount) {
require(
msg.value >= _amount,
"Not enough Ether provided."
);
_;
if (msg.value > _amount)
msg.sender.transfer(msg.value - _amount);
}
function forceOwnerChange(address _newOwner) public payable costs(200 ether) {
owner = _newOwner;
if (uint(owner) & 0 == 1) return;
}
}
listen on all transactions
Search with transaction hash or address

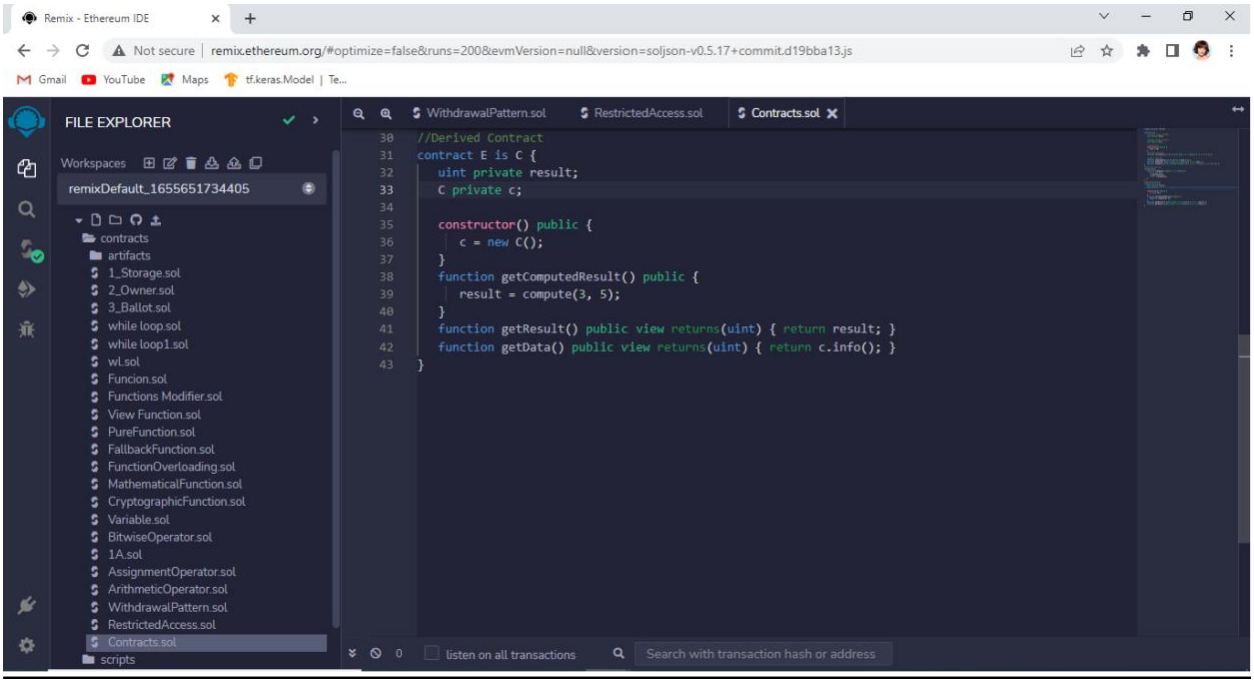

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
DEPLOY & RUN TRANSACTIONS
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 20
Deployed Contracts
TEST AT 0XCD6...99DF9 (MEMORY
changeOwner
0xcD6a42782d230D7c13...
disown
forceOwnerCh...
creationTime
owner
Low level interactions
WithdrawalPattern.sol
RestrictedAccess.sol
pragma solidity ^0.5.0;
contract Test {
address public owner = msg.sender;
uint public creationTime = now;
modifier onlyBy(address _account) {
require(
msg.sender == _account,
"Sender not authorized."
);
_;
}
function changeOwner(address _newOwner) public onlyBy(owner) {
owner = _newOwner;
}
modifier onlyAfter(uint _time) {
require(
now >= _time,
"Function called too early."
);
listen on all transactions
Search with transaction hash or address
[vm] from: 0x583...eddC4 to: WithdrawalPattern.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0xa8e...1a9ed
Debug
transact to WithdrawalPattern.becomeRichest pending ...
[vm] from: 0x583...eddC4 to: WithdrawalPattern.becomeRichest() 0xd91...39138 value: 0 wei data: 0x699...34ee7 logs: 0
hash: 0x5b3...0b3aa
Debug

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

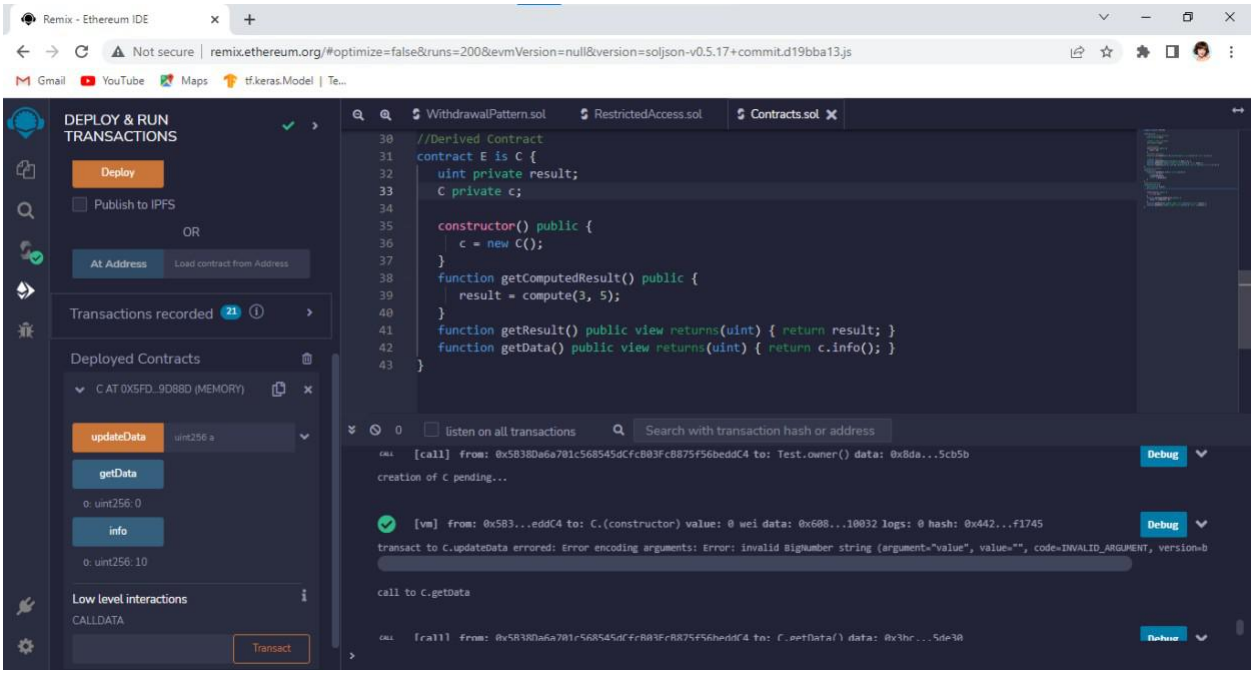
Contracts

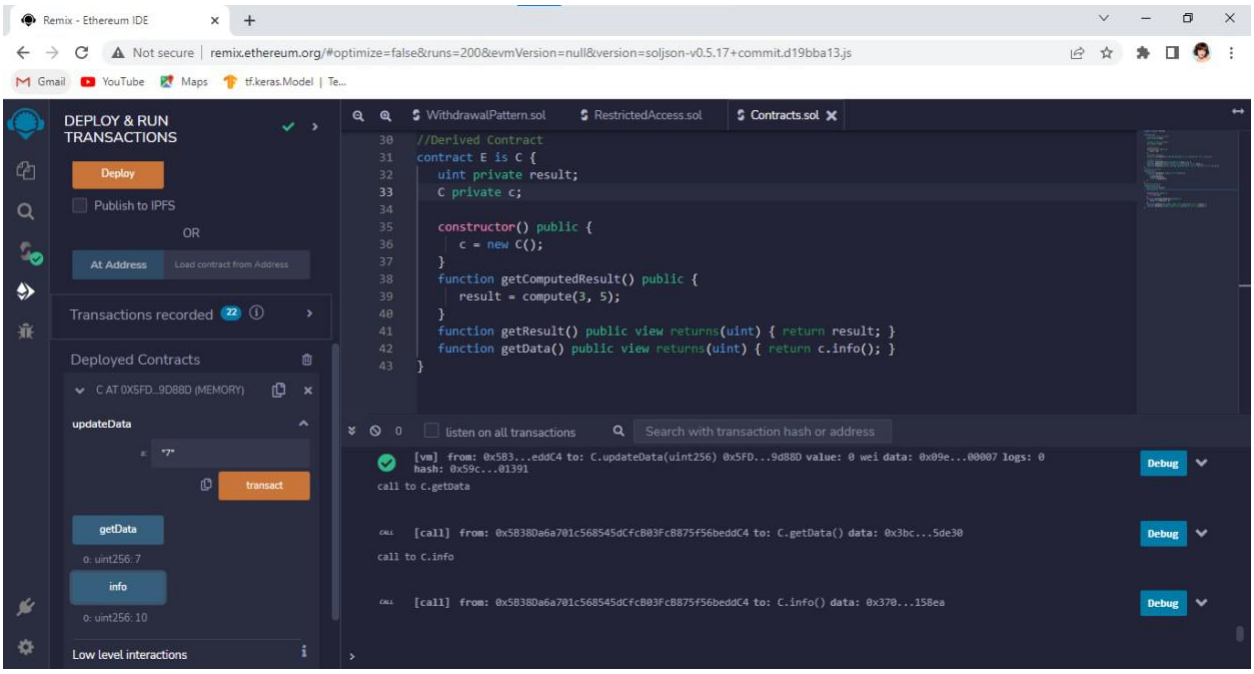
Code:





