

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

Sr.No	Practical name	Pg.No
1.	Write the following programs for Blockchain in Python:	2
a.	A simple client class that generates the private and public keys by using the builtin Python RSA algorithm and test it.	2
b.	A transaction class to send and receive money and test it.	2
c.	Create multiple transactions and display them.	3
d.	Create a blockchain, a genesis block and execute it.	4
e.	Create a mining function and test it.	5
f.	Add blocks to the miner and dump the blockchain.	6
2.	Implement and demonstrate the use of the following in Solidity:	8
a.	Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.	8
b.	Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.	26
3.	Implement and demonstrate the use of the following in Solidity:	35
a.	Withdrawal Pattern, Restricted Access.	35
b.	Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.	38
c.	Libraries, Assembly, Events, Error handling.	46
4.	Install hyperledger fabric and composer. Deploy and execute the application.	50
5.	Demonstrate the running of the blockchain node.	57
6.	Demonstrate the use of Bitcoin Core API.	58
7.	Create your own blockchain and demonstrate its use.	59

Practical No: 1

Aim: Write the following programs for Blockchain in Python:

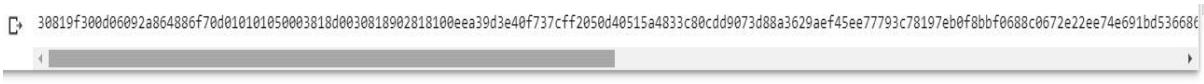
1a) A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it.

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

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

Output:



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: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3035
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434e
-----
value: 15.0
-----
time: 2022-07-30 17:35:06.809777
-----
-----
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3035
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a72cfb527dff9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2l
-----
value: 6.0
-----
time: 2022-07-30 17:35:06.811866
-----
-----
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a814
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d576
-----
value: 2.0
-----
time: 2022-07-30 17:35:06.814247
-----
-----
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100a72cfb527dff9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2b268
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434e
-----
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: 30819f300d06092a864886f70d010101050003818d0030818902818100a4a79ac0645d8483c7c97fb6b67e1bf90f17d87d1f1
-----
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: 11e0a4b57bb76496ecc6ab5a3c126165bc9dbbaaf80f664e7e192a422360c3884
after 419 iterations found nonce: 11280dfd9ab05b3dbfa86990153732941408faa4a3b0832819b161f52321c08
after 511 iterations found nonce: 1150944bd7ea429acd052da390b148f27eb881686e0f8bfcdf77f833af878e2e
after 822 iterations found nonce: 110e61d41d94b48f6dfcb6f43f9ceafe2ab7168456dab088e05126d43d0366ef
after 924 iterations found nonce: 11147bf32bafeeee4505b5cc40c3b227366d8a41ab3bd740e437c74400b7a8127
'6d80c4cf3cd4fb49aad25deced696c3a48e17974ecfb1441e2128517f0152b2'
```

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
```

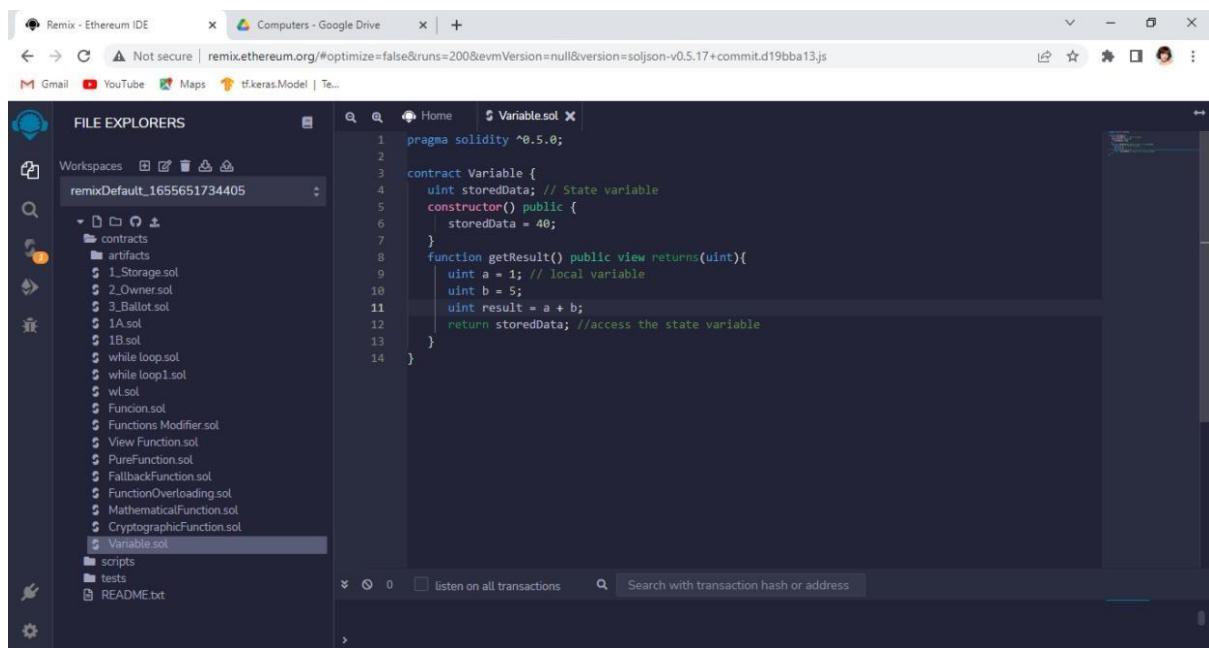
Practical No: 2

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:



The screenshot shows the Remix Ethereum IDE interface. The title bar says "Remix - Ethereum IDE". The address bar shows the URL "remix.ethereum.org/#optimize=false&runtimes=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". The left sidebar has a "FILE EXPLORERS" section with a "Workspaces" dropdown set to "remixDefault_1655651734405". Under "contracts", there are several files listed: 1.Storage.sol, 2.Owner.sol, 3.Ballot.sol, 1A.sol, 1B.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, and Variable.sol. The "scripts" and "tests" sections are also visible. The main editor area is titled "Variable.sol" and contains the following Solidity code:

```
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
    }
}
```

Output:

The screenshot shows the Ethereum IDE (Remix) interface. The top bar displays the title "Remix - Ethereum IDE" and the URL "remix.ethereum.org/#optimize=false&runtimes=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". Below the URL, there are tabs for "Gmail", "YouTube", "Maps", and "tf.keras.Model".

The main area is divided into several sections:

- Deploy & Run Transactions:** A sidebar with icons for deploying contracts, viewing logs, and interacting with deployed contracts.
- Transactions recorded:** A section showing 28 recorded transactions.
- Deployed Contracts:** A list showing "VARIABLE AT 0xEF9...10EBF (MEMO)".
- Contract Editor:** The code editor window contains the following Solidity code:

```
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
    }
}
```
- Low level interactions:** A section for interacting with the deployed contract, showing a "getResult" button and a "Transact" button.

Operators:

Arithmetic Operator

Code:

The screenshot shows the Ethereum IDE interface. In the top right, there's a browser bar with the URL remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity. The main area is titled "FILE EXPLORER" and lists several Solidity files. The file "ArithmeticOperator.sol" is selected and shown in the code editor. The code defines a contract with methods for addition, subtraction, multiplication, division, and modulus. Below the code editor, the transaction history shows two constructor calls for the "ArithmeticOperator" contract. The bottom part of the interface has a sidebar titled "DEPLOY & RUN TRANSACTIONS" with buttons for each arithmetic operation.

```
pragma solidity ^0.5.0;

contract ArithmaticOperator {
    uint16 public a = 50;
    uint16 public b = 20;

    uint public sum = a + b;
    uint public diff = a - b;
    uint public mul = a * b;

    uint public div = a / b;
    uint public mod = a % b;

    uint public dec = --b;
    uint public inc = ++a;
}
```

Output:

This screenshot shows the Ethereum IDE after the "ArithmeticOperator" contract has been deployed. The "DEPLOY & RUN TRANSACTIONS" sidebar now displays buttons for each of the eight arithmetic operations defined in the contract. The code editor still shows the same Solidity code. The transaction history at the bottom shows two constructor calls and two calls to the "sum" function, indicating that the contract is functional.

```
pragma solidity ^0.5.0;

contract ArithmaticOperator {
    uint16 public a = 50;
    uint16 public b = 20;

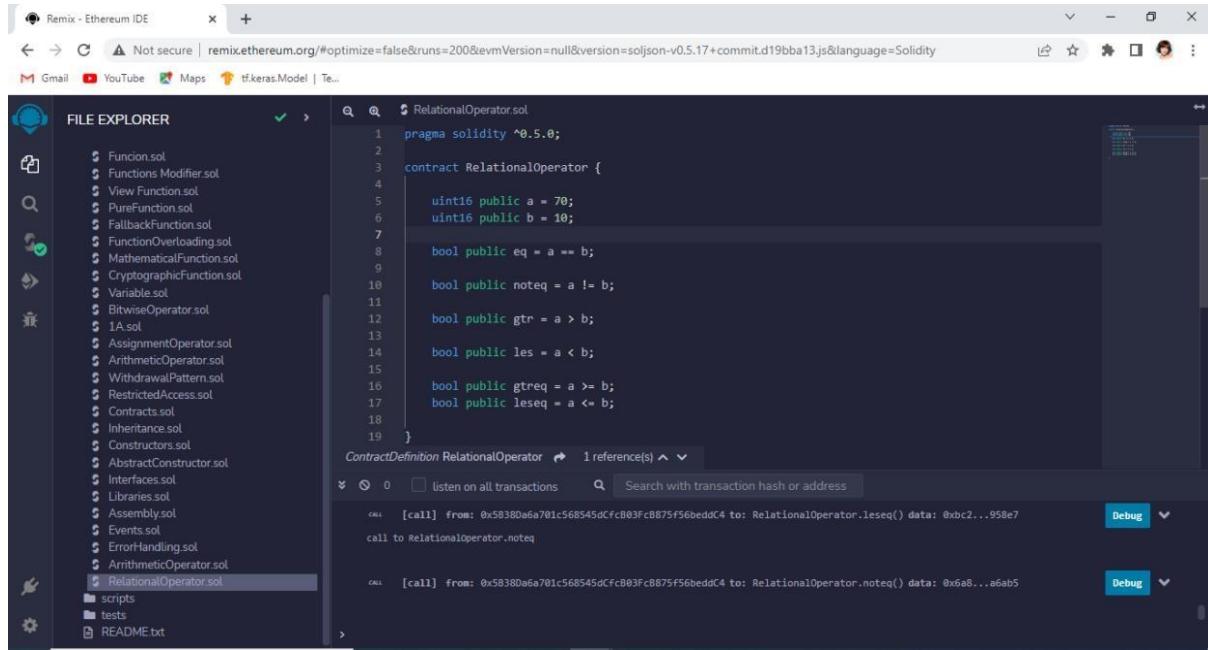
    uint public sum = a + b;
    uint public diff = a - b;
    uint public mul = a * b;

    uint public div = a / b;
    uint public mod = a % b;

    uint public dec = --b;
    uint public inc = ++a;
}
```

Relational Operator

Code:



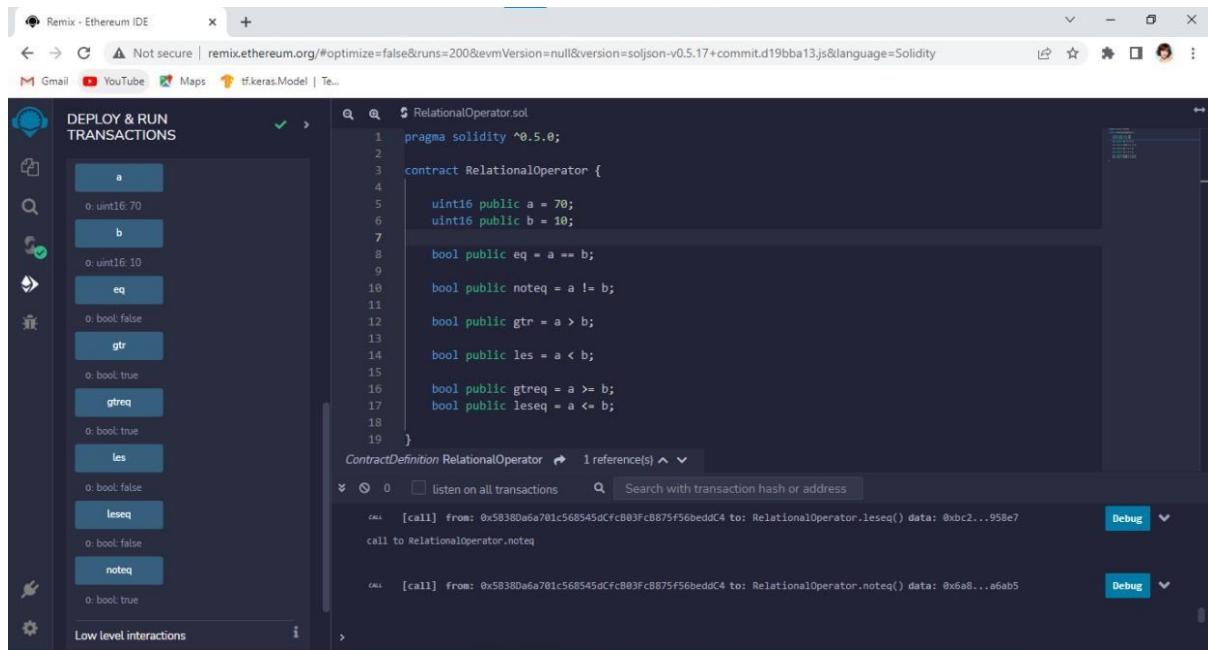
The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files: Funcion.sol, Functions.Modifiers.sol, ViewFunction.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, 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, ArithmeticOperator.sol, RelationalOperator.sol, scripts, tests, and README.txt. The "RelationalOperator.sol" file is currently selected. The main editor area displays the following Solidity code:

```
pragma solidity ^0.5.0;
contract RelationalOperator {
    uint16 public a = 70;
    uint16 public b = 10;
    bool public eq = a == b;
    bool public noteq = a != b;
    bool public gtr = a > b;
    bool public les = a < b;
    bool public gtreq = a >= b;
    bool public leseq = a <= b;
}
```

The "ContractDefinition RelationalOperator" section shows "1 reference(s)". Below the code, there are two transaction logs:

- Call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.leseq() data: 0xb2...958e7 Debug
- Call to relationaloperator.noteq
- Call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.noteq() data: 0x6a8...a6ab5 Debug

Output:

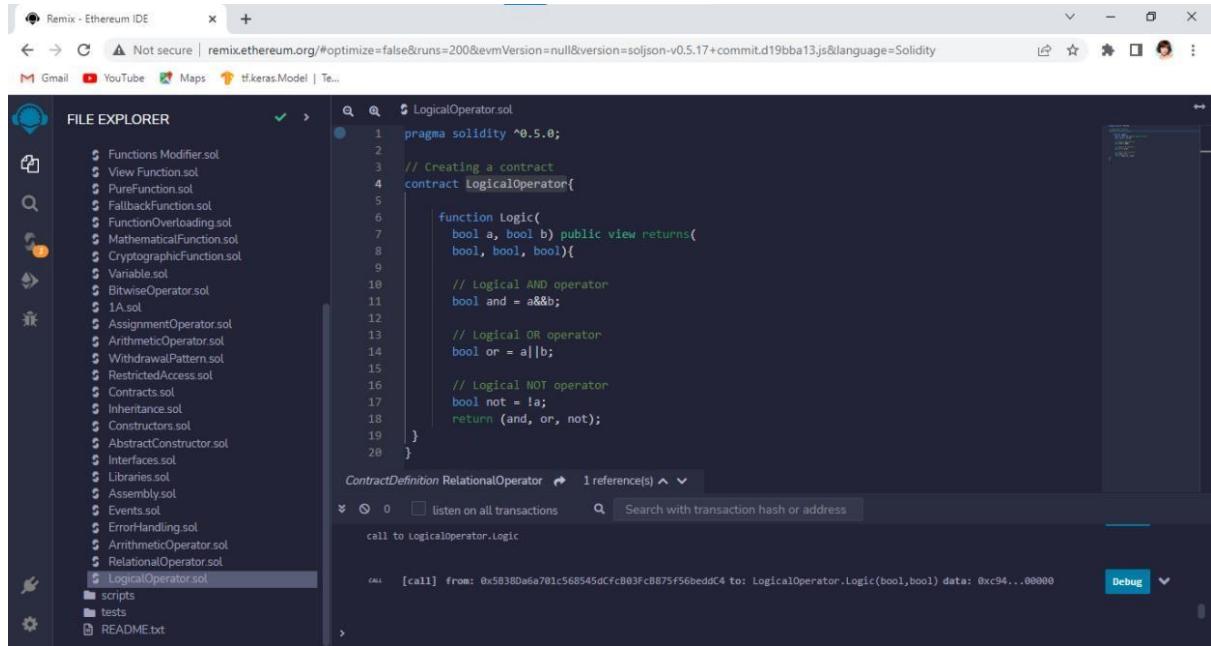


The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. On the left, there is a sidebar with buttons for "a", "b", "eq", "gt", "gtreq", "les", "lesq", "noteq", and "Low level interactions". The "noteq" button is highlighted. The main editor area shows the same Solidity code as the previous screenshot. The "ContractDefinition RelationalOperator" section shows "1 reference(s)". Below the code, there are two transaction logs:

- Call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.leseq() data: 0xb2...958e7 Debug
- Call to relationaloperator.noteq
- Call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.noteq() data: 0x6a8...a6ab5 Debug

Logical Operator

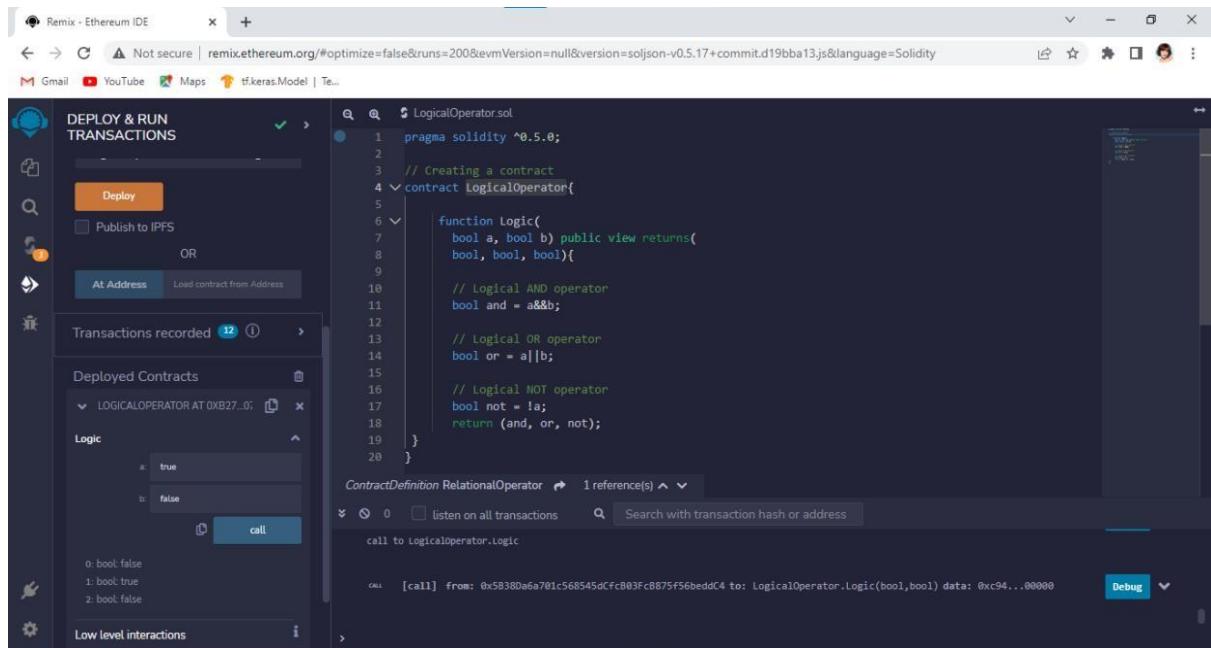
Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main editor area contains the code for "LogicalOperator.sol". The code defines a contract named "LogicalOperator" with a single function "Logic". This function takes two boolean parameters "a" and "b", and returns three boolean values: "bool and", "bool or", and "bool not". The "not" operation is implemented using the NOT operator (~). The Remix interface also shows a "ContractDefinition RelationalOperator" section and a transaction history at the bottom.

```
pragma solidity ^0.5.0;
// Creating a contract
contract LogicalOperator{
    function Logic(
        bool a, bool b) public view returns(
        bool, bool, bool){
        // Logical AND operator
        bool and = a&b;
        // Logical OR operator
        bool or = a||b;
        // Logical NOT operator
        bool not = !a;
        return (and, or, not);
    }
}
```

Output:



The screenshot shows the Remix Ethereum IDE interface in the "DEPLOY & RUN TRANSACTIONS" mode. It displays the deployed contract "LOGICALOPERATOR AT 0XB27..." with its source code. Below the code, there is a "Logic" section where inputs "a" and "b" are set to "true" and "false" respectively. A "call" button is present, and the transaction history shows the result of the call: "0: bool: false", "1: bool: true", and "2: bool: false". The Remix interface also shows a "ContractDefinition RelationalOperator" section and a transaction history at the bottom.

```
pragma solidity ^0.5.0;
// Creating a contract
contract LogicalOperator{
    function Logic(
        bool a, bool b) public view returns(
        bool, bool, bool){
        // Logical AND operator
        bool and = a&b;
        // Logical OR operator
        bool or = a||b;
        // Logical NOT operator
        bool not = !a;
        return (and, or, not);
    }
}
```

Bitwise Operator

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files, with `BitwiseOperator.sol` currently selected. The main code editor window displays the following Solidity code:

```
pragma solidity ^0.5.0;
contract BitwiseOperator {
    uint16 public a = 20;
    uint16 public b = 50;

    uint16 public and = a & b;
    uint16 public or = a | b;
    uint16 public xor = a ^ b;
    uint16 public leftshift = a << b;
    uint16 public rightshift = a >> b;
    uint16 public not = ~a;
}
```

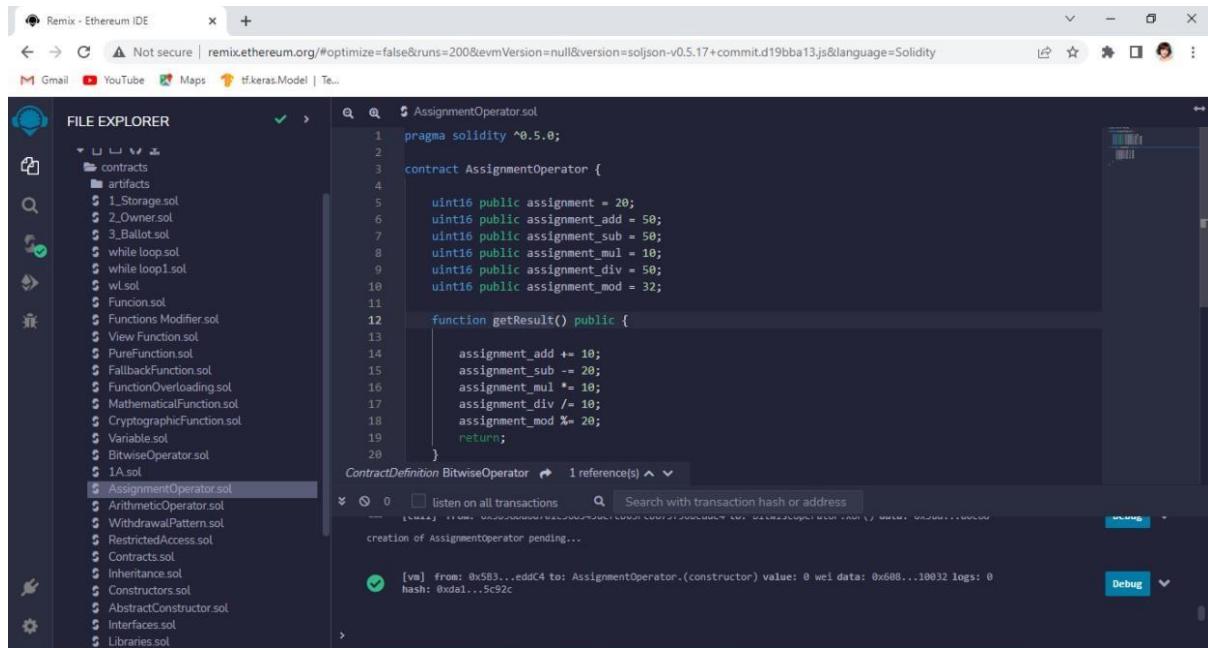
Output:

The screenshot shows the Remix Ethereum IDE interface with the DEPLOY & RUN TRANSACTIONS sidebar open. Under the `BITWISEOPERATOR AT 0x...9e` section, the available functions are listed: `a`, `and`, `b`, `leftshift`, `not`, `or`, `rightshift`, and `xor`. The main code editor window shows the same Solidity code as the previous screenshot. In the bottom right corner, the terminal output shows the deployment and constructor call:

```
[vm] from: 0x583...eddC4 to: LogicalOperator.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x5df...62867 Debug
```

Assignment Operator

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the 'FILE EXPLORER' showing a directory of Solidity files. The main area is the code editor with the file 'AssignmentOperator.sol' open. The code defines a contract with various assignment operators and a getResult() function. A transaction log is visible at the bottom.

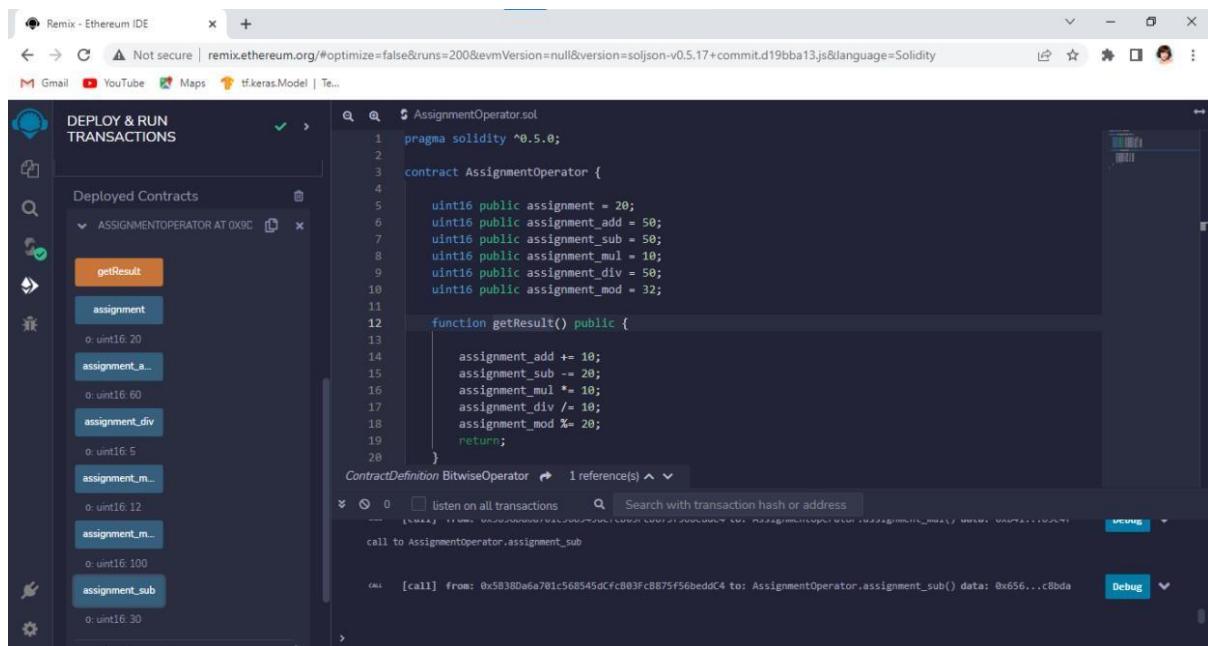
```
pragma solidity ^0.5.0;

contract AssignmentOperator {

    uint16 public assignment = 20;
    uint16 public assignment_add = 50;
    uint16 public assignment_sub = 50;
    uint16 public assignment_mul = 10;
    uint16 public assignment_div = 10;
    uint16 public assignment_mod = 32;

    function getResult() public {
        assignment_add += 10;
        assignment_sub -= 20;
        assignment_mul *= 10;
        assignment_div /= 10;
        assignment_mod %= 20;
        return;
    }
}
```

Output:



The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. It displays the deployed contract 'ASSIGNMENTOPERATOR AT 0x9C...' with several functions listed: getResult, assignment, assignment_a..., assignment_div, assignment_m..., assignment_m..., and assignment_sub. The code editor on the right shows the same 'AssignmentOperator.sol' code as the previous screenshot. Transaction logs are visible at the bottom.

```
pragma solidity ^0.5.0;

contract AssignmentOperator {

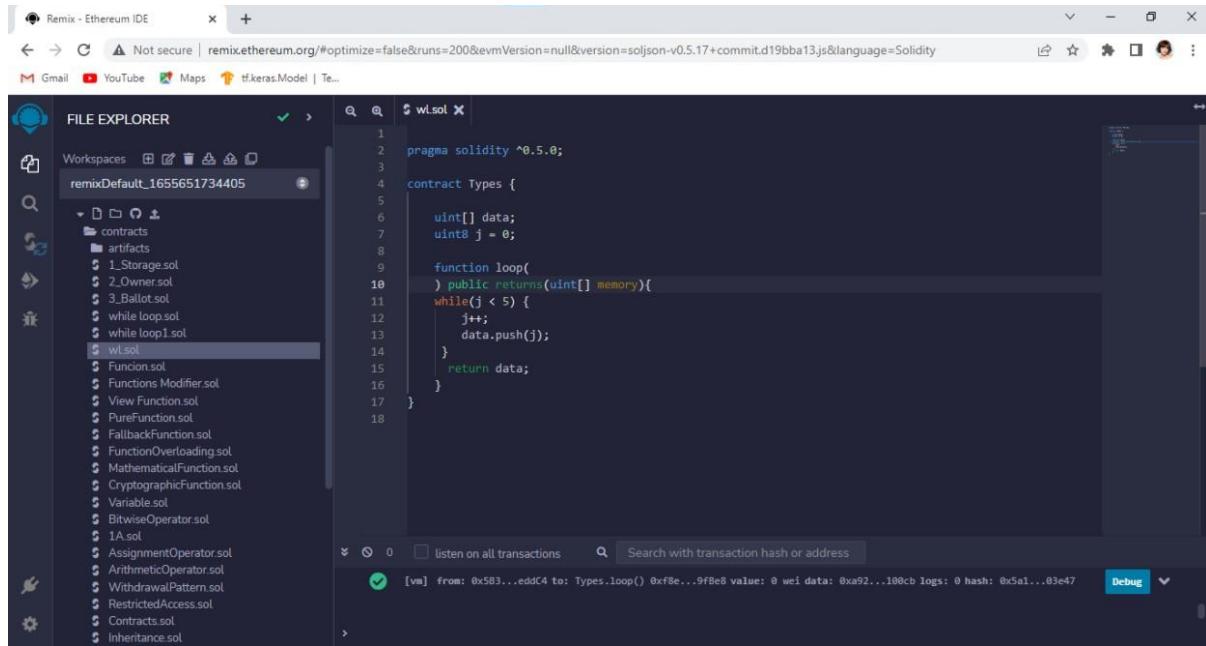
    uint16 public assignment = 20;
    uint16 public assignment_add = 50;
    uint16 public assignment_sub = 50;
    uint16 public assignment_mul = 10;
    uint16 public assignment_div = 10;
    uint16 public assignment_mod = 32;

    function getResult() public {
        assignment_add += 10;
        assignment_sub -= 20;
        assignment_mul *= 10;
        assignment_div /= 10;
        assignment_mod %= 20;
        return;
    }
}
```

Loops

While Loop

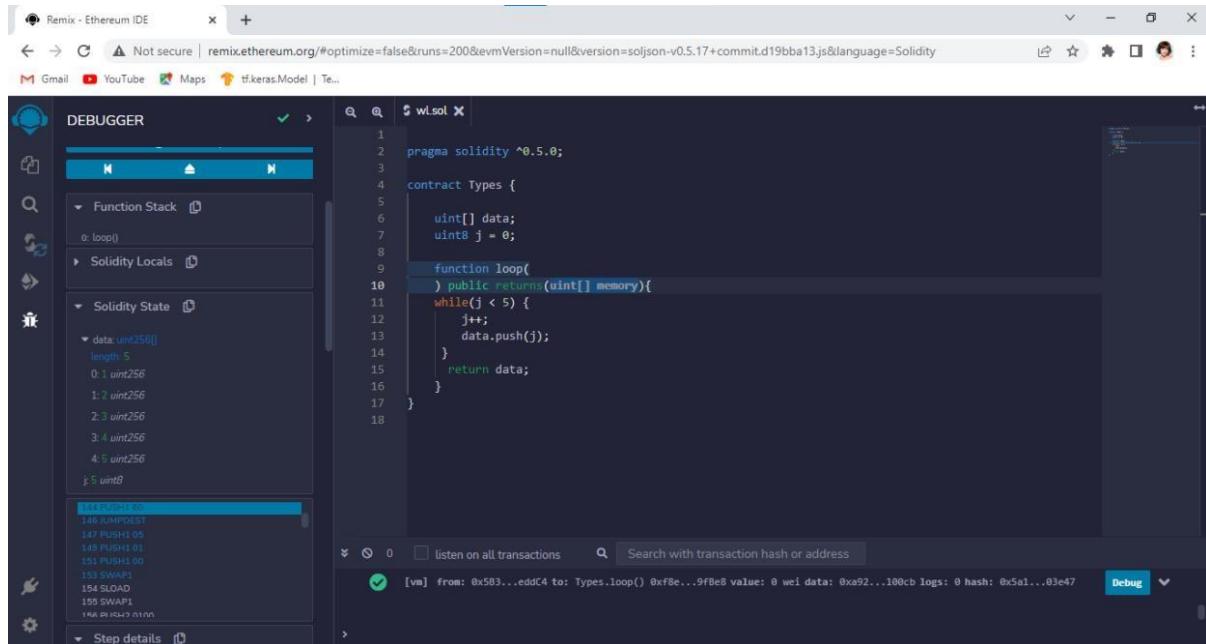
Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the File Explorer, displaying a workspace named "remixDefault_1655651734405" containing several Solidity files: Storage.sol, Owner.sol, Ballot.sol, whileloop.sol, whileloop1.sol, wl.sol, Funcion.sol, Functions.Modifier.sol, ViewFunction.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, and Inheritance.sol. The file "wl.sol" is currently selected and shown in the main code editor area. The code defines a contract "Types" with a function "loop" that returns an array of uints. The function uses a while loop to fill an array "data" with values from 0 to 4. The right side of the interface includes a debugger toolbar with buttons for step operations, a transaction list, and a search bar.

```
pragma solidity ^0.5.0;
contract Types {
    uint[] data;
    uint8 j = 0;
    function loop()
        public
        returns(uint[] memory)
    {
        while(j < 5) {
            j++;
            data.push(j);
        }
        return data;
    }
}
```

Output:

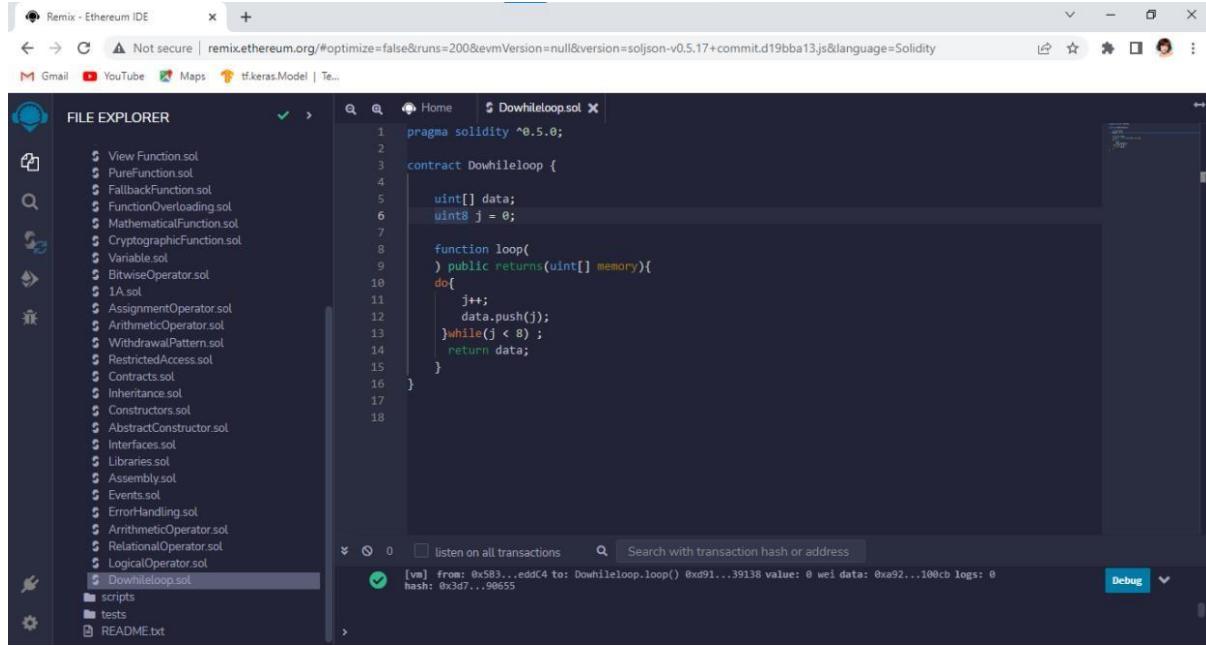


The screenshot shows the Remix Ethereum IDE with the debugger active. The left sidebar now displays the "DEBUGGER" section, which includes tabs for "Function Stack" (showing "loop()"), "Solidity Locals" (showing local variables "data" and "j" with their respective values), and "Solidity State" (showing the state of the "data" array with elements 0, 1, 2, 3, 4). The main code editor area shows the same Solidity code as before. The bottom of the interface features a detailed assembly log window showing the execution steps, starting with a series of PUSH1 and SWAP1 instructions followed by a loop of POP and SWAP operations. The assembly log ends with a RETURNPOLY instruction.

```
140 DUP1PUSH1
141 DUP1PUSH1
142 DUP1PUSH1
143 DUP1PUSH1
144 DUP1PUSH1
145 DUP1PUSH1
146 DUP1PUSH1
147 DUP1PUSH1
148 DUP1PUSH1
149 DUP1PUSH1
150 DUP1PUSH1
151 DUP1PUSH1
152 DUP1PUSH1
153 SWAP1
154 SLOAD
155 SWAP1
156 PUSH4 0x00000000
157 RETURNPOLY
```

Dowhile loop

Code:



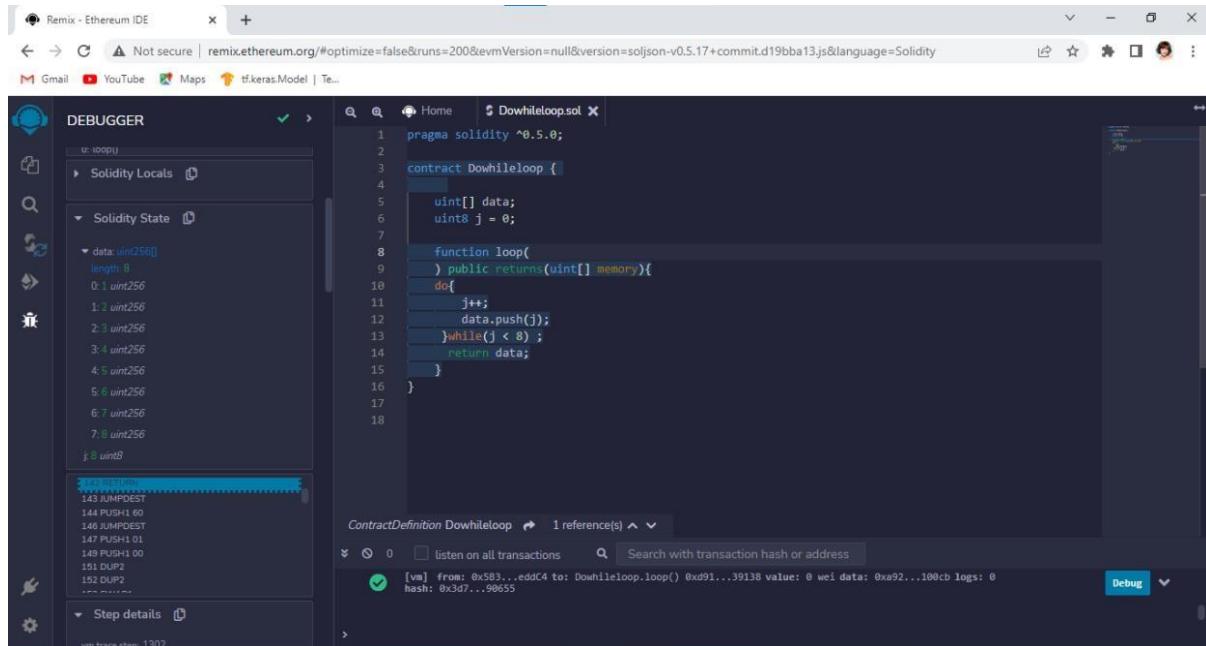
The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the File Explorer, listing various Solidity files. The main area displays the Solidity code for a contract named `Dowhileloop`:

```
pragma solidity ^0.5.0;
contract Dowhileloop {
    uint[] data;
    uint8 j = 0;

    function loop()
        public returns(uint[] memory){
        do{
            j++;
            data.push(j);
        }while(j < 8);
        return data;
    }
}
```

The status bar at the bottom shows a transaction log: [vm] from: 0x5B3...edc4 to: Dowhileloop.loop() 0xd91...39138 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x3d7...90655

Output:



The screenshot shows the Remix Ethereum IDE interface with the Debugger tab selected. The left sidebar shows the Solidity Locals and Solidity State. The Solidity State section shows the variable `j` with a value of 8, and the `data` array with values from 0 to 7.