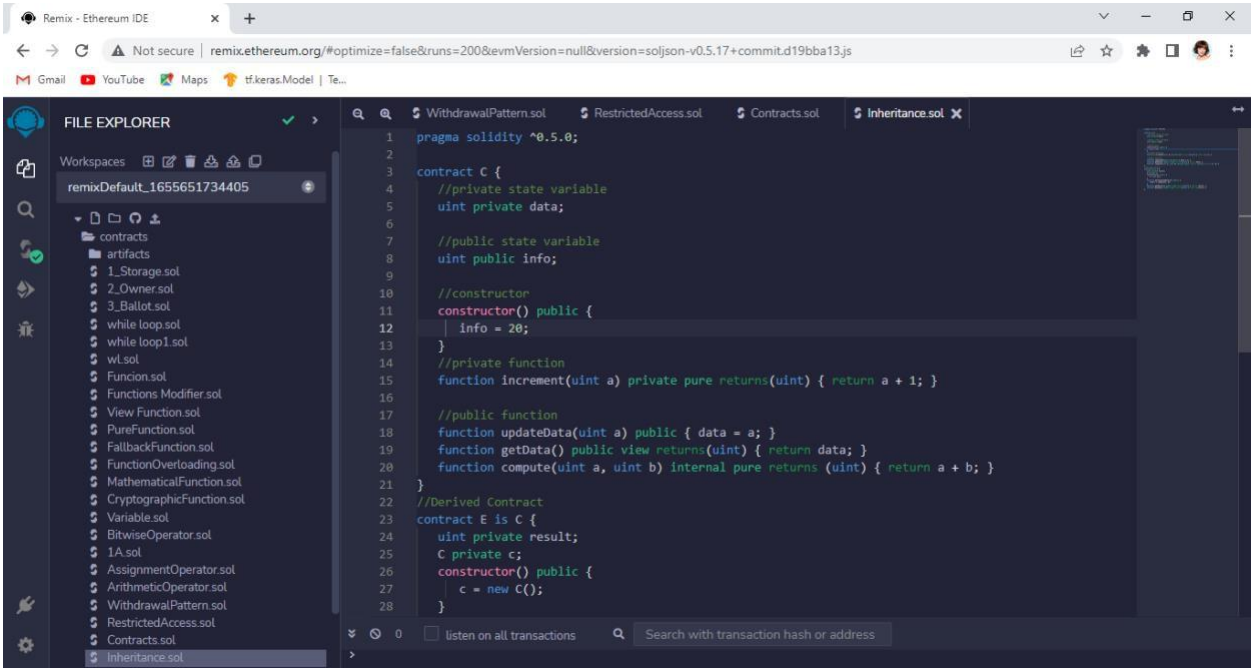
Output:

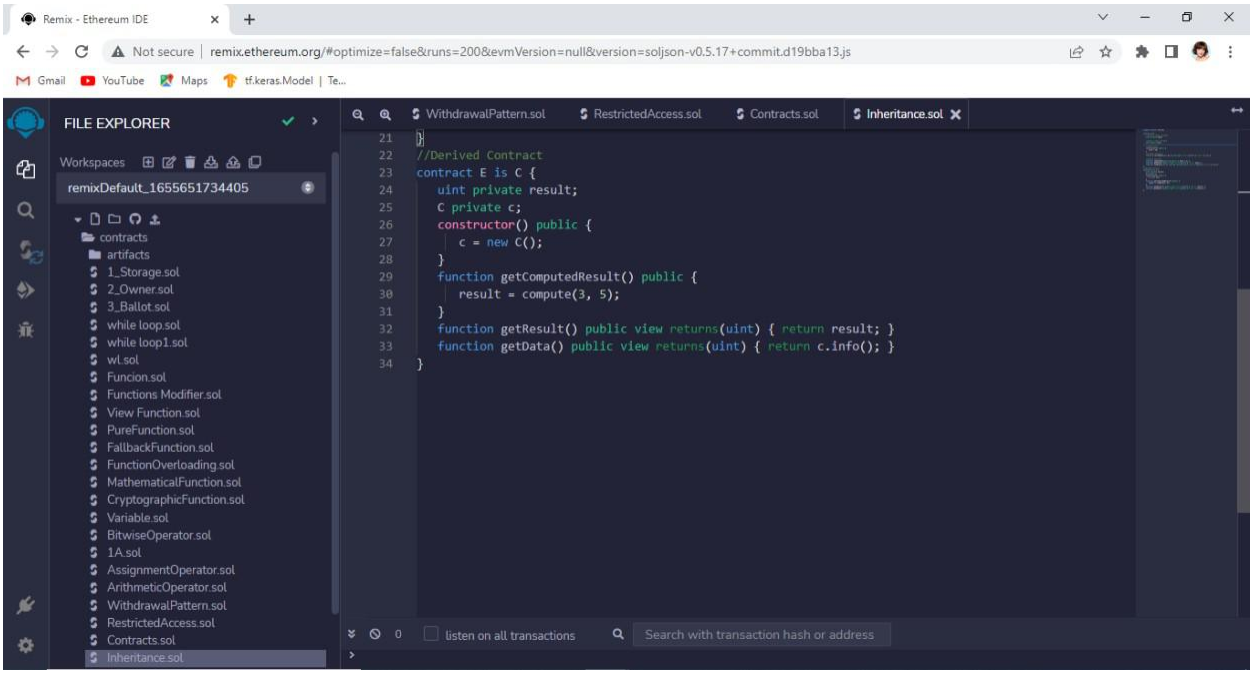

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 21
Deployed Contracts
C AT 0X5FD...9D88D (MEMORY)
updateData
uint256 a
getData
0: uint256: 0
info
0: uint256: 10
Low level interactions
CALLDATA
Transact
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
//Derived Contract
contract E is C {
uint private result;
C private c;
constructor() public {
c = new C();
}
function getComputedResult() public {
result = compute(3, 5);
}
function getResult() public view returns(uint) { return result; }
function getData() public view returns(uint) { return c.info(); }
}
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


Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 22
Deployed Contracts
C AT 0X5FD...9D88D (MEMORY)
updateData
x: "7"
transact
getData
0: uint256: 7
info
0: uint256: 10
Low level interactions
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
//Derived Contract
contract E is C {
uint private result;
C private c;
constructor() public {
c = new C();
}
function getComputedResult() public {
result = compute(3, 5);
}
function getResult() public view returns(uint) { return result; }
function getData() public view returns(uint) { return c.info(); }
}
listen on all transactions
Search with transaction hash or address
[vm] from: 0x583...eddC4 to: C.updateData(uint256) 0x5FD...9d88D value: 0 wei data: 0x09e...00007 logs: 0 hash: 0x59c...01391
Debug
call to C.getData
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: C.getData() data: 0x3bc...5de30
Debug
call to C.info
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: C.info() data: 0x370...158ea
Debug