The main area displays the same Solidity code as the previous screenshot. The status bar at the bottom shows a transaction log: [vm] from: 0x5B3...edc4 to: Dowhileloop.loop() 0xd91...39138 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x3d7...90655

For loop

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the 'FILE EXPLORER' containing various Solidity files. The main area is the 'EDITOR' showing the code for 'ForLoop.sol':

```
pragma solidity ^0.5.0;
contract ForLoop {
    uint[] data;
    function loop() public returns(uint[] memory){
        for(uint i=0; i<4; i++){
            data.push(i);
        }
        return data;
    }
}
```

The status bar at the bottom indicates a transaction from address 0x5B3...eddc4 to ForLoop.loop() with value 0 wei, data 0xa92...100cb, logs 0 hash 0x724...0add1.

Output:

The screenshot shows the Remix Ethereum IDE interface with the 'DEBUGGER' tab selected. The left sidebar shows the 'FUNCTION STACK' with 'loop()' as the current frame. The main area is the 'EDITOR' showing the same Solidity code as before. The status bar at the bottom indicates a transaction from address 0x5B3...eddc4 to ForLoop.loop() with value 0 wei, data 0xa92...100cb, logs 0 hash 0x724...0add1.

Decision making

If statement

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the File Explorer, listing various Solidity files. The main editor area contains the following Solidity code:

```
pragma solidity ^0.5.0;
contract IfStatement {
    uint i = 20;
    function decision_making() public returns(bool){
        if(i>20){
            return true;
        }
    }
}
```

The bottom pane shows two transaction logs from the VM:

- [vm] from: 0x5B3...eddC4 to: ForLoop.loop() 0xf8e...9f8e8 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x724...0add1 Debug
- [vm] from: 0x5B3...eddC4 to: IfStatement.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x484...7d5c1 Debug

Output:

The screenshot shows the Remix Ethereum IDE with the Debugger tab selected. The left sidebar includes a Debugger Configuration section with a checked checkbox for "Use generated sources (Solidity >= v0.7.2)". The main editor area shows the same Solidity code as before. The bottom pane displays the debugger interface with the following details:

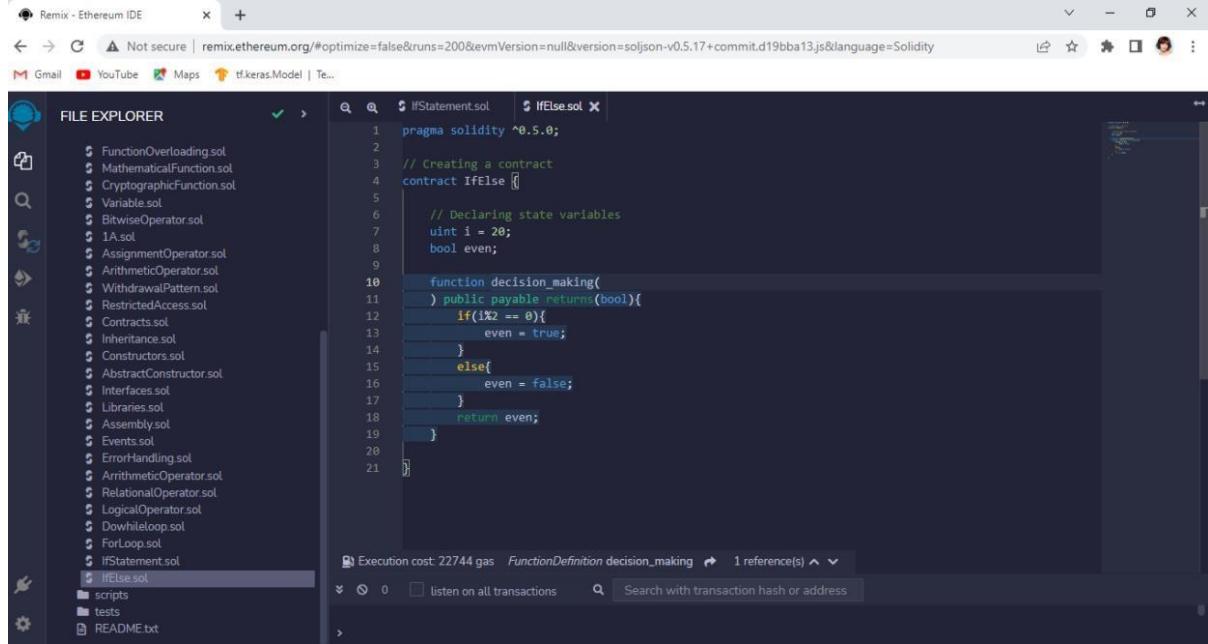
- ContractDefinition IfStatement 1 reference(s)
- input: 0x887...3aF24
- decoded input: {}
- decoded output: { "": "bool: false" }
- logs: []
- val: 0 wei

The assembly code for the current state is also visible on the left:

```
077 JUMPDEST
078 PUSH1 00
080 PUSH1 14
082 PUSH1 00
084 SLOAD
085 LT
086 JUMPI
```

If...else statement

Code:



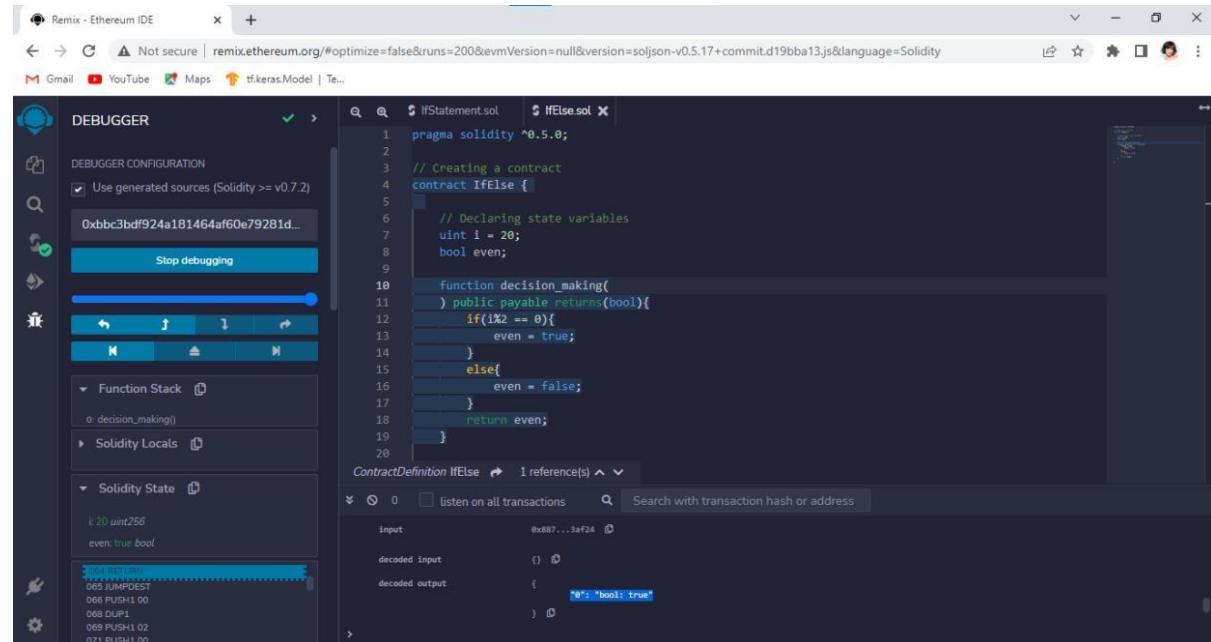
The screenshot shows the Remix Ethereum IDE interface. The left sidebar is labeled 'FILE EXPLORER' and lists several Solidity files. In the main editor area, there are two tabs: '\$ IfStatement.sol' and '\$ IfElse.sol'. The '\$ IfElse.sol' tab is active, displaying the following Solidity code:

```
pragma solidity ^0.5.0;
// Creating a contract
contract IfElse {
    // Declaring state variables
    uint i = 20;
    bool even;

    function decision_making()
        public payable returns(bool){
        if(i%2 == 0){
            even = true;
        }
        else{
            even = false;
        }
        return even;
    }
}
```

The status bar at the bottom indicates an execution cost of 22744 gas for the 'FunctionDefinition decision_making' function, with 1 reference(s).

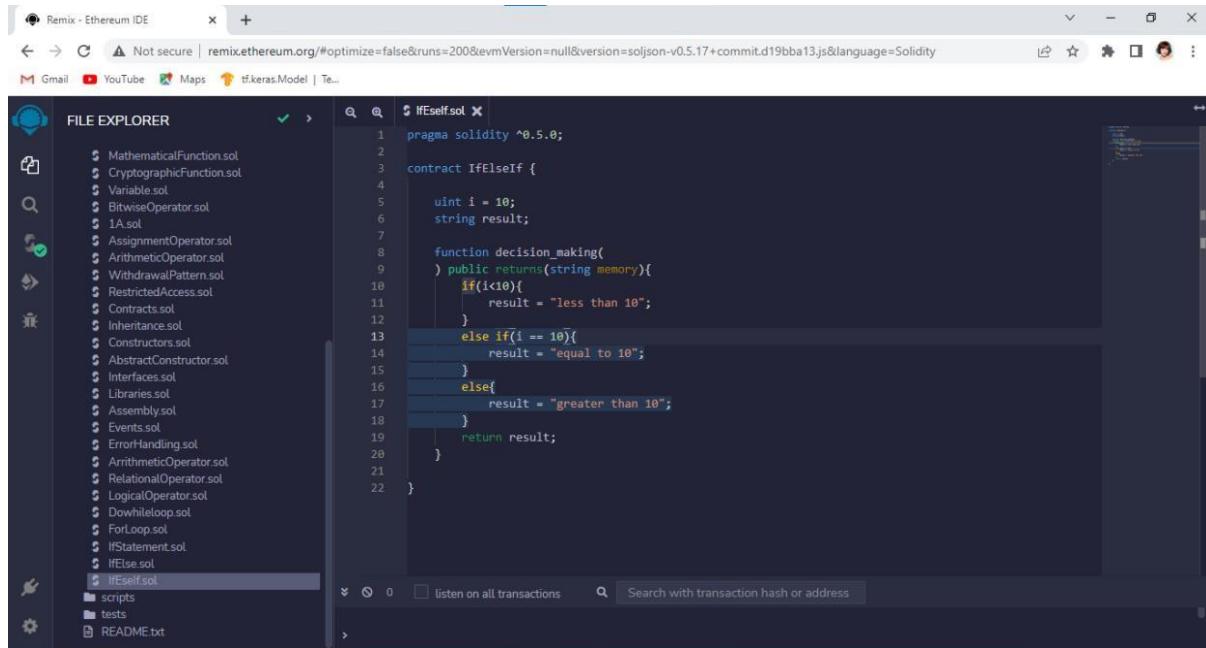
Output:



The screenshot shows the Remix Ethereum IDE interface with the 'DEBUGGER' tab selected. The left sidebar includes a 'DEBUGGER CONFIGURATION' section with a checked checkbox for 'Use generated sources (Solidity >= v0.7.2)'. The main editor area shows the same Solidity code as the previous screenshot. The debugger sidebar displays the 'Function Stack' (with one entry: 'decision_making()'), 'Solidity Locals' (showing 'even: true bool'), and 'Solidity State' (showing memory dump details). The status bar at the bottom indicates an input of 0x887...3af24, a decoded input of 0, and a decoded output of { "0": "bool: true" }.

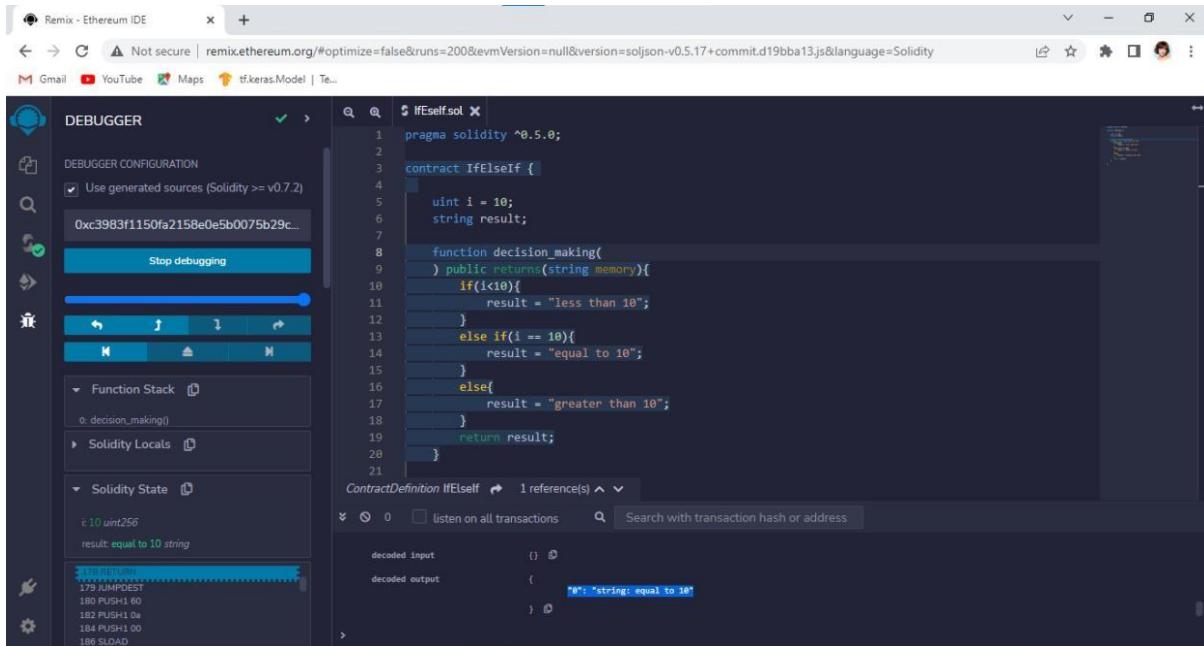
If...else if...else statement

Code:



```
pragma solidity ^0.5.0;
contract IfElseIf {
    uint i = 10;
    string result;
    function decision_making() public returns(string memory){
        if(i<10){
            result = "less than 10";
        }
        else if(i == 10){
            result = "equal to 10";
        }
        else{
            result = "greater than 10";
        }
        return result;
    }
}
```

Output:



The screenshot shows the Remix IDE's DEBUGGER tab. The code editor still contains the same Solidity contract. The DEBUGGER panel on the left shows a 'Function Stack' with 'o. decision_making()' at the top. The bottom section shows the 'Solidity State' with the variable 'result' set to 'equal to 10' and its type as 'string'. The assembly code pane at the bottom shows the EVM assembly for the function, starting with 100 JUMPDEST and ending with 186 SLOAD.

String

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER tab is active, displaying a list of Solidity source files. In the center, the CODE tab shows the Solidity code for a contract named `StringExample`:

```
pragma solidity ^0.5.0;
contract StringExample {
    string text;
    function setText () public returns (string memory)
    {
        text = "Hello World";
        return text;
    }
}
```

The code defines a contract with a single function `setText()` that sets the `text` variable to "Hello World" and returns it.

Output:

The screenshot shows the Remix Ethereum IDE interface with the DEBUGGER tab active. A transaction has been deployed, and the output of the `setText()` function is shown in the Solidity State panel:

text	string
0x02b5a78ec919c54377899e8881ef...	Hello World

The Solidity State panel also displays the assembly code for the function call:

```
133 PUSH1 0x02
134 DUP1
135 CALLDATACOPY
136 POP
137 JUMPDEST
138 PUSH1 40
139 DUP1
140 MLOAD
141 DUP1
142 PUSH1 40
143 AND
```

Array

Code:

The screenshot shows the Ethereum IDE interface. The left sidebar is the File Explorer, listing various Solidity files. The main editor window contains the following Solidity code:

```
// Creating a contract
contract ArrayExample {
    uint[] data
    | = [10, 20, 30, 40, 50];
    int[] data1;
    function dynamic_array() public returns(
        uint[] memory, int[] memory){
        data1
        | = [int(-60), 70, -80, 90, -100, -120, 140];
        return (data, data1);
    }
}
```

The status bar at the bottom shows the date and time: 7/24/2022 3:34 PM.

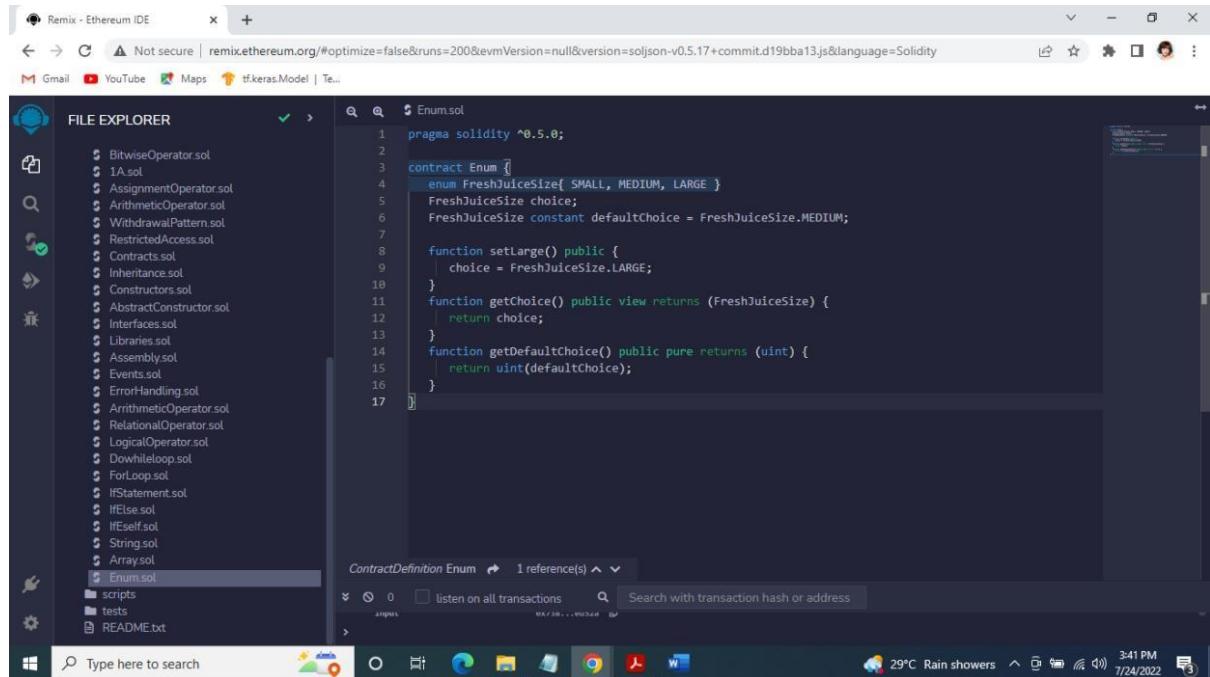
Output:

The screenshot shows the Ethereum IDE interface with the Debugger tab selected. The left sidebar includes a Debugger Configuration section with a checked checkbox for "Use generated sources (Solidity >= v0.7.2)". The main editor window shows the same Solidity code as before. The bottom pane displays the debugger output, including the execution cost, input, decoded input, decoded output, and logs. The decoded output shows the arrays data and data1 with their respective values.

Decoded Output
"0": "uint256[]": 10, 20, 30, 40, 50", "1": "int256[]": -60, 70, -80, 90, -100, -120, 140"

Enums

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the File Explorer, listing various Solidity files. The main editor area contains the following Solidity code:

```
pragma solidity ^0.5.0;

contract Enum {
    enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
    FreshJuiceSize choice;
    FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;

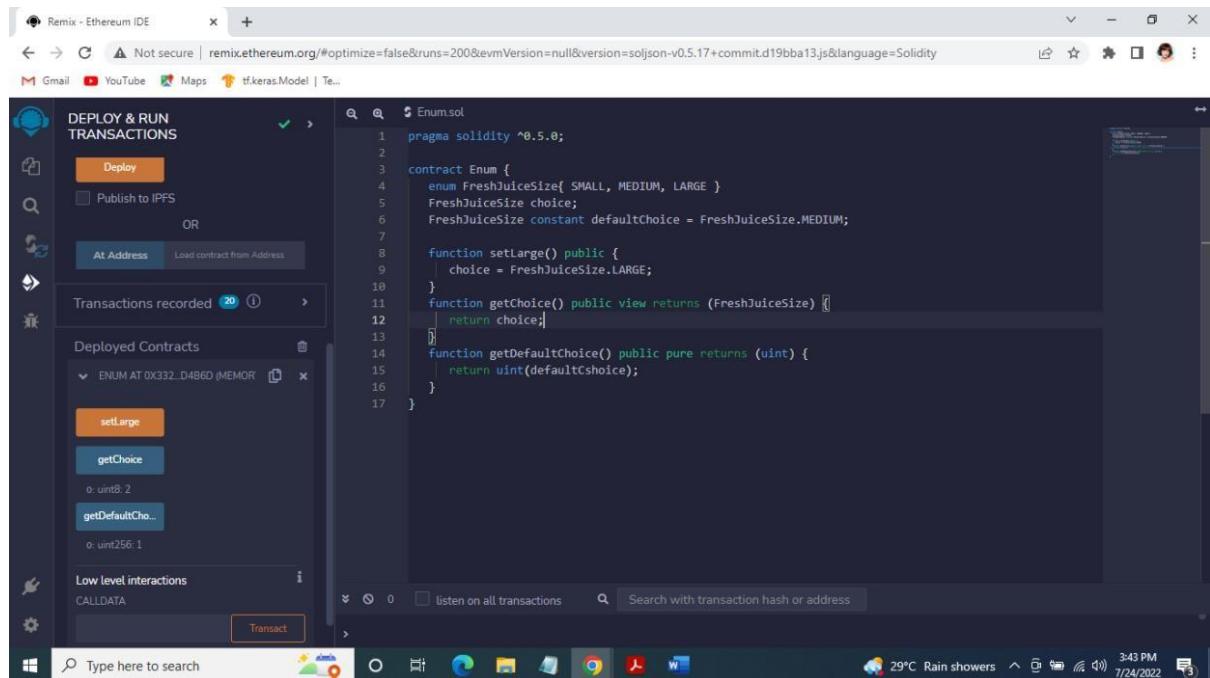
    function setLarge() public {
        choice = FreshJuiceSize.LARGE;
    }

    function getChoice() public view returns (FreshJuiceSize) {
        return choice;
    }

    function getDefaultChoice() public pure returns (uint) {
        return uint(defaultChoice);
    }
}
```

The code defines an enum named FreshJuiceSize with three values: SMALL, MEDIUM, and LARGE. It includes a constant defaultChoice set to MEDIUM. There are three functions: setLarge(), getChoice(), and getDefaultChoice(). The IDE interface includes a status bar at the bottom showing weather and system information.

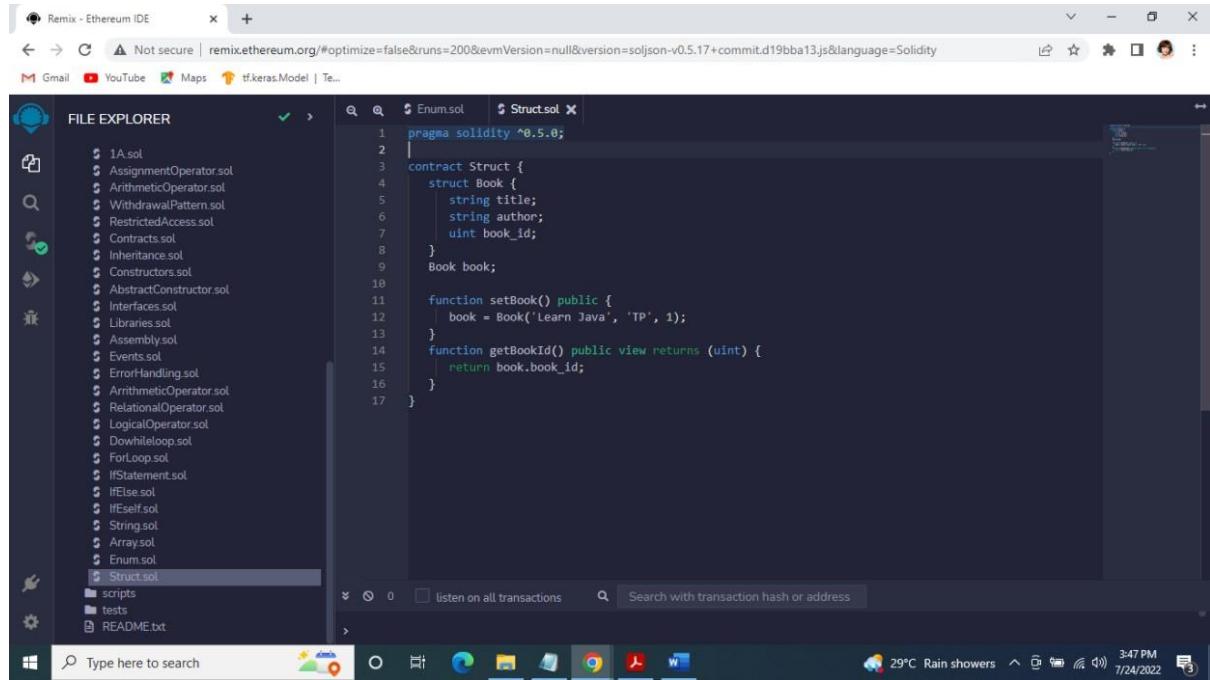
Output:



The screenshot shows the Remix Ethereum IDE interface after deployment. The left sidebar now displays the Deploy & Run Transactions section, which includes options like Deploy, Publish to IPFS, or At Address. Under Deployed Contracts, it lists "ENUM AT 0X332-D486D (MEMORY)". Below this, there are three buttons for interacting with the contract: "setLarge", "getChoice", and "getDefaultCho...". The "getChoice" button is currently highlighted. The main editor area shows the same Solidity code as the previous screenshot. The status bar at the bottom remains the same.

Struct

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree with various Solidity files like 1A.sol, AssignmentOperator.sol, etc. The central code editor window contains the following Solidity code:

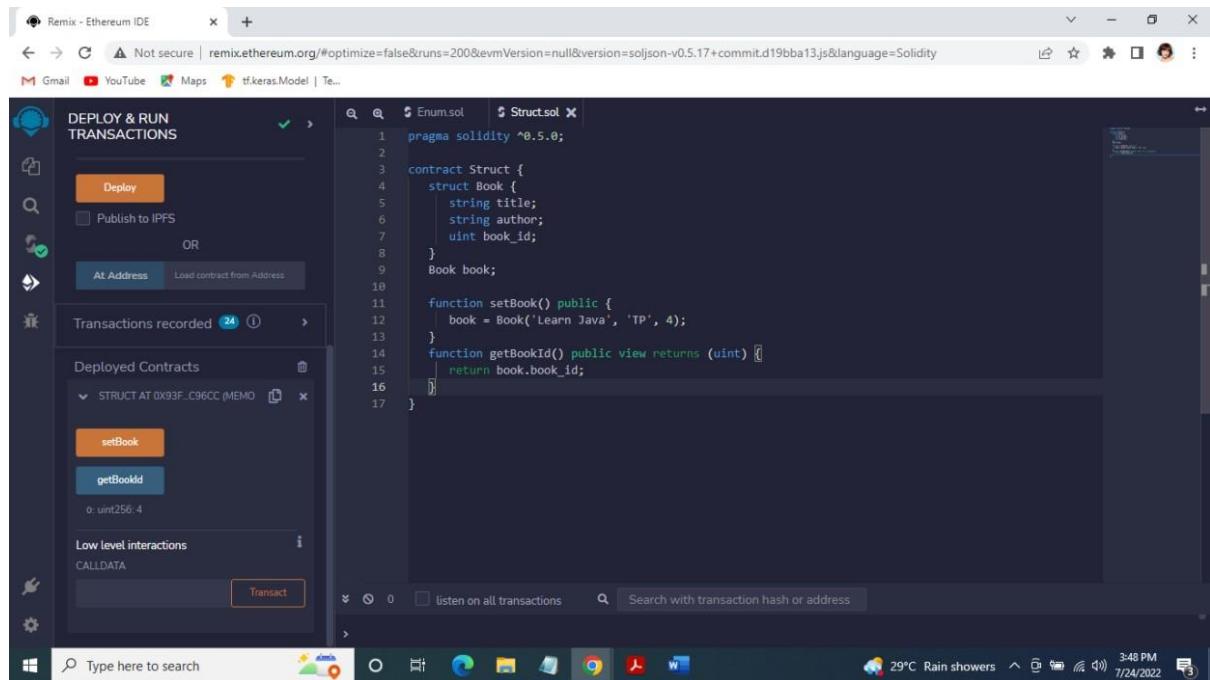
```
pragma solidity ^0.5.0;

contract Struct {
    struct Book {
        string title;
        string author;
        uint book_id;
    }
    Book book;

    function setBook() public {
        book = Book('Learn Java', 'TP', 1);
    }

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

Output:



The screenshot shows the Remix Ethereum IDE interface after deployment. The left sidebar now includes a "Deploy & Run Transactions" section with a "Deploy" button. The code editor window remains the same as in the previous screenshot. The bottom-left panel shows the "Transactions recorded" section with 24 entries and a "Deployed Contracts" section listing "STRUCT AT 0X93F..C96CC (MEMO)". Below the code editor, there are buttons for "setBook" and "getBookId". The status bar at the bottom indicates "3:47 PM 29°C Rain showers 7/24/2022".

Mapping

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the FILE EXPLORER, listing various Solidity files. The main editor window contains the following Solidity code:

```
pragma solidity ^0.4.18;

contract mapping_example {
    struct student {
        string name;
        string subject;
        uint8 marks;
    }

    mapping (
        address => student) result;
    address[] public student_result;

    function adding_values() public {
        var student
            = result[0xDEE7796E89C82C36BAdd1375076f39D69FafE252];

        student.name = "John";
        student.subject = "Chemistry";
        student.marks = 88;
        student_result.push(
            0xDEE7796E89C82C36BAdd1375076f39D69FafE252) -1;
    }
}
```

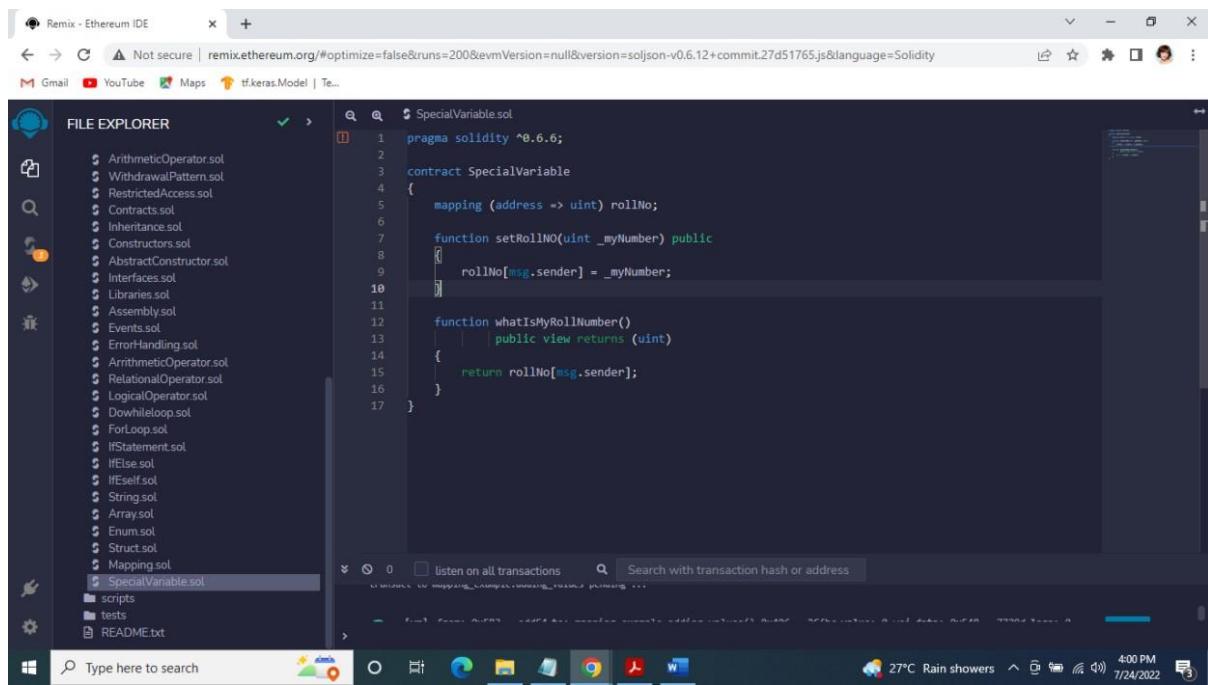
The code defines a contract named `mapping_example` with a `student` struct and a mapping from `address` to `student`. It includes a `adding_values` function that adds a new entry to the mapping.

Output:

The screenshot shows the Remix Ethereum IDE interface with the DEBUGGER tab selected. The left sidebar is the DEBUGGER, showing the Function Stack, Solidity Locals, and Solidity State. The Solidity State panel displays the state of the `result` mapping, showing an entry for address `0xDEE7796E89C82C36BAdd1375076f39D69FafE252` with values `name: John`, `subject: Chemistry`, and `marks: 88`. The main editor window shows the same Solidity code as above. The bottom status bar indicates the date and time as 7/24/2022 3:55 PM.

Special Variable

Code:



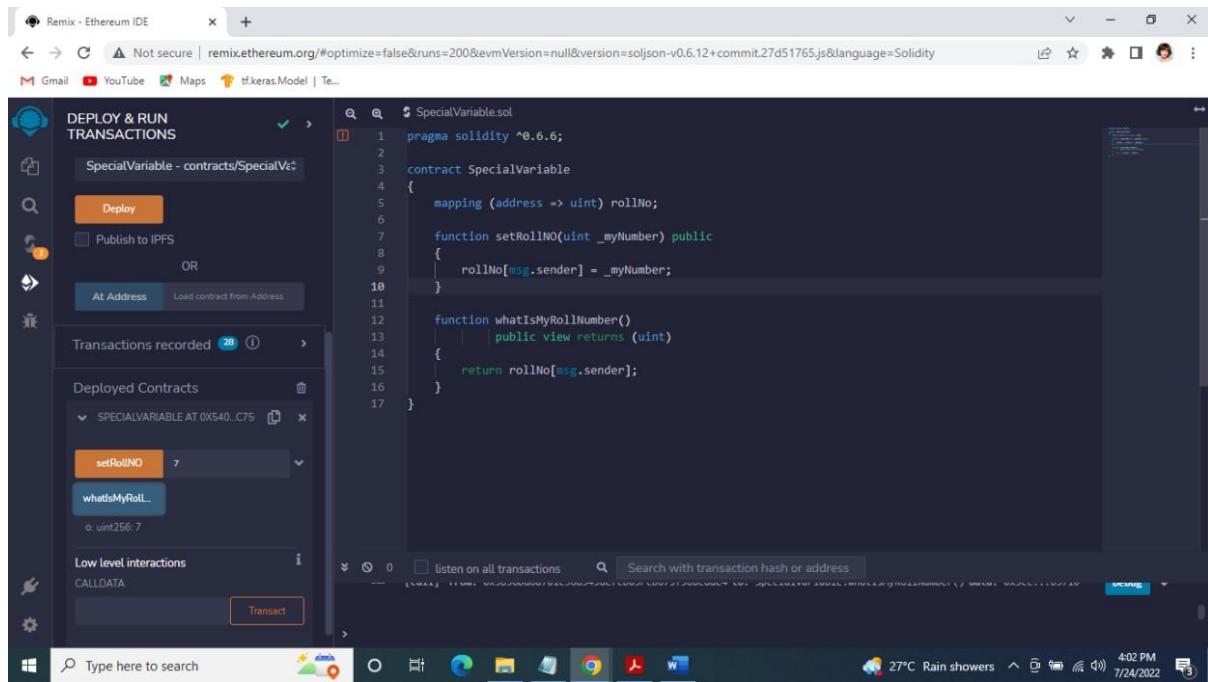
```
pragma solidity ^0.6.6;

contract SpecialVariable
{
    mapping (address => uint) rollNo;

    function setRollNO(uint _myNumber) public
    {
        rollNo[msg.sender] = _myNumber;
    }

    function whatIsMyRollNumber()
    | | | public view returns (uint)
    {
        return rollNo[msg.sender];
    }
}
```

Output:



The screenshot shows the Remix Ethereum IDE interface with the "Deploy & Run Transactions" tab selected. The code for `SpecialVariable.sol` is displayed in the center. Below the code, the "Transactions recorded" section shows 28 entries. Under "Deployed Contracts", there is a list titled "SPECIALVARIABLE AT 0x540...C75" which includes the `setRollNO` and `whatIsMyRoll...` functions. The `setRollNO` function has a value of 7. The `whatIsMyRoll...` function has a value of 0. uint256: 7.

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

Function:

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays the 'FILE EXPLORERS' panel with a list of contracts and scripts. The 'Function.sol' file is currently selected. The main code editor window contains the following Solidity code:

```
pragma solidity ^0.5.0;

contract Function{
    function getResult() public view returns (uint product, uint sum){
        uint a=11;
        uint b=20;
        product=a*b;
        sum=a+b;
    }
}
```

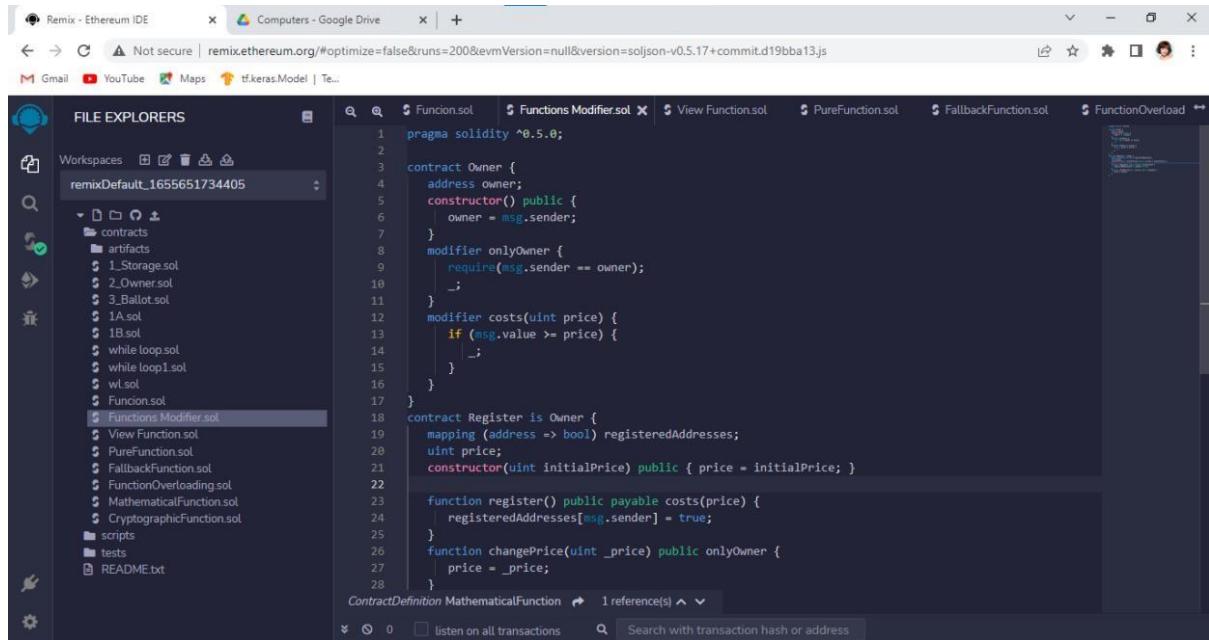
The status bar at the bottom indicates a transaction from a local VM, and there is a 'Debug' button.

Output:

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' panel active. The left sidebar displays the 'FILE EXPLORERS' panel with a list of contracts and scripts. The 'Function.sol' file is currently selected. The main code editor window contains the same Solidity code as before. The status bar at the bottom indicates a transaction call from a local VM, and there is a 'Debug' button.

Functions Modifiers

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORERS" and lists several Solidity files: Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, and FunctionOverload. The "Functions Modifier.sol" file is currently selected. The main editor area displays the following Solidity code:

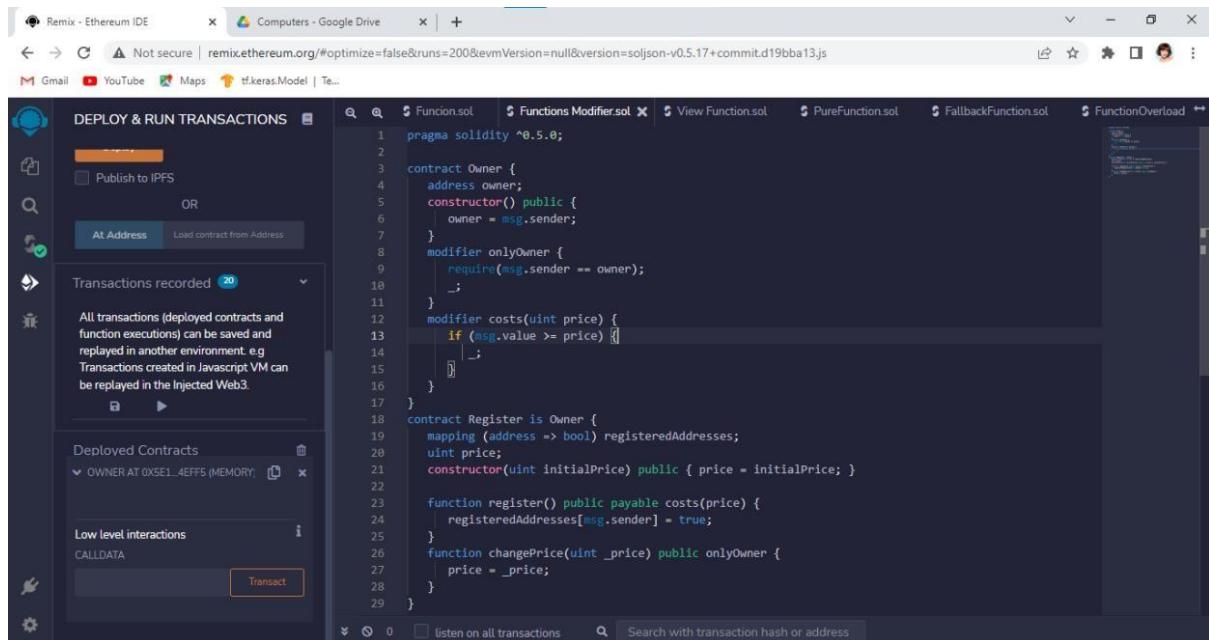
```
pragma solidity ^0.5.0;

contract Owner {
    address owner;
    constructor() public {
        owner = msg.sender;
    }
    modifier onlyOwner {
        require(msg.sender == owner);
    }
    modifier costs(uint price) {
        if (msg.value >= price) {
        }
    }
}

contract Register is Owner {
    mapping (address => bool) registeredAddresses;
    uint price;
    constructor(uint initialPrice) public { price = initialPrice; }

    function register() public payable costs(price) {
        registeredAddresses[msg.sender] = true;
    }
    function changePrice(uint _price) public onlyOwner {
        price = _price;
    }
}
```

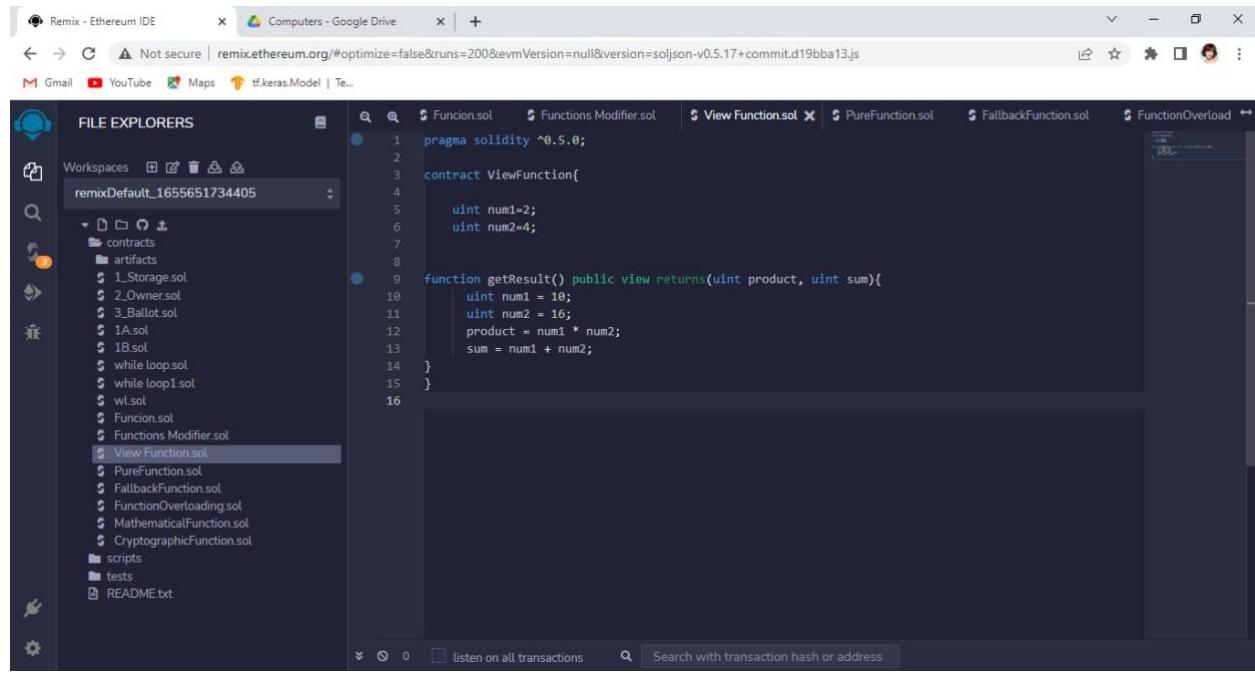
Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active. The left sidebar shows deployment options: "Publish to IPFS" (unchecked), "OR", and "At Address". Below this, it says "Transactions recorded 20" and provides instructions: "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.". The "Deployed Contracts" section shows "OWNER AT 0xSEL1_4EFF5 (MEMORY)". The "Low level interactions" section has a "CALLDATA" button and a "Transact" button. The main editor area shows the same Solidity code as the previous screenshot. The status bar at the bottom indicates "0" transactions and a search bar for "Search with transaction hash or address".

View function

Code:



The screenshot shows the Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and the URL "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". Below the title bar is a toolbar with icons for back, forward, search, and other functions. The main window is divided into several panes. On the left is the "FILE EXPLORERS" pane, which shows a workspace named "remixDefault_1655651734405". Inside this workspace, there is a "contracts" folder containing several Solidity files: 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, 1A.sol, 1B.sol, while_loop.sol, while_loop1.sol, wl.sol, Funcion.sol, Functions Modifier.sol, View Function.sol (which is currently selected), PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, scripts, tests, and README.txt. The central pane displays the Solidity code for the "View Function" contract. The code defines a public view function "getResult" that calculates the product and sum of two uint variables, num1 and num2, initialized to 2 and 4 respectively.

```
pragma solidity ^0.5.0;

contract ViewFunction{
    uint num1=2;
    uint num2=4;

    function getResult() public view returns(uint product, uint sum){
        uint num1 = 10;
        uint num2 = 16;
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar includes tabs for Remix - Ethereum IDE, Computers - Google Drive, and a browser tab for remix.ethereum.org. Below the tabs, there are links for Gmail, YouTube, Maps, and tf.keras.Model.

The main area is titled "DEPLOY & RUN TRANSACTIONS". It features a sidebar with icons for deploying contracts, running transactions, and viewing logs. A message in the sidebar states: "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."

The central workspace displays the Solidity code for the `ViewFunction` contract:

```
pragma solidity ^0.5.0;
contract ViewFunction{
    uint num1=2;
    uint num2=4;

    function getResult() public view returns(uint product, uint sum){
        uint num1 = 10;
        uint num2 = 16;
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

Below the code, the "Deployed Contracts" section shows the deployed contract `VIEWFUNCTION AT 0xE28...4157A`. It contains a button labeled "getResult" which, when clicked, displays the results: "0: uint256: product 160" and "1: uint256: sum 26".

The bottom right corner of the interface has a search bar with the placeholder "Search with transaction hash or address".

Pure function

Code:

The screenshot shows the Remix Ethereum IDE interface, similar to the previous one but with a different file structure. The top navigation bar and sidebar are identical.

The main area is titled "FILE EXPLORERS". The left sidebar shows a tree view of the project structure under "remixDefault_1655651734405". The "contracts" folder contains several files: 1.Storage.sol, 2.Owner.sol, 3.Ballot.sol, 1A.sol, 1B.sol, while loop.sol, while loop1.sol, wLsol, Functions.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, and CryptographicFunction.sol. The "scripts" and "tests" folders are also listed.

The central workspace displays the Solidity code for the `PureFunction` contract:

```
pragma solidity ^0.5.0;
contract PureFunction {
    function getResult() public pure returns(uint product, uint sum){
        uint num1 = 2;
        uint num2 = 10;
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

The bottom right corner of the interface has a search bar with the placeholder "Search with transaction hash or address".

Ouput:

The screenshot shows the Ethereum IDE interface. In the center, there is a code editor with Solidity code for a `PureFunction` contract. The code defines a `getResult()` function that returns the product of two numbers. Below the code editor, the "Deploy & Run Transactions" panel is visible, showing a deployed contract at address `0x1C9...2B4BD`. Under the "getResults" section, it shows the output: `0: uint256: product 20` and `1: uint256: sum 12`. At the bottom of the interface, a transaction log is displayed: `[vm] from: 0x583...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x0e9...6a4d6`.

```
pragma solidity ^0.5.0;

contract PureFunction {
    function getResult() public pure returns(uint product, uint sum){
        uint num1 = 2;
        uint num2 = 10;
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

Fallback function

Code:

The screenshot shows the Ethereum IDE interface with the "FILE EXPLORERS" panel open, displaying a workspace named `remixDefault_1655651734405`. The "contracts" folder contains several Solidity files, including `Storage.sol`, `Owner.sol`, `Ballot.sol`, `A.sol`, `B.sol`, `while loop.sol`, `wLsol.sol`, `Funcion.sol`, `Functions Modifier.sol`, `View Function.sol`, `PureFunction.sol`, `FallbackFunction.sol`, `FunctionOverloading.sol`, `MathematicalFunction.sol`, and `CryptographicFunction.sol`. The `FallbackFunction.sol` file is currently selected and shown in the code editor. The code defines a `FallbackFunction` contract with a `x` variable and an external function `external { x = 1; }`. It also includes contracts `Sink` and `Caller`, and functions `callTest` and `callSink`.

```
pragma solidity ^0.5.0;

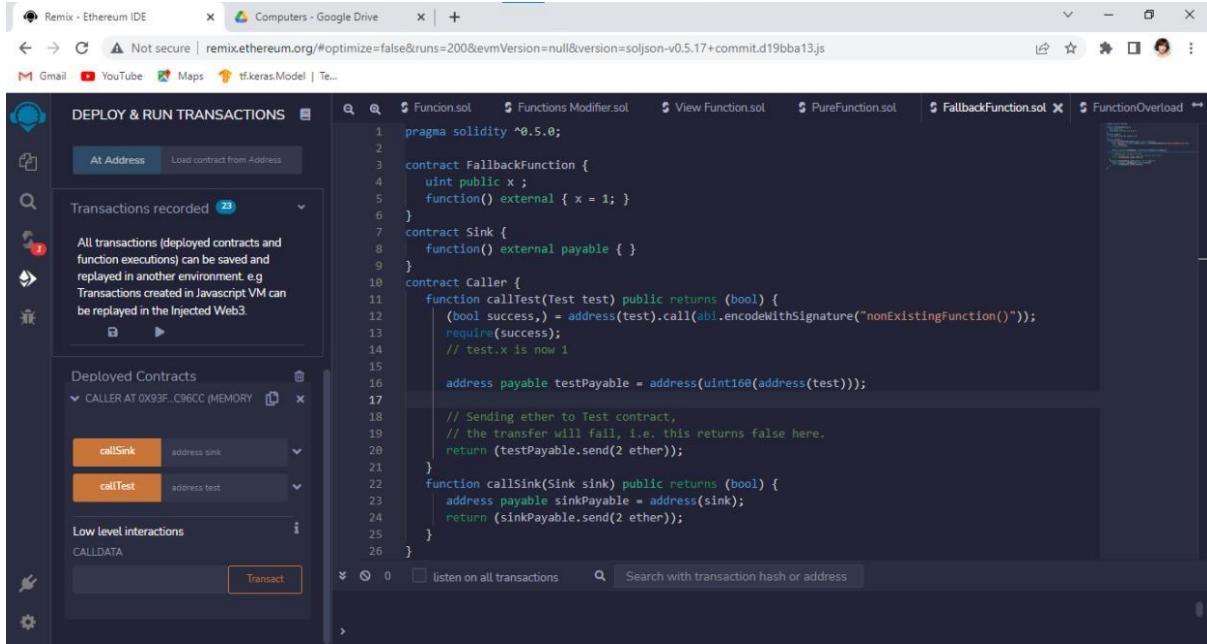
contract FallbackFunction {
    uint public x;
    function() external { x = 1; }
}

contract Sink {
    function() external payable {}
}

contract Caller {
    function callTest(Test test) public returns (bool) {
        (bool success,) = address(test).call(abi.encodeWithSignature("nonExistingFunction()"));
        require(success);
        // test.x is now 1
    }
    address payable testPayable = address(uint160(address(test)));
    // Sending ether to Test contract,
    // the transfer will fail, i.e. this returns false here.
    return (testPayable.send(2 ether));
}

function callSink(Sink sink) public returns (bool) {
    address payable sinkPayable = address(sink);
    return (sinkPayable.send(2 ether));
}
```

Output:



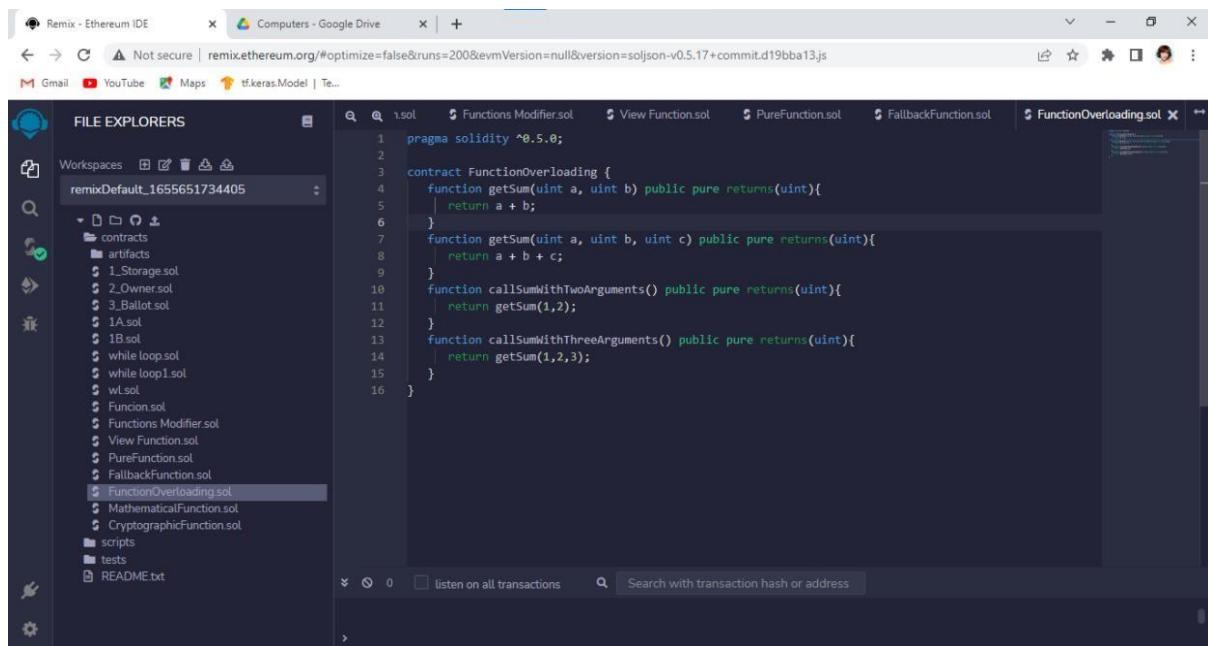
The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays the 'FILE EXPLORERS' section with a workspace named 'remixDefault_165561734405'. Inside this workspace, there are several contracts and scripts listed under 'contracts' and 'scripts'. The main editor area contains the following Solidity code for 'FunctionOverloading.sol':

```
pragma solidity ^0.5.0;

contract FunctionOverloading {
    function getSum(uint a, uint b) public pure returns(uint){
        return a + b;
    }
    function getSum(uint a, uint b, uint c) public pure returns(uint){
        return a + b + c;
    }
    function callSumWithTwoArguments() public pure returns(uint){
        return getSum(1,2);
    }
    function callSumWithThreeArguments() public pure returns(uint){
        return getSum(1,2,3);
    }
}
```

Function Overloading

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays the 'FILE EXPLORERS' section with a workspace named 'remixDefault_165561734405'. Inside this workspace, there are several contracts and scripts listed under 'contracts' and 'scripts'. The main editor area contains the following Solidity code for 'FunctionOverloading.sol':

```
pragma solidity ^0.5.0;

contract FunctionOverloading {
    function getSum(uint a, uint b) public pure returns(uint){
        return a + b;
    }
    function getSum(uint a, uint b, uint c) public pure returns(uint){
        return a + b + c;
    }
    function callSumWithTwoArguments() public pure returns(uint){
        return getSum(1,2);
    }
    function callSumWithThreeArguments() public pure returns(uint){
        return getSum(1,2,3);
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays the 'DEPLOY & RUN TRANSACTIONS' section, which includes a note about saving and replaying transactions. Below this is the 'Deployed Contracts' section, which lists a single contract named 'FUNCTIONOVERLOADING' at address 0x5AB... . The main code editor window contains the following Solidity code:

```
pragma solidity ^0.5.0;

contract FunctionOverloading {
    function getSum(uint a, uint b) public pure returns(uint){
        return a + b;
    }
    function getSum(uint a, uint b, uint c) public pure returns(uint){
        return a + b + c;
    }
    function callSumWithTwoArguments() public pure returns(uint){
        return getSum(1,2);
    }
    function callSumWithThreeArguments() public pure returns(uint){
        return getSum(1,2,3);
    }
}
```

The 'Low level interactions' section shows two transaction examples: 'callSumWithTwoArguments' with value 6 and 'callSumWithThreeArguments' with value 3. The 'getSum' function is highlighted in blue.