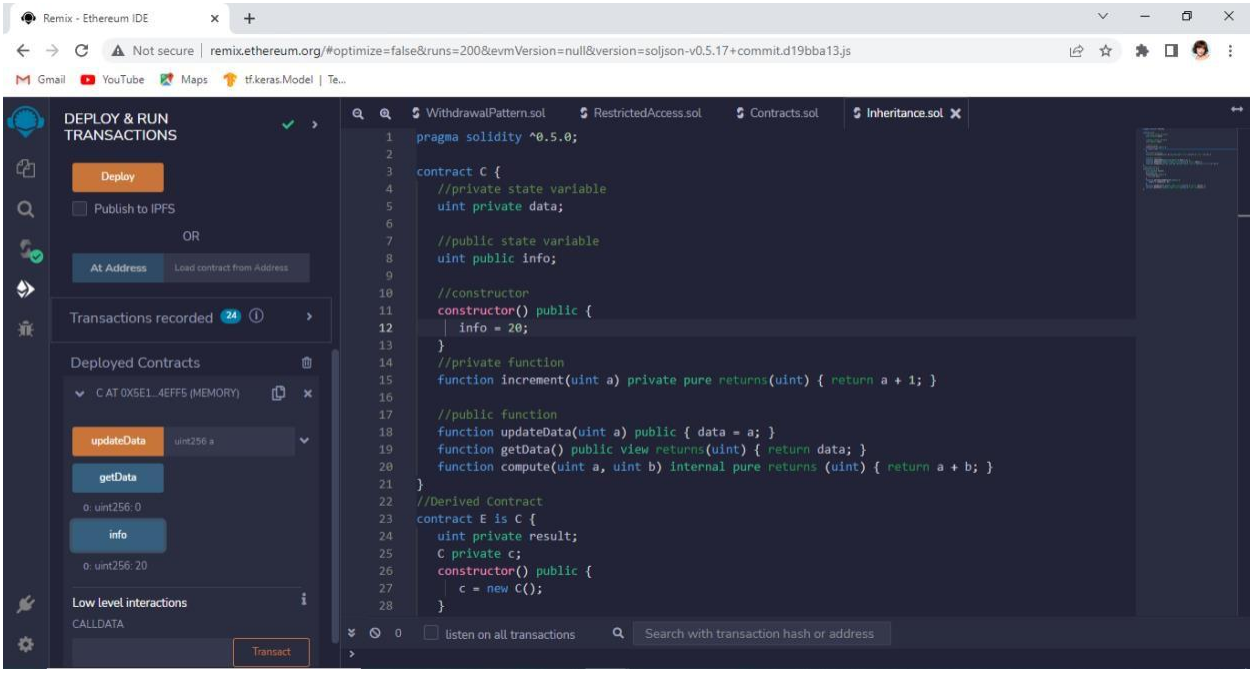
Inheritance

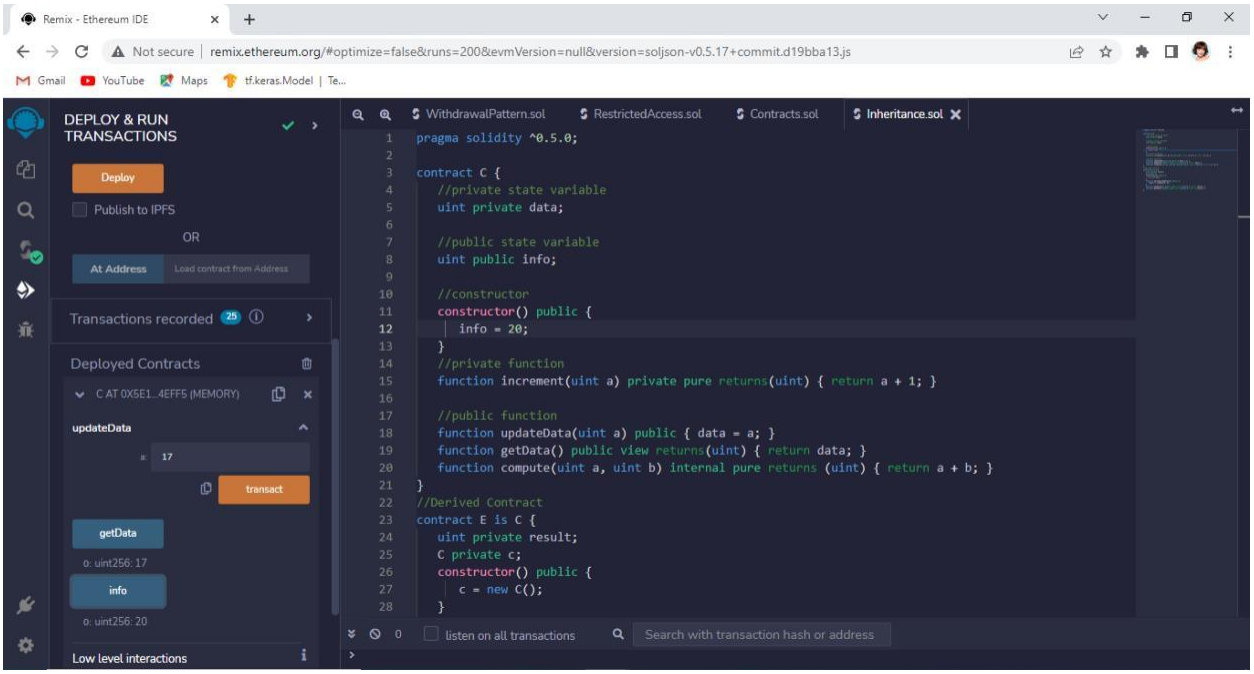
Code:





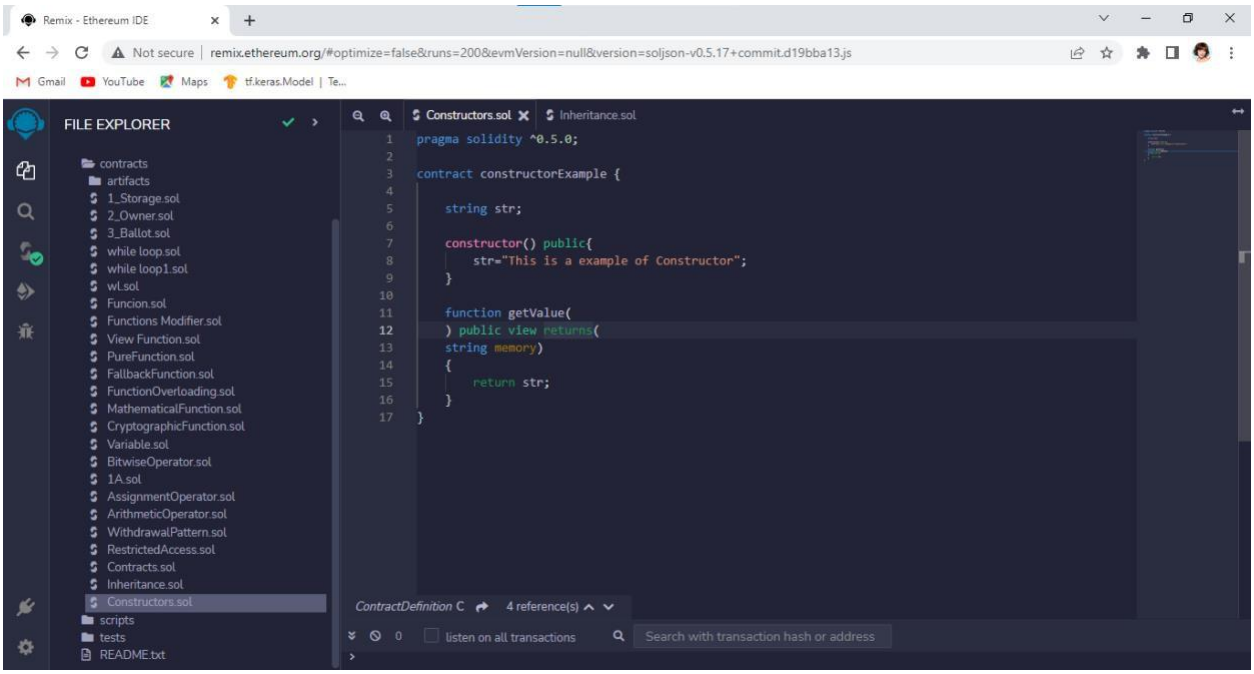
Output:


Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
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
pragma solidity ^0.5.0;
contract C {
//private state variable
uint private data;
//public state variable
uint public info;
//constructor
constructor() public {
info = 20;
}
//private function
function increment(uint a) private pure returns(uint) { return a + 1; }
//public function
function updateData(uint a) public { data = a; }
function getData() public view returns(uint) { return data; }
function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}
//Derived Contract
contract E is C {
uint private result;
C private c;
constructor() public {
c = new C();
}
listen on all transactions
Search with transaction hash or address