Mathematical Function

Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORERS' sidebar open. The sidebar lists various workspace files, including contracts like Storage.sol, Owner.sol, Ballot.sol, 1A.sol, 1B.sol, while loop.sol, while loop1.sol, wLsol, Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, and MathematicalFunction.sol. The 'MathematicalFunction.sol' file is currently selected and highlighted in blue. The main code editor window contains the following Solidity code:

```
pragma solidity ^0.5.0;

contract MathematicalFunction {
    function callAddMod() public pure returns(uint){
        return addmod(14, 15, 13);
    }
    function callMulMod() public pure returns(uint){
        return mulmod(14, 15, 13);
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar includes tabs for "Remix - Ethereum IDE" and "Computers - Google Drive". Below the tabs, there are browser-like controls for back, forward, and search, along with a message indicating "Not secure". The main workspace displays a Solidity code editor with the file "MathematicalFunction.sol" open. The code defines a contract with two functions: `callAddMod` and `callMulMod`, both returning uint256 values. To the left of the code editor is a sidebar titled "DEPLOY & RUN TRANSACTIONS". It shows a summary of "Transactions recorded" (25) and a section for "Deployed Contracts" where "MATHEMATICALFUNCTION AT 0x40f..." is listed. Under this contract, two function calls are shown: `callAddMod` returning 3 and `callMulMod` returning 2. The bottom right of the interface has a search bar and transaction monitoring controls.

Cryptographic function

Code:

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar includes tabs for "Remix - Ethereum IDE" and "Computers - Google Drive". Below the tabs, there are browser-like controls for back, forward, and search, along with a message indicating "Not secure". The main workspace displays a Solidity code editor with the file "CryptographicFunction.sol" open. The code defines a contract with one function: `callKeccak256`, which returns a bytes32 result and uses the `keccak256` function with the argument "ABC". To the left of the code editor is a sidebar titled "FILE EXPLORERS". It shows a file tree under "remixDefault_165561734405" with several contracts like 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, 1A.sol, 1B.sol, while loop.sol, while loop1.sol, wLsol, Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, and CryptographicFunction.sol. The "CryptographicFunction.sol" file is currently selected. The bottom right of the interface has a search bar and transaction monitoring controls.

Output:

The screenshot shows the Ethereum IDE (Remix) interface. The top navigation bar includes tabs for Remix - Ethereum IDE, Computers - Google Drive, and several other open files like PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, and CryptographicFunction.sol. The main workspace displays the Solidity code for the CryptographicFunction contract:

```
pragma solidity ^0.5.0;
contract CryptographicFunction {
    function callKeccak256() public pure returns(bytes32 result){
        return keccak256("ABC");
    }
}
```

The left sidebar features a "DEPLOY & RUN TRANSACTIONS" panel. It shows a list of recorded transactions (26 total) and a section for deployed contracts. Under "Deployed Contracts", there is an entry for "CRYPTOGRAPHICFUNCTION AT 0x4e...". Below it, a button labeled "callKeccak256" is shown, with a transaction history entry for the first call:

0: bytes32 result 0xa1629b94dd0600b30c7908
34d09e7189c10773fa148a3360701bae95d54
f3cb

At the bottom of the sidebar, there are sections for "Low level interactions" and "CALLDATA", along with a "Transact" button.

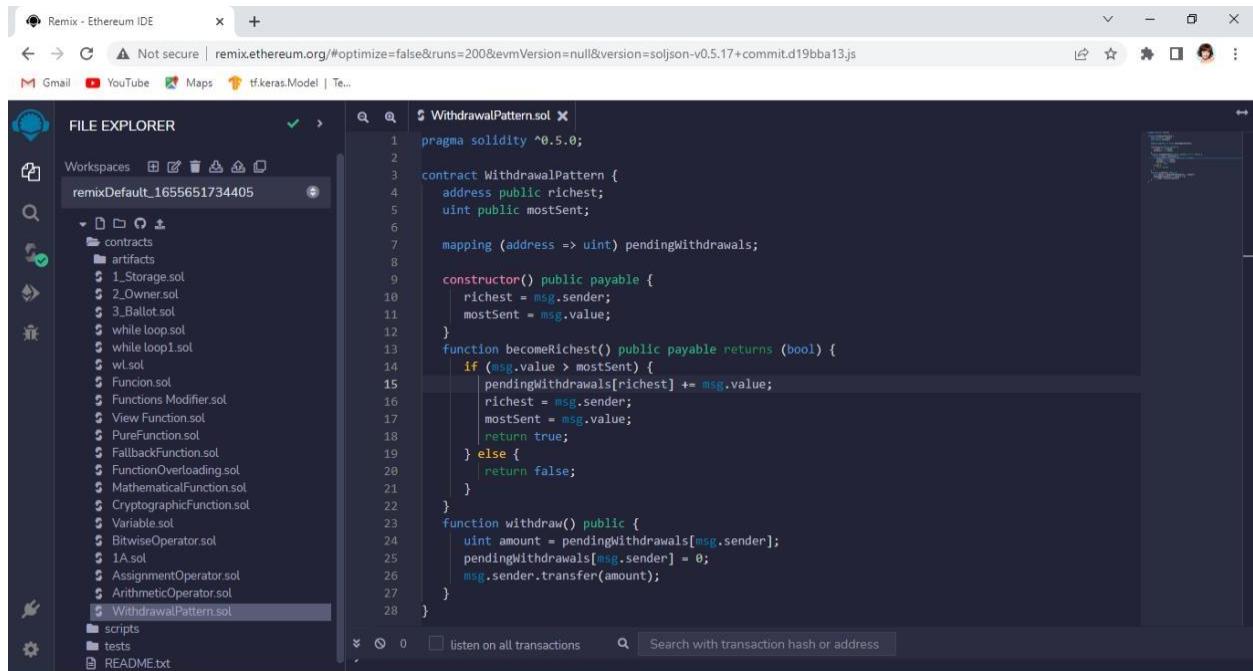
Practical No: 3

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

3a) Withdrawal Pattern, Restricted Access.

Withdrawal Pattern

Code:



The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and a warning "Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". Below the top bar is a toolbar with various icons. The main area is divided into two panes: the left pane is the "FILE EXPLORER" showing a workspace named "remixDefault_1655651734405" containing several Solidity files like "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", and "WithdrawalPattern.sol"; the right pane is the "EDITOR" showing the Solidity code for "WithdrawalPattern.sol". The code implements the Withdrawal Pattern with a constructor setting the richest address and mostSent value, a function to become the new richest if msg.value > mostSent, and a withdraw function to transfer funds from pendingWithdrawals[msg.sender] to msg.sender.

```
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;
        }
    }

    function withdraw() public {
        uint amount = pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
        msg.sender.transfer(amount);
    }
}
```

Output:

```

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;
        }
    }

    function withdraw() public {
        require(pendingWithdrawals[msg.sender] > 0);
        uint amount = pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
        msg.sender.transfer(amount);
    }
}

```

Restricted Access

Code:

```

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."
        );
    }

    function disown() public onlyBy(owner) onlyAfter(creationTime + 6 weeks) {
        delete owner;
    }

    modifier costs(uint _amount) {
        require(
            msg.value >= _amount,
            "Insufficient funds"
        );
    }
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left is the FILE EXPLORER sidebar, which lists several workspace files under 'remixDefault_1655651734405'. The main area displays two Solidity source files: 'WithdrawalPattern.sol' and 'RestrictedAccess.sol'. The code for 'WithdrawalPattern.sol' is as follows:

```
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;
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. It displays the deployment and interaction details for the 'WithdrawalPattern' contract. The 'Transactions recorded' section shows a single transaction with status 'Success' and hash '0x0D7A...F771B'. The 'Deployed Contracts' section shows the deployed contract with its address: '0x5838Da6a701c568545dCfcB03FcB875f56beddC4'. The 'Low level interactions' section shows the contract's creation time and owner information.

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with sections for 'DEPLOY & RUN TRANSACTIONS' (with options like 'Publish to IPFS' and 'At Address'), 'Transactions recorded' (showing 20 entries), and 'Deployed Contracts' (listing 'TEST AT 0xC0D6...99DF9 (MEMORY)' with three functions: 'changeOwner', 'disown', and 'forceOwnerCh...', and a variable 'creationTime'). The main area has two tabs open: 'WithdrawalPattern.sol' and 'RestrictedAccess.sol'. The 'WithdrawalPattern.sol' tab contains the following Solidity code:

```

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."
        );
    }
}

```

Below the code, there are logs for transactions:

- [vm] from: 0x58...edd4 to: WithdrawalPattern.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xa8e...1a9ed transact to withdrawalPattern.becomeRichest pending ...
- [vm] from: 0xb3...39138 to: WithdrawalPattern.becomeRichest() 0xd91...39138 value: 0 wei data: 0x699...34ee7 logs: 0 hash: 0x3b3...0b3ea

On the right, there are 'Debug' buttons.

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

Contracts

Code:

The screenshot shows the Remix Ethereum IDE with the 'FILE EXPLORER' sidebar open, displaying a workspace named 'remixDefault_1655651734405' containing several contracts: 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, while_loop.sol, while_loop1sol, wl.sol, Function.sol, Functions.Modifier.sol, View.Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, IA.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, and Contracts.sol. The main editor window shows the 'Contracts.sol' file with the following Solidity code:

```

pragma solidity ^0.5.0;

contract C {
    //private state variable
    uint private data;

    //public state variable
    uint public info;

    //constructor
    constructor() public {
        info = 10;
    }

    //private function
    function increment(uint a) private pure returns(uint) { return a + 1; }

    //public function
    function updateData(uint a) public { data = a; }

    function getData() public view returns(uint) { return data; }

    function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}

//External Contract
contract D {
    function readData() public returns(uint) {
        C c = new C();
        c.updateData(7);
        return c.getData();
    }
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar displays a workspace named "remixDefault_1655651734405" containing several Solidity files under the "contracts" folder, such as 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, whileloop.sol, withdrawPattern.sol, Contracts.sol, and RestrictedAccess.sol. The main central area shows the source code for the "Contracts.sol" file:

```
//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(); }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar active. It includes options for "Deploy", "Publish to IPFS", and "At Address". The "Transactions recorded" section shows 21 entries. The "Deployed Contracts" section lists a deployed contract at address 0x05FD...9D88D (MEMORY). The "Low level interactions" section shows methods like updateData, getData, info, and a "Transact" button. The central code editor area shows the same "Contracts.sol" code as before. The bottom right corner features a "Debug" button. Transaction logs are visible in the terminal-like interface at the bottom:

```
[call] from: 0x5830a6a701c568545dCfcB03FcB875f56beddC4 to: Test.owner() data: 0x8da...5cb5b creation of c pending...

[vm] from: 0x5830a6a701c568545dCfcB03FcB875f56beddC4 to: C.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x442...f1745
transact to C.updateData errored: Error encoding arguments: Error: invalid BigNumber string (argument="value", value="", code=INVALID_ARGUMENT, version=b

call to C.getData

[call] from: 0x5830a6a701c568545dCfcB03FcB875f56beddC4 to: C.getData() data: 0x30e...5de30
```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with options for deploying transactions, publishing to IPFS, or running contracts at an address. It also displays a list of deployed contracts and their addresses. The main area shows the Solidity code for a contract named `WithdrawalPattern.sol`. The code defines a base contract `C` with a private state variable `data` and a public function `info` returning `uint`. It also defines a derived contract `E` that inherits from `C`, has a constructor that initializes `c = new C();`, and overrides the `info` function to return `c.info()`. Below the code editor, the transaction history shows three calls to the `info` function of the deployed contract at address `0x5FD...9D88D`.

```

//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(); }
}

```

Inheritance

Code:

The screenshot shows the Remix Ethereum IDE interface with the file explorer open, displaying a workspace named `remixDefault_165565173405`. The file tree shows several contracts including `1.Storage.sol`, `2.Owner.sol`, `3.Ballot.sol`, `while loop.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`, and `Inheritance.sol`. The code editor shows the Solidity code for `Inheritance.sol`, which contains two contracts: `C` and `E`. Contract `C` has a private state variable `data` and a public function `info`. Contract `E` inherits from `C` and overrides the `info` function to return `c.info()`.

```

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();
    }
}

```

```
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 }
```

Output:

```
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     }
29 }
```

```

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();
    }

    function getResult() public view returns(uint) { return result; }

    function setResult(uint result) public { this.result = result; }

    function callC() public {
        c.increment();
    }
}

```

Constructors

Code:

```

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;
    }
}

```

Output:

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options for "Deploy" and "At Address". Below it, "Transactions recorded" and "Deployed Contracts" are listed, with "CONSTRUCTOREXAMPLE AT 0X1C" expanded to show its code and a "getValue" button. The main area displays two Solidity files: "Constructors.sol" and "Inheritance.sol". "Constructors.sol" contains the following code:

```
pragma solidity ^0.5.0;

contract constructorExample {
    string str;

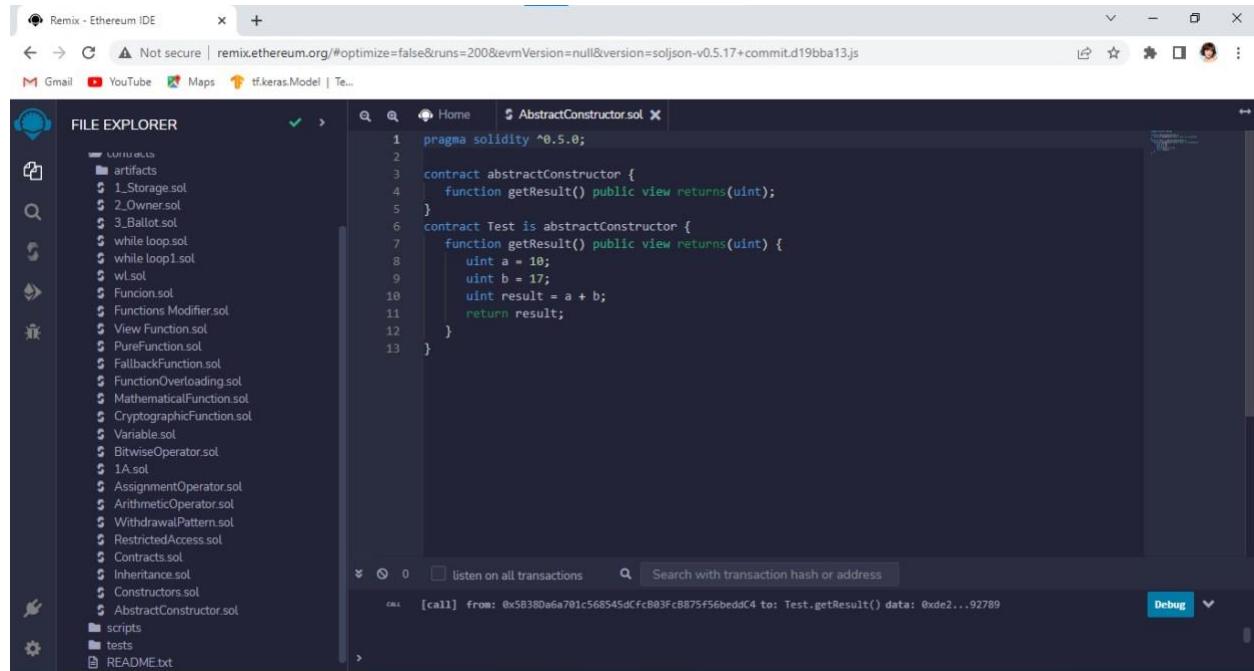
    constructor() public{
        str="This is a example of Constructor";
    }

    function getValue()
    public view returns(
        string memory)
    {
        return str;
    }
}
```

The "Inheritance.sol" file is shown as a reference. At the bottom, transaction logs are displayed, showing the creation of the contract and a call to its "getValue" function.

Abstract Contracts

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists various Solidity files such as 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, while loop.sol, white 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, IA.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, scripts, tests, and README.txt. The main editor window is titled "AbstractConstructor.sol" and contains the following Solidity code:

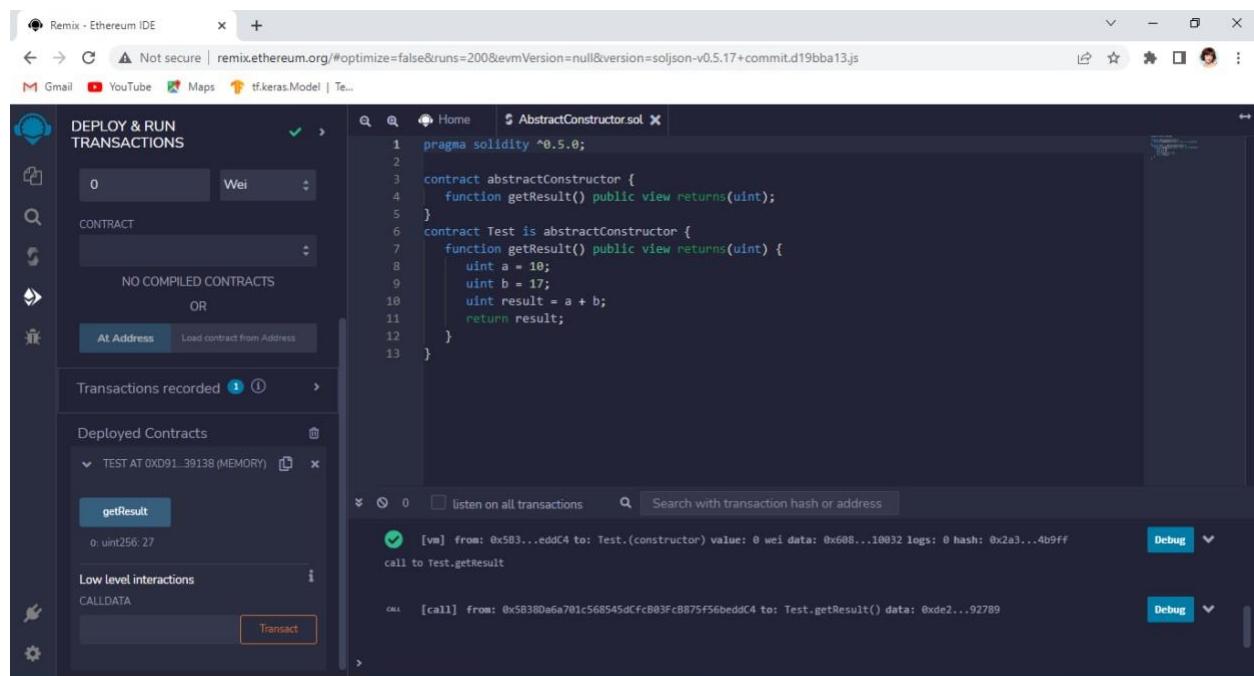
```
pragma solidity ^0.5.0;

contract abstractConstructor {
    function getResult() public view returns(uint);
}

contract Test is abstractConstructor {
    function getResult() public view returns(uint) {
        uint a = 10;
        uint b = 17;
        uint result = a + b;
        return result;
    }
}
```

The bottom right corner of the editor window shows a "Debug" button.

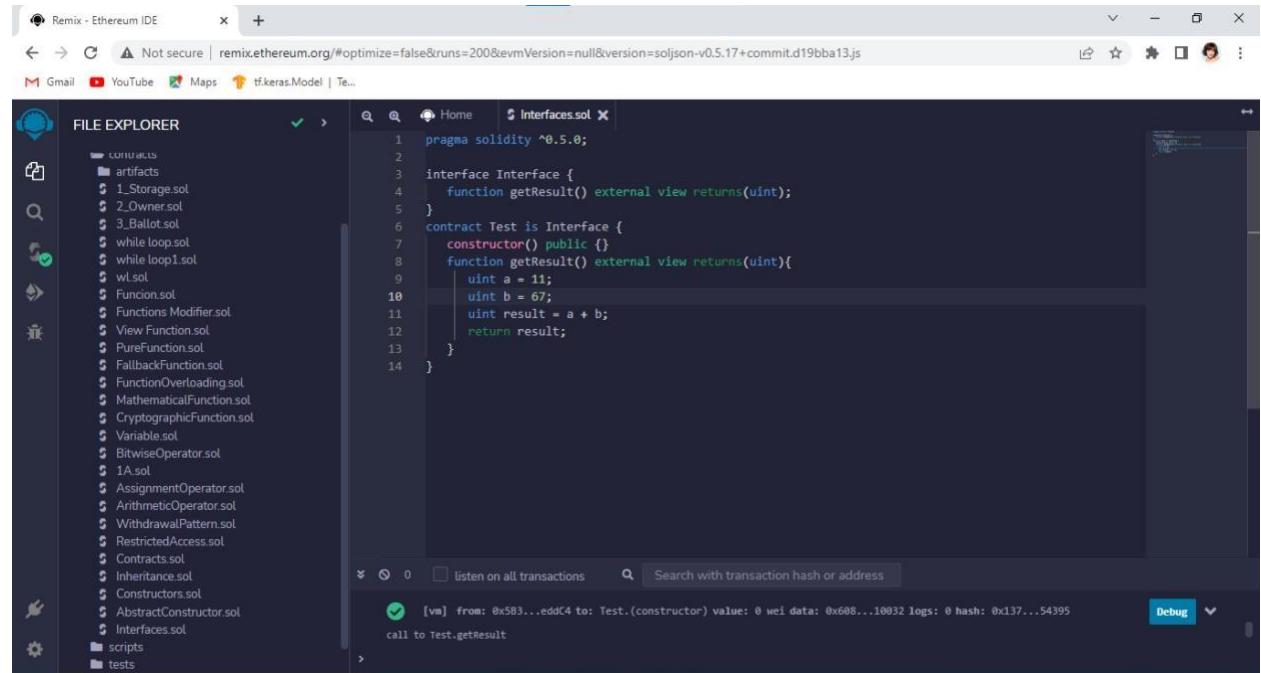
Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. The sidebar includes fields for "Wei" and "At Address". It also displays "CONTRACT" and "NO COMPILED CONTRACTS" sections. Below these, under "Deployed Contracts", there is a section for "TEST AT 0xD91...39138 (MEMORY)". Under this section, the "getResult" function is shown with its output: "0: uint256: 27". The bottom part of the interface shows the same Solidity code as the previous screenshot, and the bottom right corner of the editor window shows a "Debug" button.

Interfaces

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files under the "contracts" folder, including 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, white_loop.sol, white_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, scripts, and tests.

The main editor window is titled "Interfaces.sol" and contains the following Solidity code:

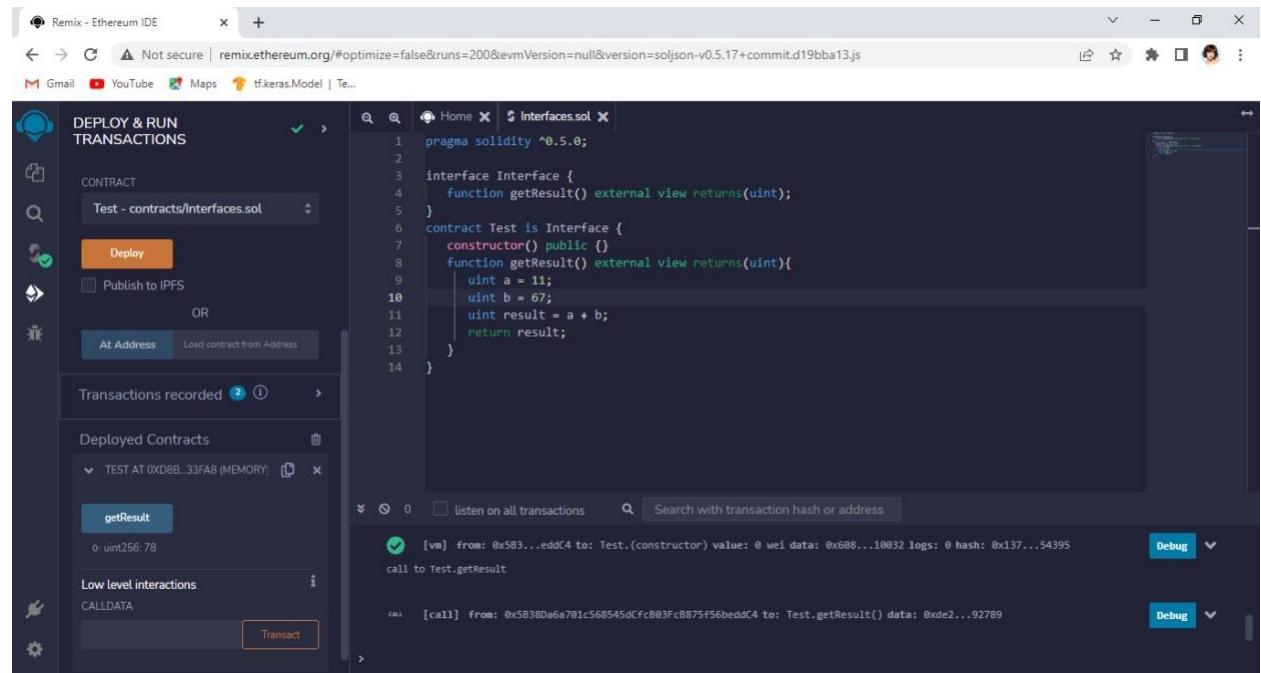
```
pragma solidity ^0.5.0;

interface Interface {
    function getResult() external view returns(uint);
}

contract Test is Interface {
    constructor() public {}
    function getResult() external view returns(uint){
        uint a = 11;
        uint b = 67;
        uint result = a + b;
        return result;
    }
}
```

The bottom right corner of the editor shows a "Debug" button. Below the editor, the transaction history pane shows a single transaction: "[vm] from: 0x5B3...eddC4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x137...54395 call to Test.getResult".

Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. The "CONTRACT" dropdown is set to "Test - contracts/Interfaces.sol". The "Deploy" button is highlighted. Below it, there are options for "Publish to IPFS" and "At Address". The "At Address" option is selected, and the address "TEST AT 0xD8B. 33FA8 (MEMORY)" is shown. Under "Transactions recorded", the "getResult" function is listed with the output "0: uint256: 78".

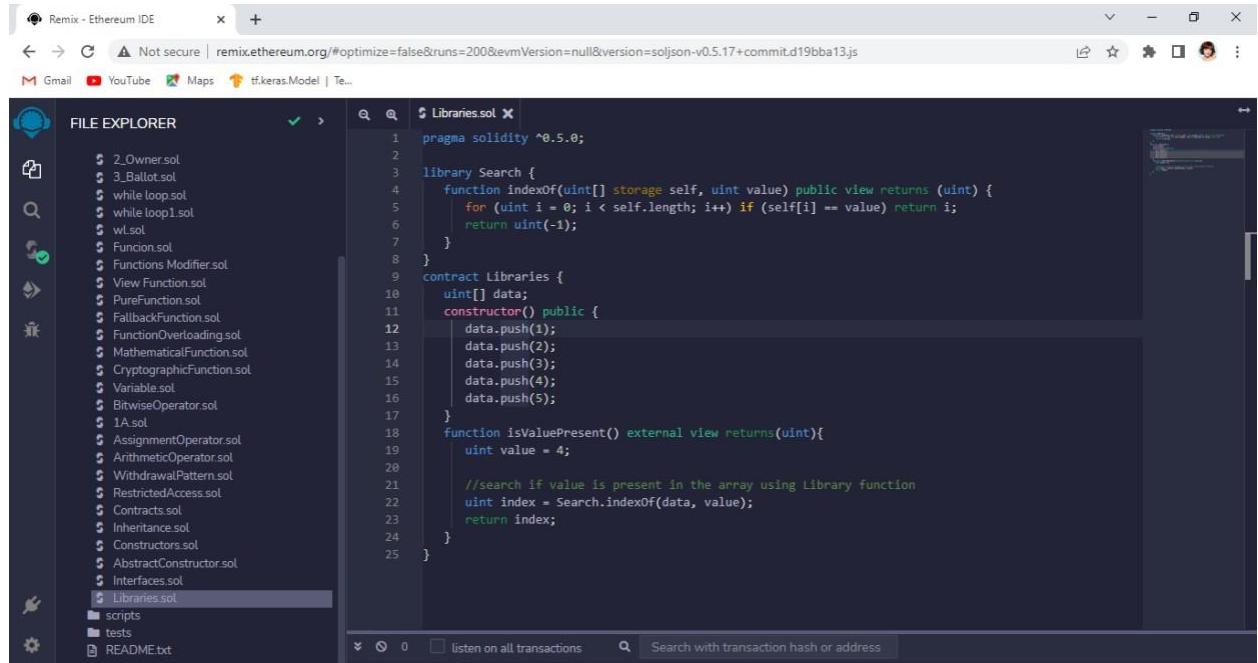
The main editor window is titled "Interfaces.sol" and contains the same Solidity code as the previous screenshot.

The bottom right corner of the editor shows a "Debug" button. Below the editor, the transaction history pane shows two transactions: "[vm] from: 0x5B3...eddC4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x137...54395 call to Test.getResult" and "[call] from: 0x5B38D0a6a701c568545dCfcB03FcB875f56beddC4 to: Test.getResult() data: 0xde2...92789".

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

Libraries

Code:



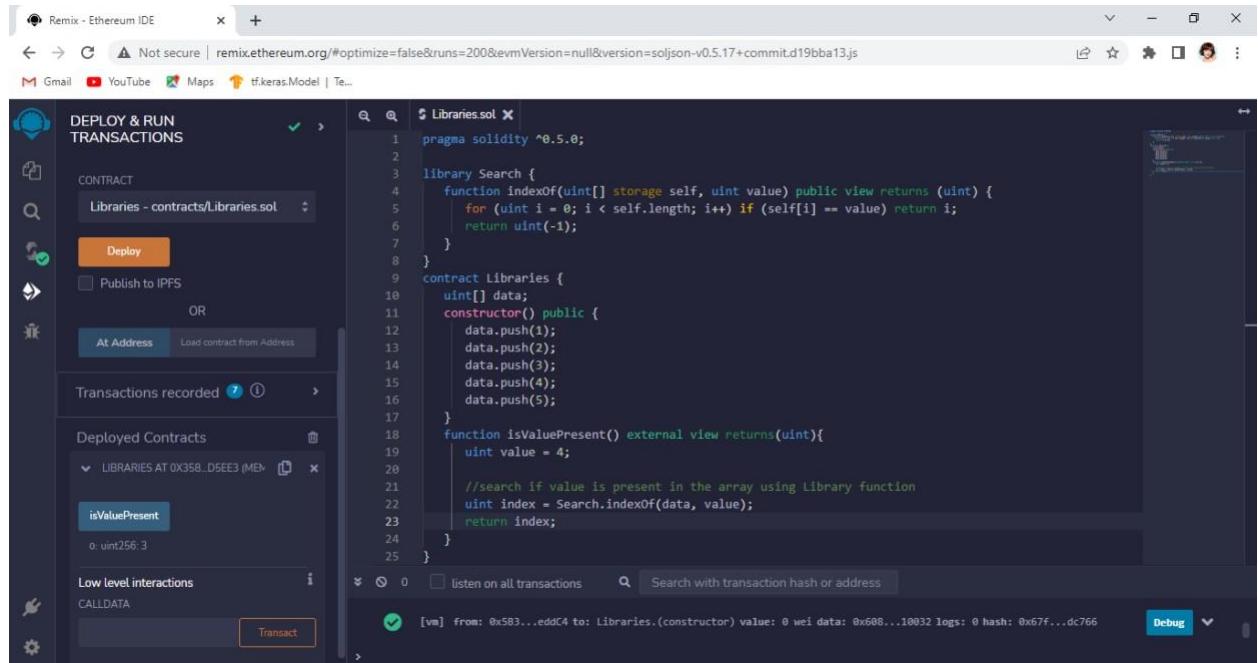
The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled 'FILE EXPLORER' and lists several Solidity files: 2_Owner.sol, 3_Ballot.sol, while.loop.sol, while.loop1.sol, wl.sol, Funcion.sol, Functions.Modifier.sol, View.Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, 1A.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, Libraries.sol, scripts, tests, and README.txt. The 'Libraries.sol' file is currently selected and open in the main editor area. The code implements a library named 'Search' and a contract named 'Libraries'. The 'Search' library contains a function 'indexOf' that returns the index of a value in an array. The 'Libraries' contract contains a constructor that initializes an array with values 1 through 5, and an external view function 'isValuePresent' that uses the 'Search' library's function to find the index of a given value.

```
pragma solidity ^0.5.0;

library Search {
    function indexOf(uint[] storage self, uint value) public view returns (uint) {
        for (uint i = 0; i < self.length; i++) if (self[i] == value) return i;
        return uint(-1);
    }
}

contract Libraries {
    uint[] data;
    constructor() public {
        data.push(1);
        data.push(2);
        data.push(3);
        data.push(4);
        data.push(5);
    }
    function isValuePresent() external view returns(uint){
        uint value = 4;
        //search if value is present in the array using Library function
        uint index = Search.indexOf(data, value);
        return index;
    }
}
```

Output:



The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab active. The 'CONTRACT' section shows 'Libraries - contracts/Libraries.sol' selected, with a 'Deploy' button highlighted. Below it, there are options to 'Publish to IPFS' or 'At Address'. The 'Transactions recorded' section shows a deployment transaction for 'LIBRARIES AT 0X358...D5EE3 (MEN)' with a status of '0: uint256: 3'. The 'Deployed Contracts' section shows the same address with a status of 'isValuePresent'. The bottom section shows 'Low level interactions' and 'CALLDATA' with a 'Transact' button. The right side of the screen displays the same Solidity code as the previous screenshot, and the status bar at the bottom indicates a successful deployment with the message '[vm] from: 0x5B3...eddC4 to: Libraries.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x67f...dc766'.

Assembly

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files: 3_Ballot.sol, while_loop.sol, while_loops.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, scripts, tests, and README.txt. The file "Assembly.sol" is currently selected and highlighted in blue. The main editor window displays the Solidity code for the "Assembly" contract, which contains an assembly block for the "add" function. The code uses EVM assembly instructions like add, mstore, sload, and let to perform arithmetic operations. The status bar at the bottom shows a transaction call from address 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to the contract, with data 0x100...0001.

```
pragma solidity ^0.4.0;

contract Assembly {

    function add(uint a) view returns (uint b) {
        assembly {
            let c := add(a, 16)
            mstore(0x80, c)
            {
                let d := add(sload(c), 12)
                // assign the value of 'd' to 'b'
                b := d
                // 'd' is deallocated now
            }
            b := add(b, c)
        }
    }
}
```

Output:

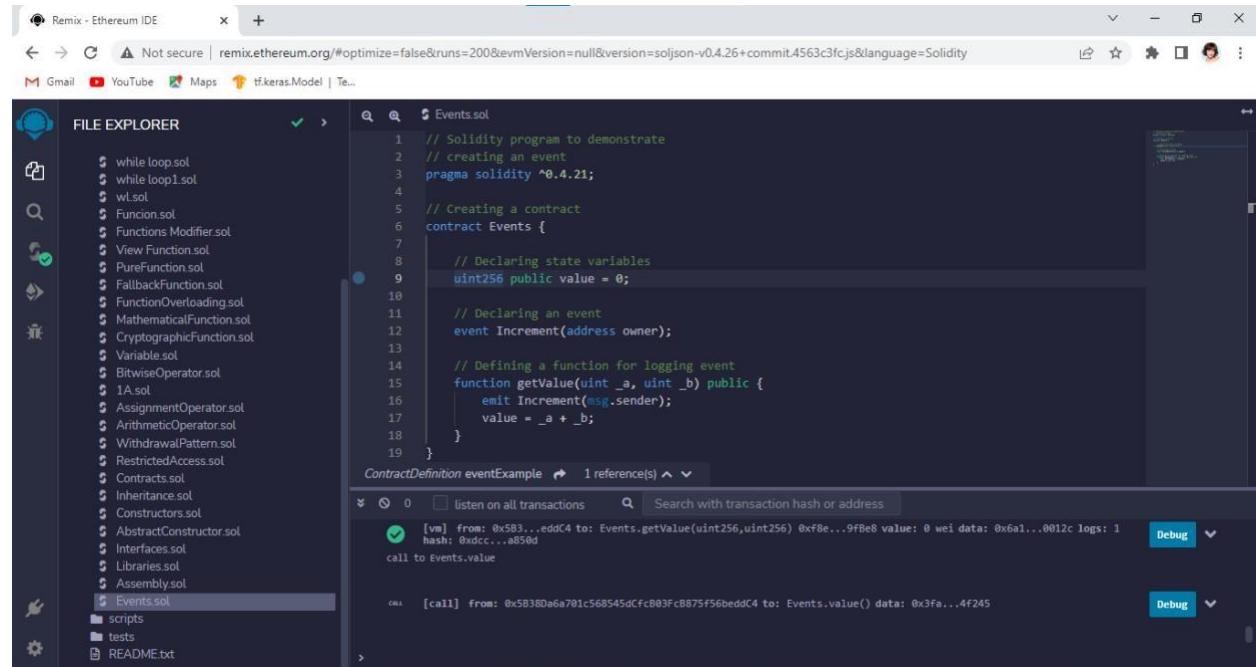
The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active. The "CONTRACT" dropdown is set to "Assembly - contracts/Assembly.sol". Below it, there is a "Deploy" button and options to "Publish to IPFS" or "At Address". The "Transactions recorded" section shows a single entry: "ASSEMBLY AT 0xD91...3913B (MEN)". The main editor window shows the same Solidity code as the previous screenshot. The status bar at the bottom shows a transaction call from address 0x5B3...eddC4 to the constructor of the Assembly contract, with data 0x608...20029 and logs 0 hash: 0xf4b...e5b5d. A message below indicates an error: "call to Assembly.add errored: Error encoding arguments: Error: invalid BigNumber string (argument='value', value='', code=INVALID_ARGUMENT, version=bignumber@2.3.1)".