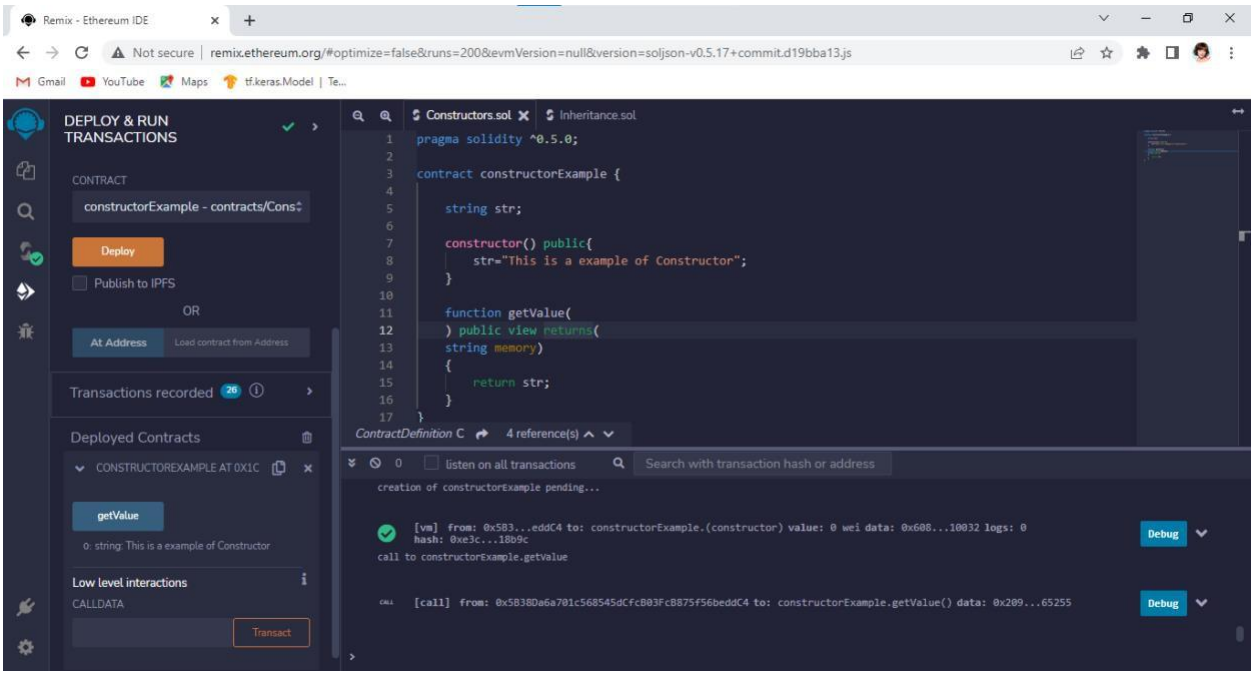

Remix - Ethereum IDE
Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 25
Deployed Contracts
C AT 0X5E1...4EFF5 (MEMORY)
updateData
17
transact
getData
0: uint256: 17
info
0: uint256: 20
Low level interactions
WithdrawalPattern.sol
RestrictedAccess.sol
Contracts.sol
Inheritance.sol
pragma solidity ^0.5.0;
contract C {
//private state variable
uint private data;
//public state variable
uint public info;
//constructor
constructor() public {
info = 20;
}
//private function
function increment(uint a) private pure returns(uint) { return a + 1; }
//public function
function updateData(uint a) public { data = a; }
function getData() public view returns(uint) { return data; }
function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}
//Derived Contract
contract E is C {
uint private result;
C private c;
constructor() public {
c = new C();
}
listen on all transactions
Search with transaction hash or address

Constructors

Code:

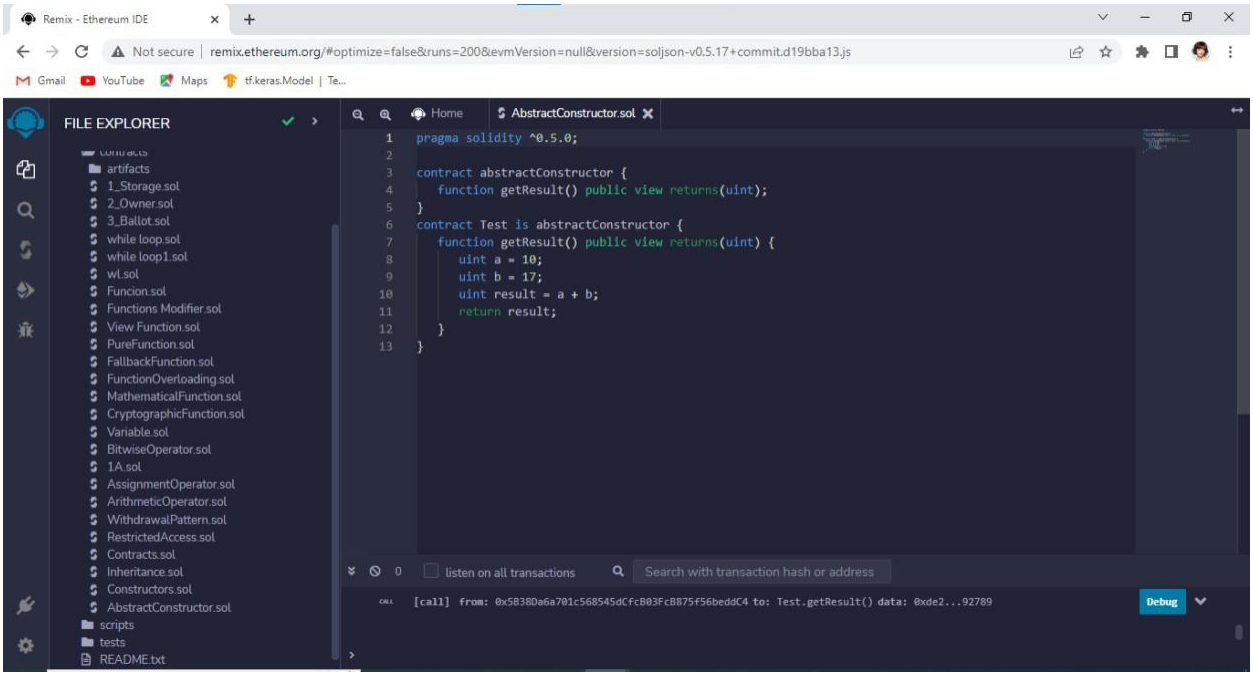


Output:

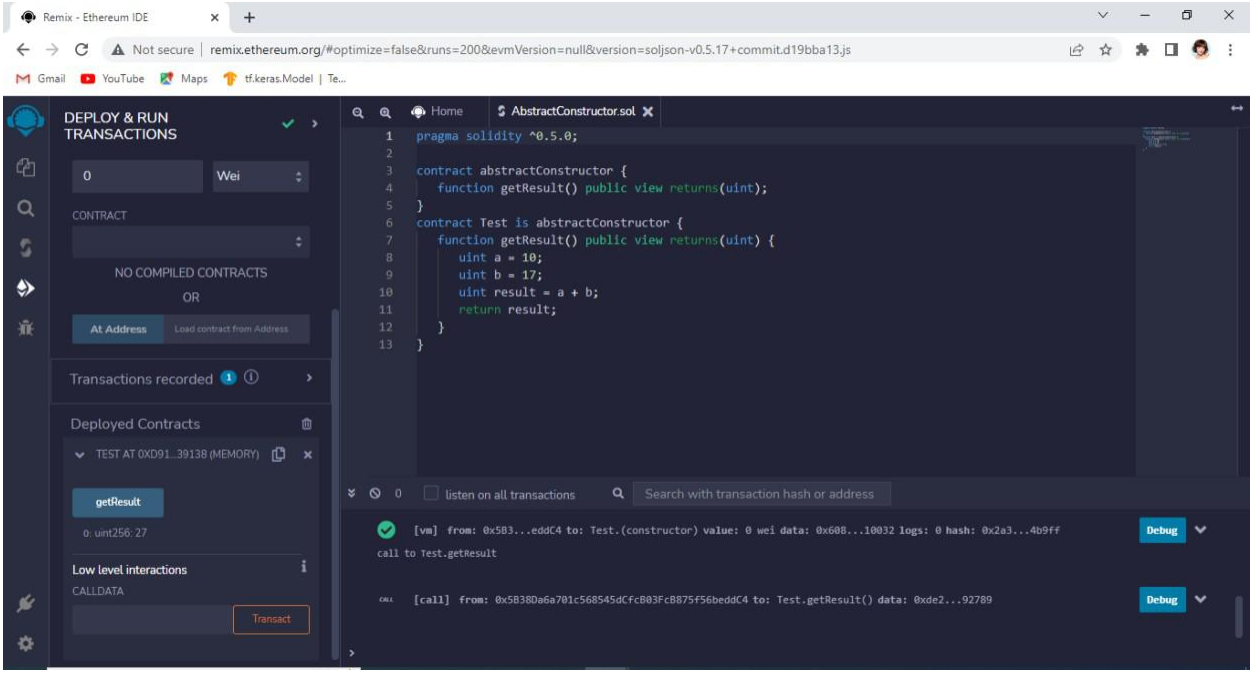


Abstract Contracts

Code:

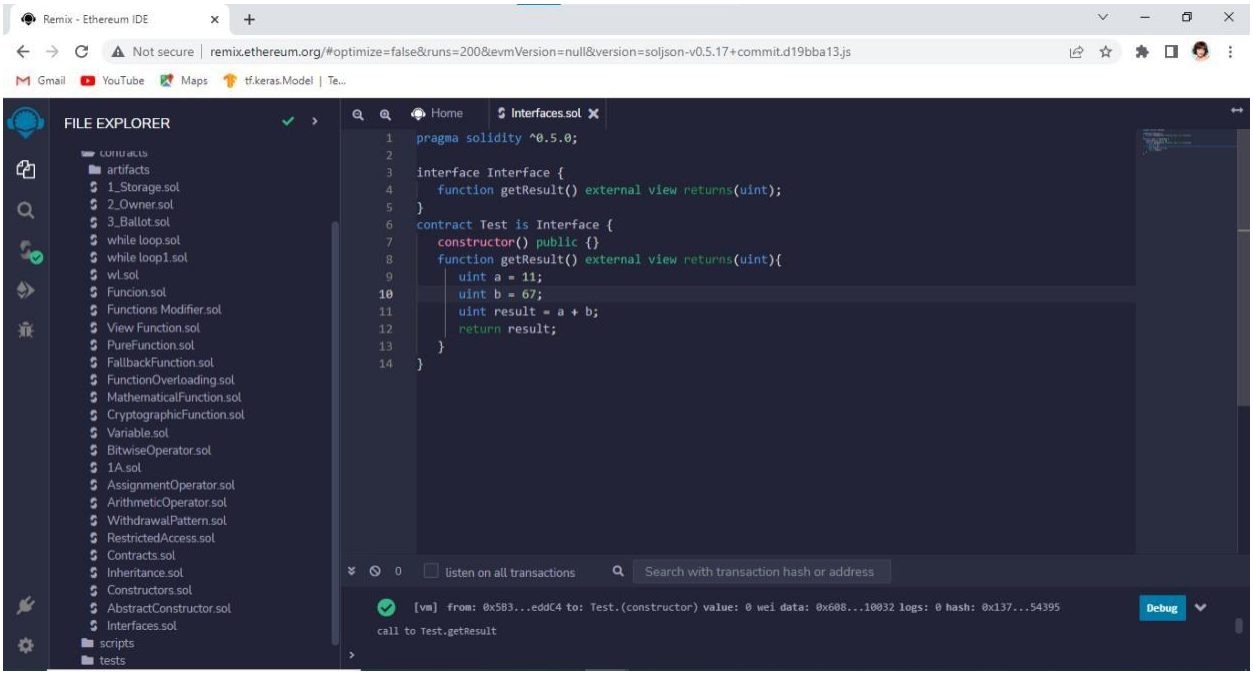


Output:

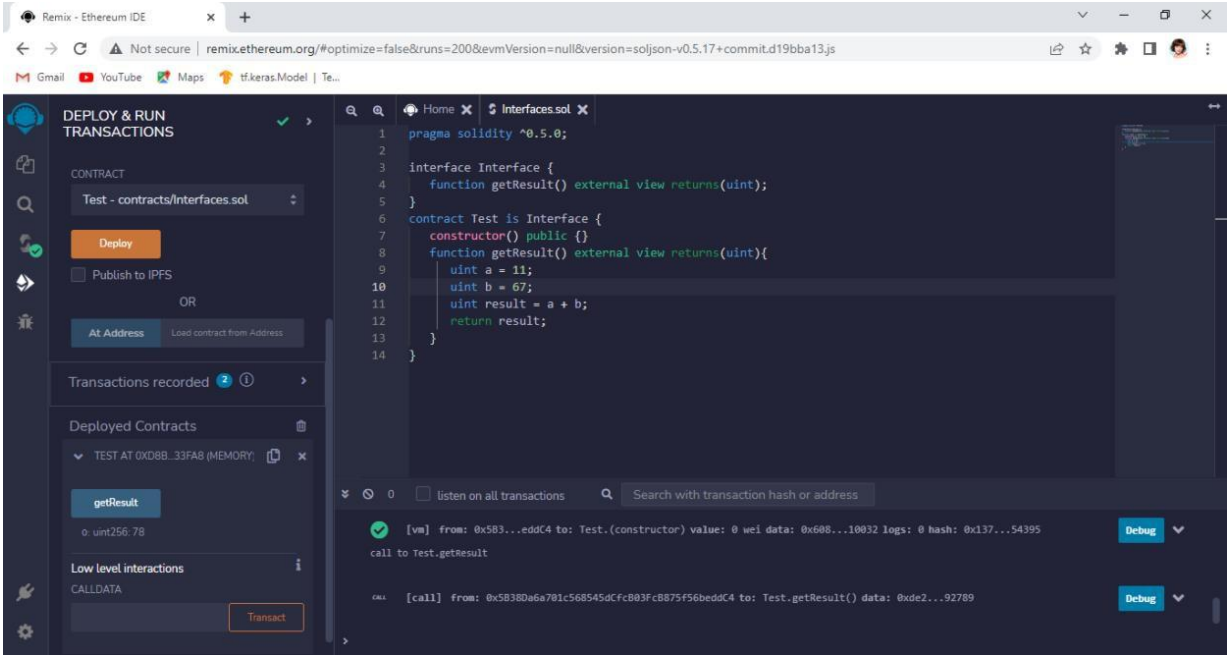


Interfaces

Code:



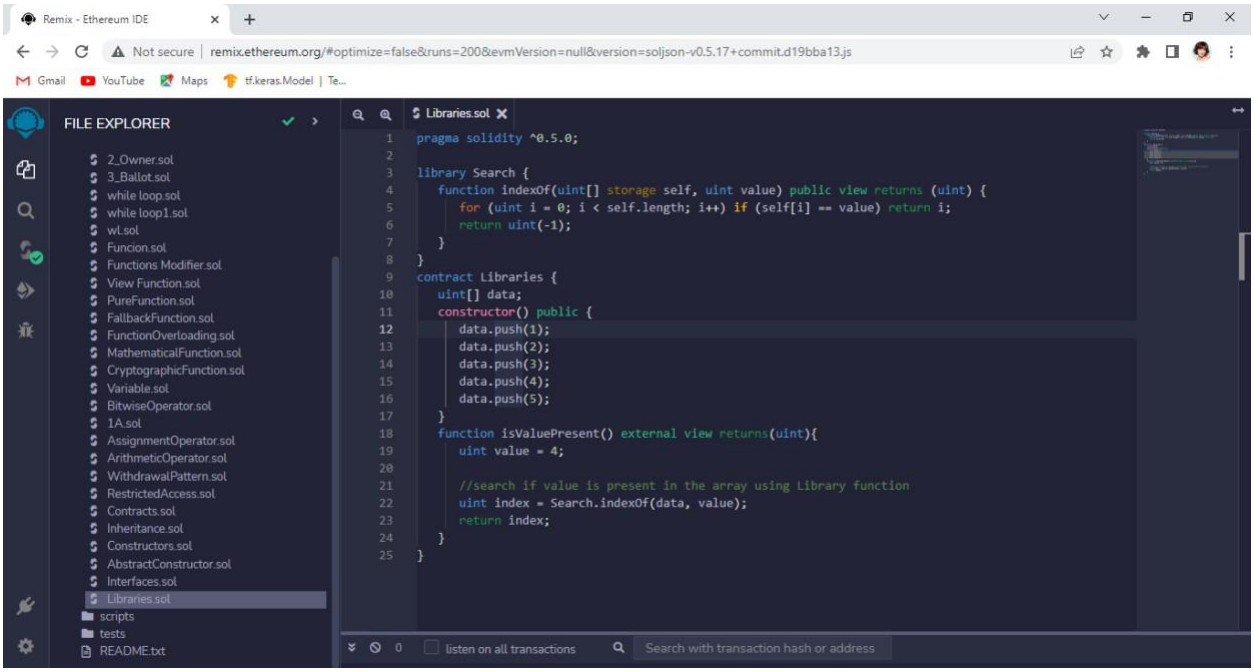
Output:



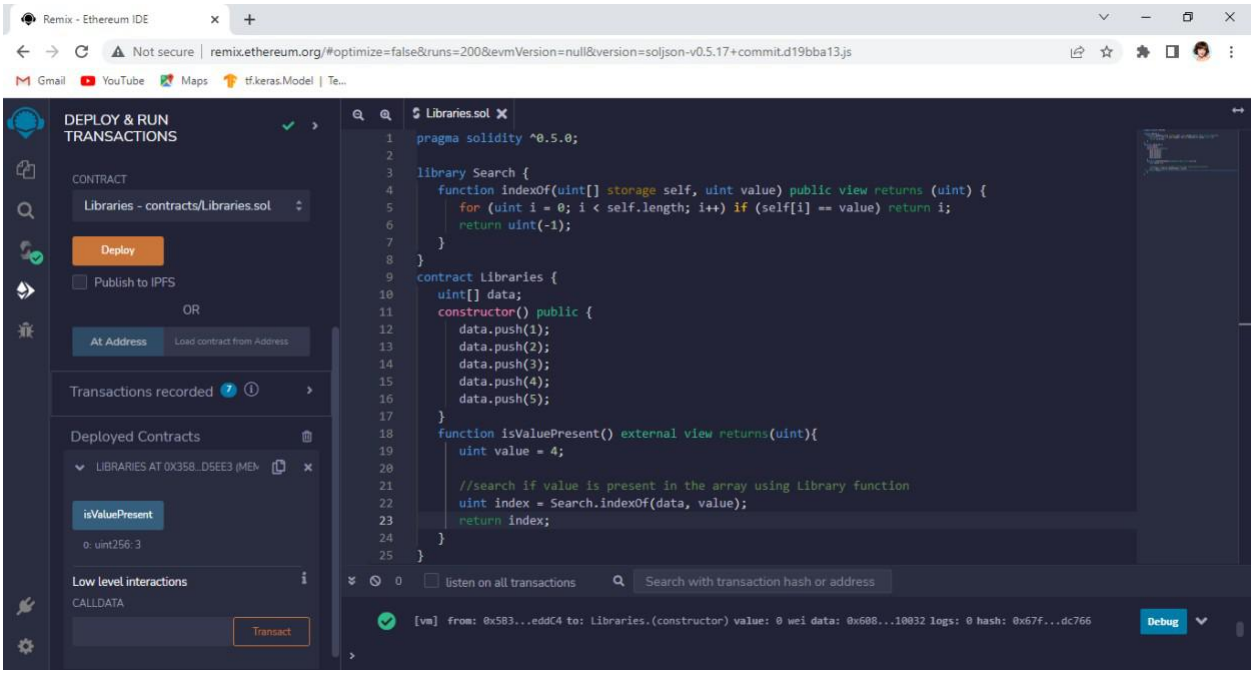
3c) Libraries, Assembly, Events, Error handling.

Libraries

Code:

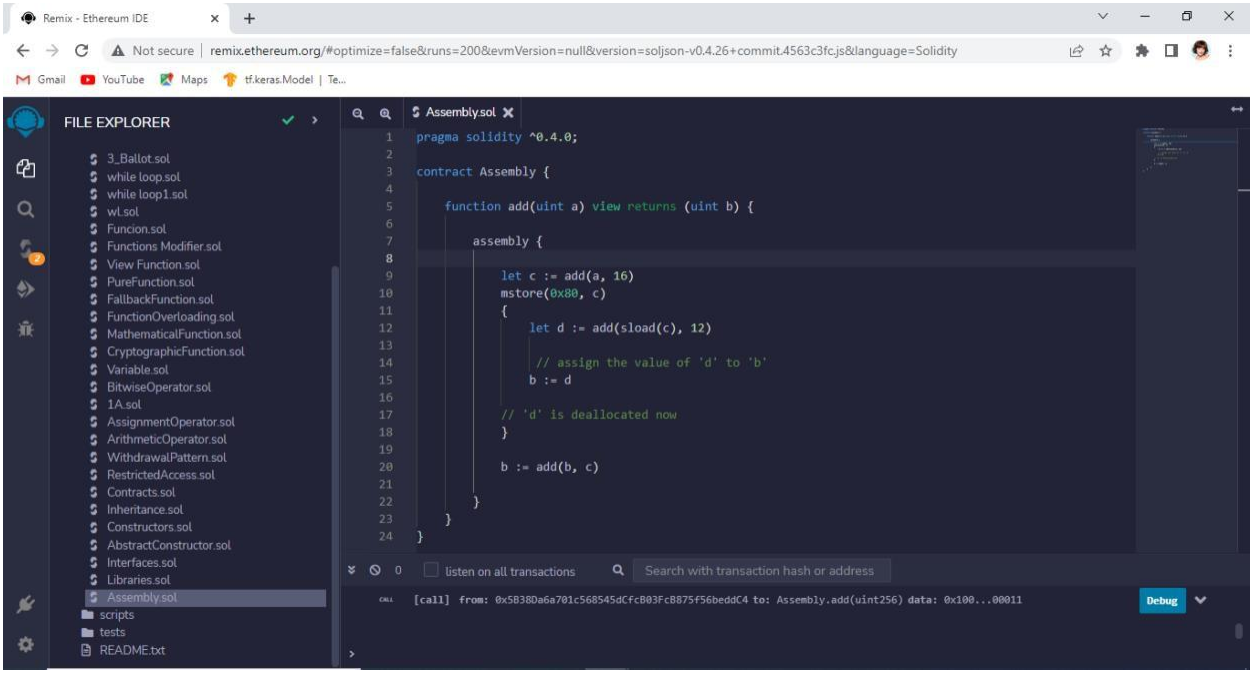


Output:

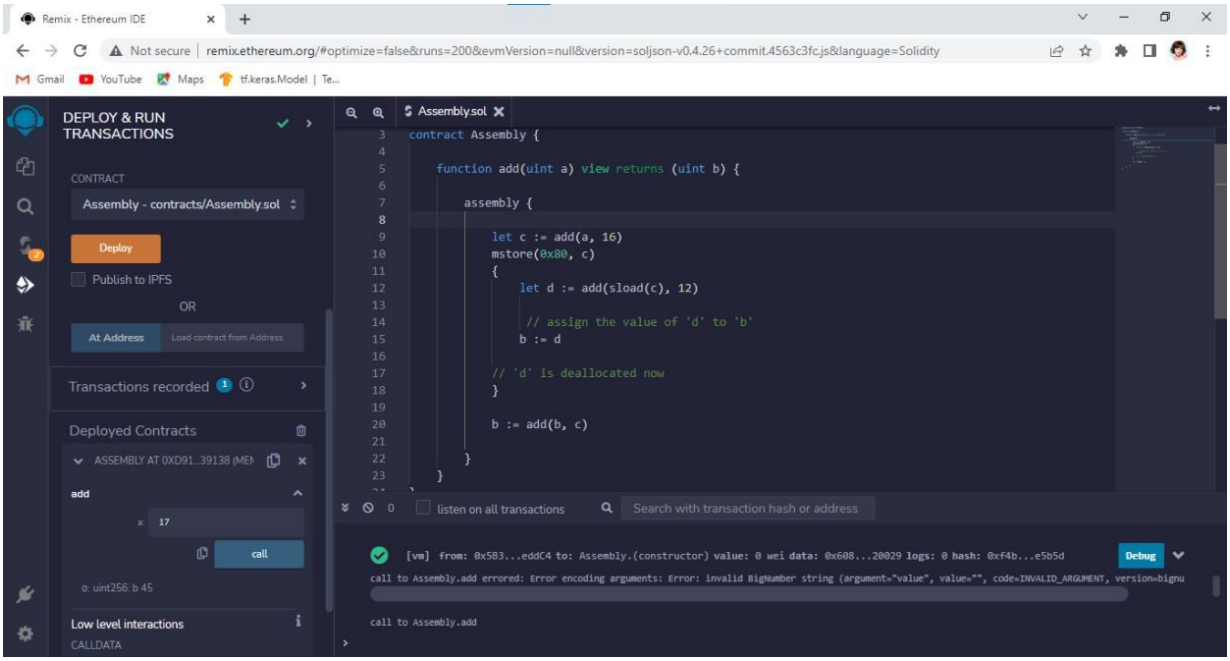


Assembly

Code:

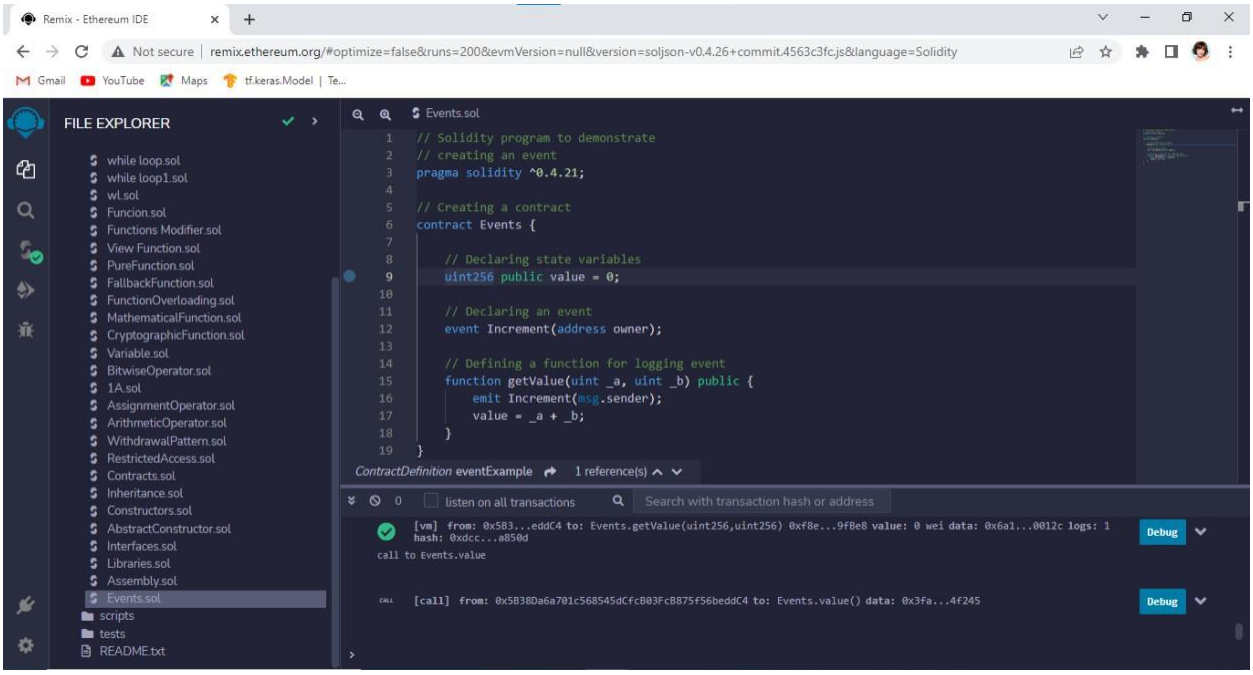


Output:

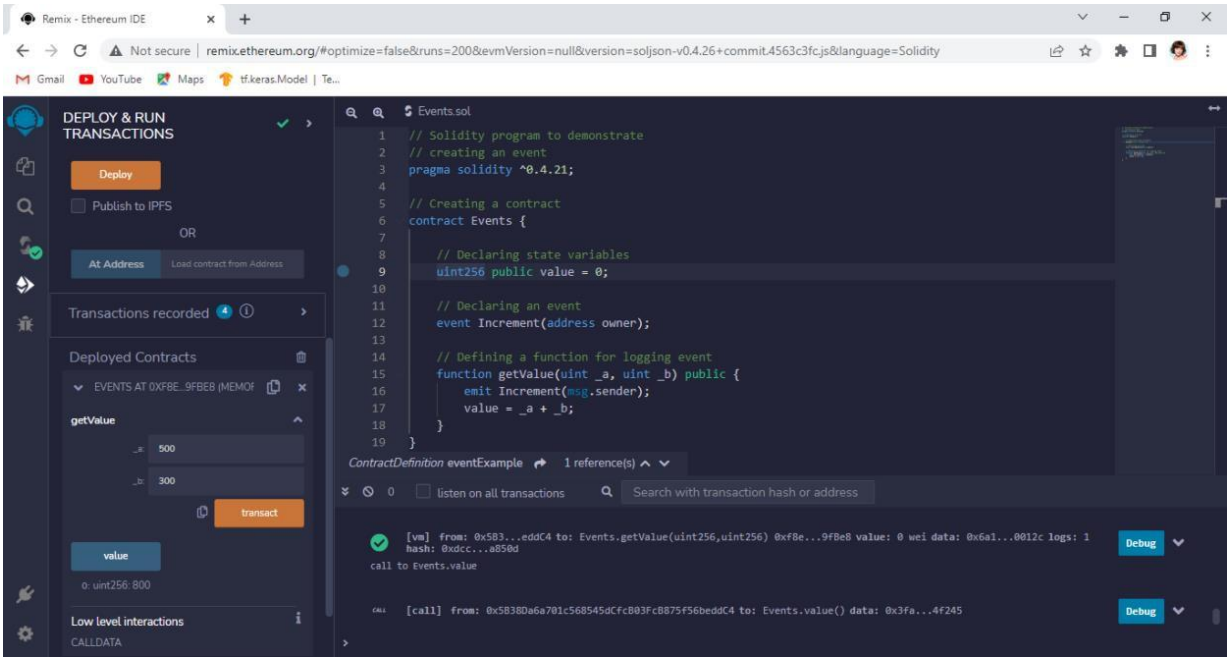


Events

Code:

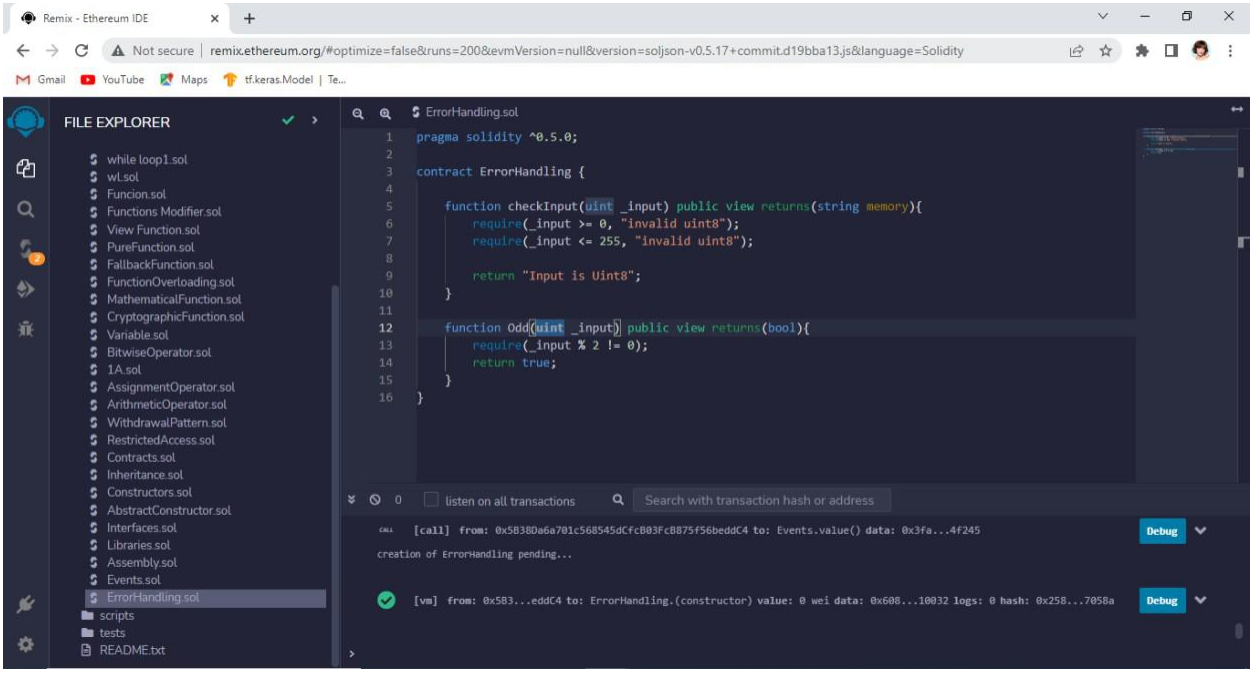


Output:

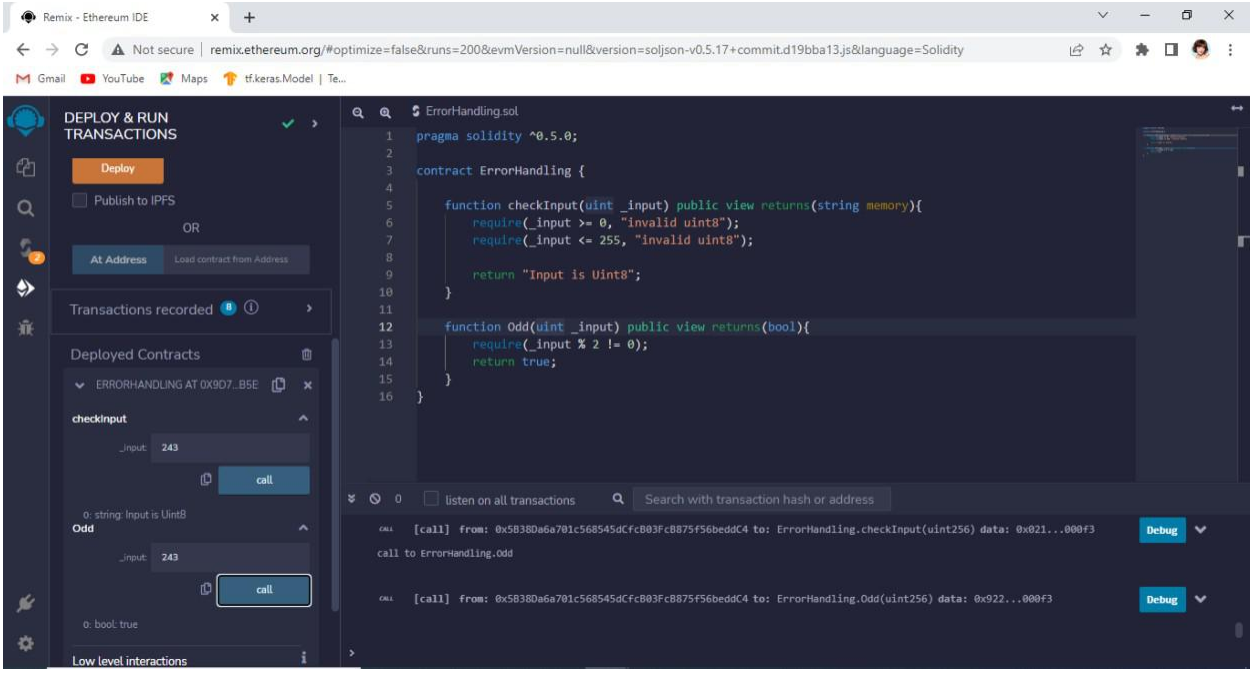


Error Handling

Code:



Output:



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
```

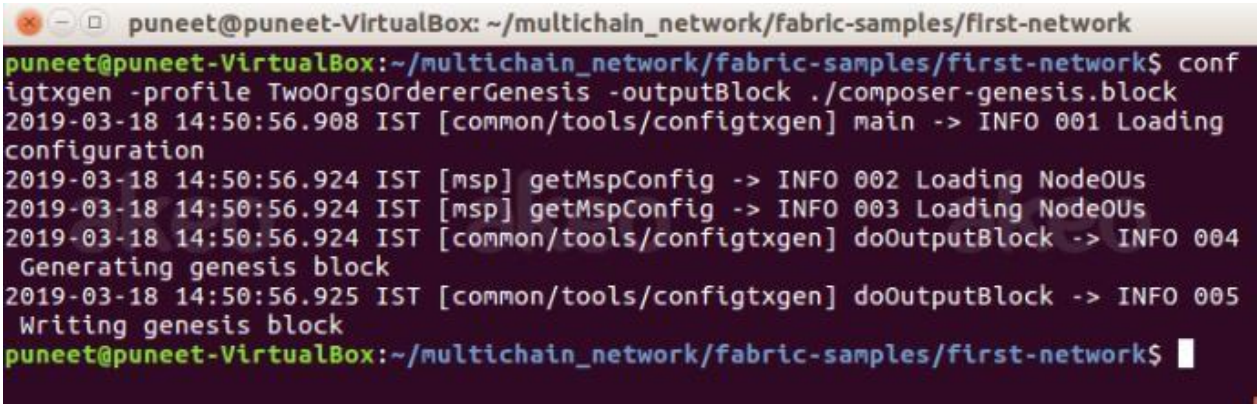
This will create all certificates for orderers and peers in crypto-config folder.

Copy certificates for all peers and orderer to temporary folder.

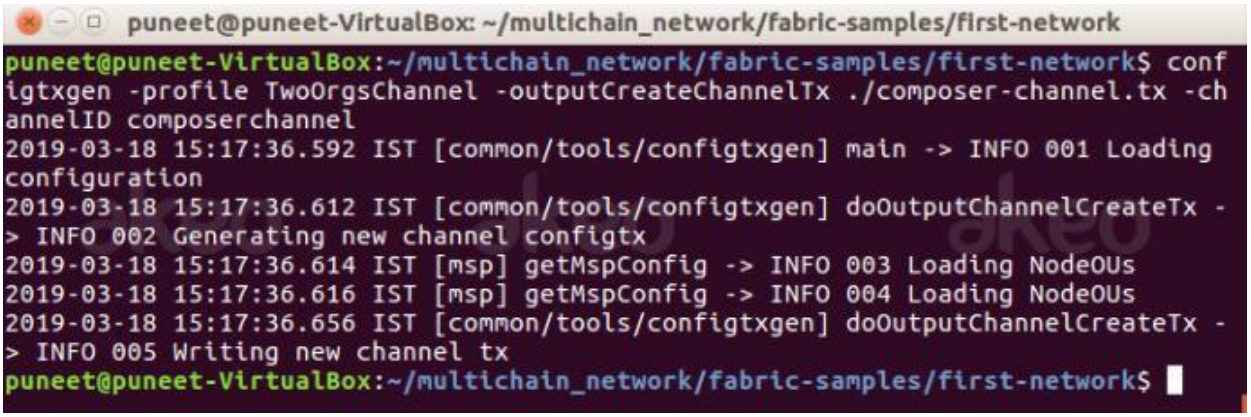
```
In first-network folder run this command — mkdir -p tmp/composer/org1

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.

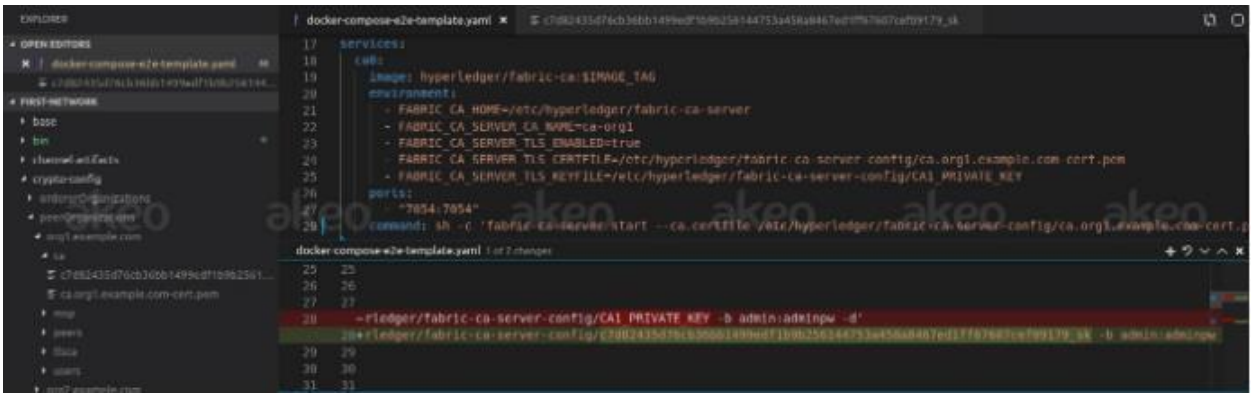
Under services section we have two certificate authorities named “ca0” and “ca1”.

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"
```

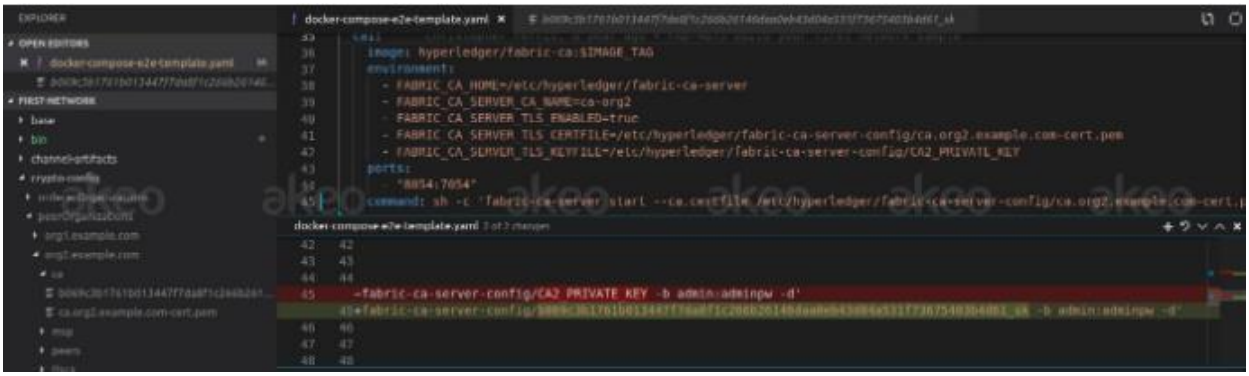


For ca1 — goto command section under ca1 and find CA2_PRIVATE_KEY and replace it with private key –

```
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 –

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.

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 SHA256(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__=='__main__':
    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