```
contract Assembly {

    function add(uint a) view returns (uint b) {
        assembly {
            let c := add(a, 16)
            mstore(0x80, c)
            {
                let d := add(sload(c), 12)
                // assign the value of 'd' to 'b'
                b := d
                // 'd' is deallocated now
            }
            b := add(b, c)
        }
    }
}
```

Events

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files: 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, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, Libraries.sol, Assembly.sol, and Events.sol. The file "Events.sol" is currently selected and highlighted in blue. The main editor area displays the following Solidity code:

```
// Solidity program to demonstrate
// creating an event
pragma solidity ^0.4.21;

// Creating a contract
contract Events {

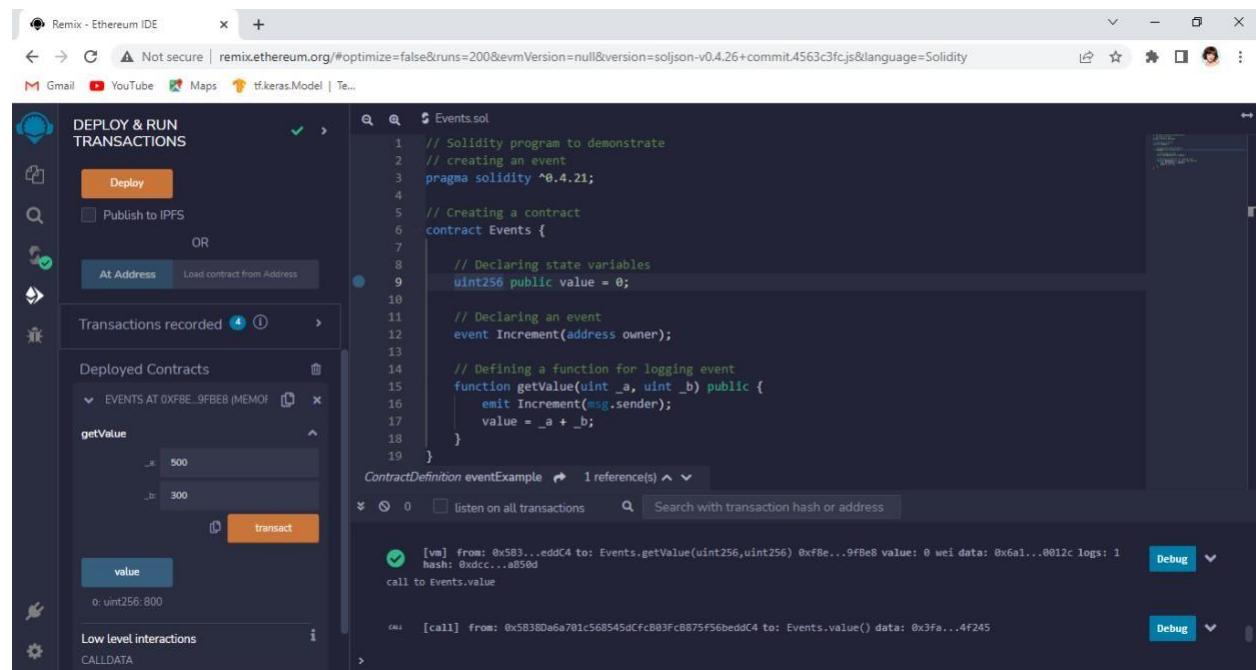
    // Declaring state variables
    uint256 public value = 0;

    // Declaring an event
    event Increment(address owner);

    // Defining a function for logging event
    function getValue(uint _a, uint _b) public {
        emit Increment(msg.sender);
        value = _a + _b;
    }
}
```

Below the code editor, the "ContractDefinition" section shows "eventExample" with "1 reference(s)". The bottom pane shows transaction logs and a debugger interface.

Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active. The left sidebar shows the "Deploy" button is highlighted. The main editor area displays the same Solidity code as the previous screenshot. The bottom pane shows the transaction logs and a debugger interface. The "Transactions recorded" section indicates "EVENTS AT 0XF8E_9FBEB (MEMO)" and shows a call to the "getValue" function with parameters $_a = 500$ and $_b = 300$, resulting in a value of $o: \text{uint256: } 800$. There is also a "Low level interactions" section with "CALLDATA".

Error Handling

Code:

The screenshot shows the Ethereum IDE interface with the file `ErrorHandling.sol` open in the central editor area. The code defines a Solidity contract named `ErrorHandling` with two functions: `checkInput` and `Odd`. The `checkInput` function takes a uint input and returns a string indicating if it's an even or odd number. The `Odd` function takes a uint input and returns a boolean value indicating if the input is odd.

```
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;
    }
}
```

Output:

The screenshot shows the Ethereum IDE interface with the `DEPLOY & RUN TRANSACTIONS` tab selected. It displays the deployed contract `ERRORHANDLING AT 0x907...B5E` and its interactions. The `checkInput` function is called with input `243`, returning the string `Input is Uint8`. The `Odd` function is also called with input `243`, returning the boolean value `true`.

Transactions recorded:

- call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.checkInput(uint256) data: 0x021...000f3
- call to ErrorHandling.Odd
- call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.Odd(uint256) data: 0x922...000f3

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.



```
puneet@puneet-VirtualBox: ~/multichain_network/fabric-samples/first-network
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ cryp
togen generate --config=./crypto-config.yaml
org1.example.com
org2.example.com
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$
```

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
```

```
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./composer-genesis.block
2019-03-18 14:50:56.908 IST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2019-03-18 14:50:56.924 IST [msp] getMspConfig -> INFO 002 Loading NodeOUs
2019-03-18 14:50:56.924 IST [msp] getMspConfig -> INFO 003 Loading NodeOUs
2019-03-18 14:50:56.924 IST [common/tools/configtxgen] doOutputBlock -> INFO 004 Generating genesis block
2019-03-18 14:50:56.925 IST [common/tools/configtxgen] doOutputBlock -> INFO 005 Writing genesis block
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$
```

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

```
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./composer-channel.tx -channelID composerchannel
2019-03-18 15:17:36.592 IST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2019-03-18 15:17:36.612 IST [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 002 Generating new channel configtx
2019-03-18 15:17:36.614 IST [msp] getMspConfig -> INFO 003 Loading NodeOUs
2019-03-18 15:17:36.616 IST [msp] getMspConfig -> INFO 004 Loading NodeOUs
2019-03-18 15:17:36.656 IST [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 005 Writing new channel tx
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$
```

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"
```

```

services:
  ca1:
    image: hyperledger/fabric-ca:$IMMUNE_TAG
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-org1
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-server-config/CA1_PRIVATE_KEY
    ports:
      - "7054:7054"
    command: sh -c 'Fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem --admin.adminpw -d'

```

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
```

```

services:
  ca1:
    image: hyperledger/fabric-ca:$IMMUNE_TAG
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-org2
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org2.example.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-server-config/CA2_PRIVATE_KEY
    ports:
      - "7054:7054"
    command: sh -c 'Fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org2.example.com-cert.pem --admin.adminpw -d'

```

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 –

```

10  orderer.example.com:
11    container_name: orderer.example.com
12    image: hyperledger/fabric-orderer:$IMAGE_TAG
13    environment:
14      - ORDERER_GENERAL_LOGLEVEL=INFO
15      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
16      - ORDERER_GENERAL_GENESISMETHOD=file
17      - ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block
18      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
19      - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
20    # enabled TLS
21    - ORDERER_GENERAL_TLS_ENABLED=true
22    - ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
23    - ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
24    - ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
25  working_dir: /opt/gopath/src/github.com/hyperledger/fabric
26  command: orderer
27  volumes:
28    - ./composer-genesis.block:/var/hyperledger/orderer/orderer.genesis.block
29    - ./crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hyperledger/orderer/msp
30    - ./crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls:/var/hyperledger/orderer/tls
31  ports:
32    - 7050:7050

```

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

```

- ./etc/configtx

```

```

35  peer0.org1.example.com:
36    extends:
37      file: peer-base.yaml
38    service: peer-base
39    environment:
40      - CORE_PEER_ID=peer0.org1.example.com
41      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
42      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org1.example.com:7051
43      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
44      - CORE_PEER_LOCALMSPID=Org1MSP
45    volumes:
46      - ./etc/configtx
47      - /var/run/:/host/var/run/
48      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp
49      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/etc/hyperledger/fabric/tls
50    ports:
51      - 7051:7051
52      - 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 couchdb6 ...
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 peer0.org1.example.com ...
Creating peer0.org2.example.com ... done
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	NAMES	COMMAND	CREATED	STATUS
badfe5ea8a92	hyperledger/fabric-peer:latest	"peer node start" peer1.org1.example.com		4 hours ago	Up 3 hours
50d9fa8f5dfb	hyperledger/fabric-peer:latest	"peer node start" peer0.org2.example.com		4 hours ago	Up 3 hours
ee4b501b7d00	hyperledger/fabric-peer:latest	"peer node start" peer0.org1.example.com		4 hours ago	Up 3 hours
32f0f4426a22	hyperledger/fabric-peer:latest	"peer node start" peer1.org2.example.com		4 hours ago	Up 3 hours
ec8f9df17258	hyperledger/fabric-couchdb 4369/tcp, 9100/tcp, 0.0.0.0:8984->5984/tcp	couchdb3 "tini -- /docker-entrypoint-initdb.d"		4 hours ago	Up 4 hours
2b4be64efcfe	hyperledger/fabric-couchdb 4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp	couchdb2 "tini -- /docker-entrypoint-initdb.d"		4 hours ago	Up 4 hours
f922625a027e	hyperledger/fabric-couchdb 4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp	couchdb0 "tini -- /docker-entrypoint-initdb.d"		4 hours ago	Up 4 hours
8de729e174b0	hyperledger/fabric-couchdb 4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp	couchdb1 "tini -- /docker-entrypoint-initdb.d"		4 hours ago	Up 4 hours
18830d7ccf5e	hyperledger/fabric-tools:latest	/bin/bash cli		11 hours ago	Up 11 hours
c7ef15868cd5	hyperledger/fabric-orderer:latest 0.0.0.0:7050->7050/tcp	"orderer" orderer.example.com		11 hours ago	Up 11 hours

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(nonce)      new_hash =
SHA256(text)      if new_hash.startswith(prefix_str):
        print(f"Yay! Successfully mined bitcoins with nonce value:{nonce}")
        return new_hash

    raise BaseException(f"Couldn't find correct hash 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
0000de957fbfdfc77582e0d0b20c53d2d1d83d8bb8cf3693521f672bf2a6021
```

Practical No: 7

Aim: Create your own blockchain and demonstrate its use.

Refer Practical No: 1, 2 3

Advance IOT

Sr.no.	Date	Topic	Pg.No.
1.	24/08/2018	Displaying different LED patterns with Raspberry Pi.	62
2.	05/09/2018	Displaying Time over 4-Digit 7-Segment Display using Raspberry Pi.	64
3.	10/09/2018	Raspberry Pi Based Oscilloscope.	65
4.	19/09/2018	Controlling Raspberry Pi with Telegram.	76
5.	24/09/2018	Setting up Wireless Access Point using Raspberry Pi.	83
6.	28/09/2018	Raspberry Pi GPS Module Interfacing.	93
7.	03/10/2018	IoT based Web Controlled Home Automation using Raspberry Pi.	95
8.	08/10/2018	Interfacing Raspberry Pi with Pi Camera.	96
9.	12/10/2018	Interfacing Raspberry Pi with RFID.	98
10.	17/10/2018	Installing Windows 10 IoT Core on Raspberry Pi	105

1	Displaying different LED patterns with Raspberry Pi.
	<pre> import RPi.GPIO as GPIO import time x=1 numTimes=int(input("Enter total 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) </pre>

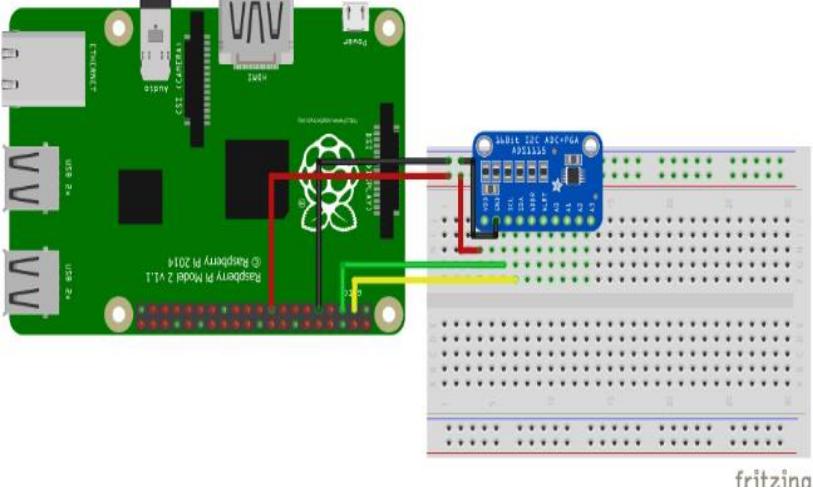
```
GPIO.output(10,False)
print ("Iteration ", (i+1))
time.sleep(speed)

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. <p>Control 4 digits-7 segments LED display with TM1637 controller</p> <p>Connection scheme Raspberry Pi</p> <p>Connect the LED to your Raspberry Pi according to the following diagram:</p> <p>TM1637 script</p> <p>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:</p> <pre>wget https://raspberrytips.nl/files/tm1637.py</pre> <p>Code:</p> <pre>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.SetBrightness(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)</pre>
----------	--

3	Raspberry Pi Based Oscilloscope
	<p>Project Requirements</p> <p>The requirement for this project can be classified into two:</p> <ol style="list-style-type: none"> 1. Hardware Requirements 2. Software Requirements <p>Hardware requirements</p> <p>To build this project, the following components/part are required;</p> <ol style="list-style-type: none"> 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) <p>Software Requirements</p> <p>The software requirements for this project are basically the python modules (<i>matplotlib and drawnow</i>) 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.</p> <p>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.</p> <p>With all the hardware components in place, let's create the schematics and connect the components together.</p> <p>Circuit Diagram:</p> <p>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.</p>



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 ADS1115 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.

```
pi@raspberrypi: ~/Adafruit_Python_ADS1x15

Installed /usr/local/lib/python2.7/dist-packages/Adafruit_GPIO-1.0.3-py2.7.egg
Searching for adafruit-pureio
Reading https://pypi.python.org/simple/adafruit-pureio/
Downloading https://pypi.python.org/packages/55/fa/99b1006fb4bb356762357b297d8db
6ec9ffal3af480692ab72aa4a0dd0c4/Adafruit_PureIO-0.2.1.tar.gz#md5=5b3276059eb55d6
c37429a8413a92029
Best match: Adafruit-PureIO 0.2.1
Processing Adafruit_PureIO-0.2.1.tar.gz
Writing /tmp/easy_install-0wCISv/Adafruit_PureIO-0.2.1/setup.cfg
Running Adafruit_PureIO-0.2.1/setup.py -q bdist_egg --dist-dir /tmp/easy_install
-0wCISv/Adafruit_PureIO-0.2.1/egg-dist-tmp-F00Kpv
zip_safe flag not set; analyzing archive contents...
Moving Adafruit_PureIO-0.2.1-py2.7.egg to /usr/local/lib/python2.7/dist-packages
Adding Adafruit-PureIO 0.2.1 to easy-install.pth file

Installed /usr/local/lib/python2.7/dist-packages/Adafruit_PureIO-0.2.1-py2.7.egg
Searching for spidev==3.3
Best match: spidev 3.3
Adding spidev 3.3 to easy-install.pth file

Using /usr/lib/python2.7/dist-packages
Finished processing dependencies for Adafruit-ADS1x15==1.0.2
pi@raspberrypi:~/Adafruit_Python_ADS1x15 $
```

Step 4: Test the library and I2C 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 sampletest.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.

```

PuTTY (inactive)

pi@raspberrypi:~ $ sudo apt-get install python-matplotlib
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fonts-lyx libglade2-0 libjs-jquery-ui python-cycler python-dateutil
  python-functools32 python-glade2 python-imaging python-matplotlib-data
  python-pyparsing python-subprocess32 python-tz ttf-bitstream-vera
Suggested packages:
  libjs-jquery-ui-docs python-cycler-doc python-gtk2-doc dvipng ffmpeg
  ghostscript inkscape ipython python-cairocffi python-configobj
  python-excelerator python-matplotlib-doc python-nose python-qt4 python-
  python-sip python-tornado python-traits python-wxgtk3.0 texlive-extra-
  texlive-latex-extra ttf-staypuft python-pyparsing-doc
The following NEW packages will be installed:
  fonts-lyx libglade2-0 libjs-jquery-ui python-cycler python-dateutil
  python-functools32 python-glade2 python-imaging python-matplotlib
  python-matplotlib-data python-pyparsing python-subprocess32 python-tz
  ttf-bitstream-vera
0 upgraded, 14 newly installed, 0 to remove and 4 not upgraded.
Need to get 7,115 kB of archives.
After this operation, 23.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://raspbian.mirror.garr.it/mirrors/raspbian/raspbian/stretch/m
```

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.

```
pi@raspberrypi:~ $ sudo pip install drawnow
Collecting drawnow
  Downloading drawnow-0.71.3.tar.gz
Requirement already satisfied: matplotlib>=1.5 in /usr/lib/python2.7/dist-packages (from drawnow)
Building wheels for collected packages: drawnow
  Running setup.py bdist_wheel for drawnow ... done
  Stored in directory: /root/.cache/pip/wheels/83/90/79/cc7449a69f925bfbee33f582fa58febee3e2d0944ccb058
Successfully built drawnow
Installing collected packages: drawnow
Successfully installed drawnow-0.71.3
pi@raspberrypi:~ $
```

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('Oscilloscope')
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
```

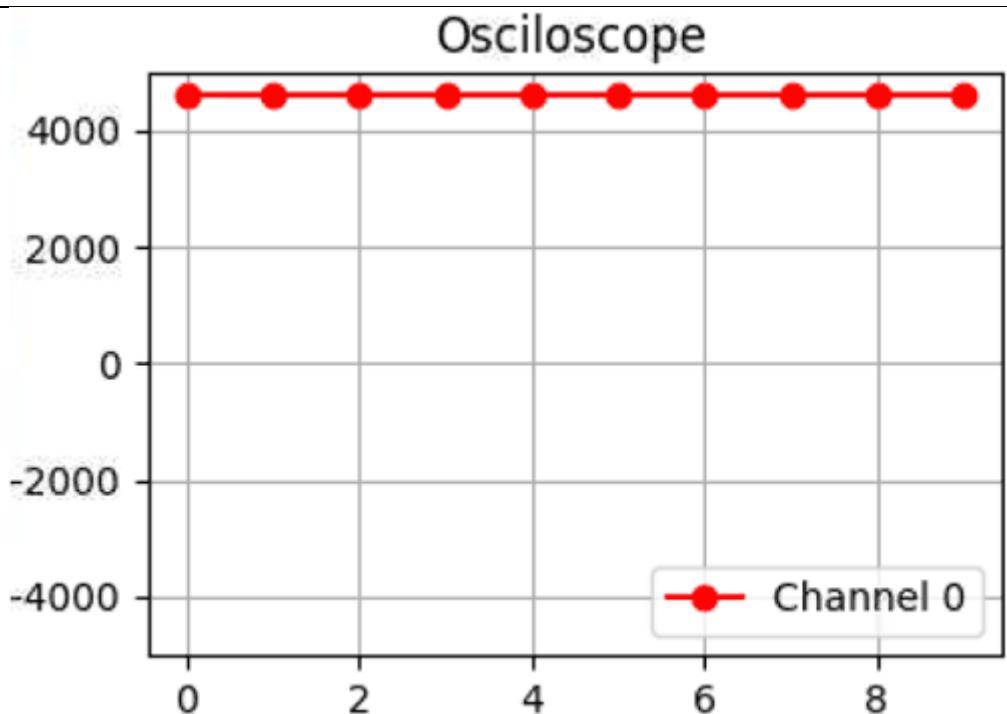
If you used a different name other than scope.py, don't forget to change this to match.

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)
```

```
pi@raspberrypi:~ $ sudo nano scope.py  
pi@raspberrypi:~ $ sudo python scope.py  
Reading ADS1x15 channel 0  
Channel 0: 4618  
/usr/lib/python2.7/dist-packages/matplotlib/backend  
plotlibDeprecationWarning: Using default event loop  
cific to this GUI is implemented  
    warnings.warn(str, mplDeprecation)  
Channel 0: 4615  
Channel 0: 4616  
Channel 0: 4615  
Channel 0: 4614  
Channel 0: 4613  
Channel 0: 4614
```



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('Oscilloscope')
    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)
```

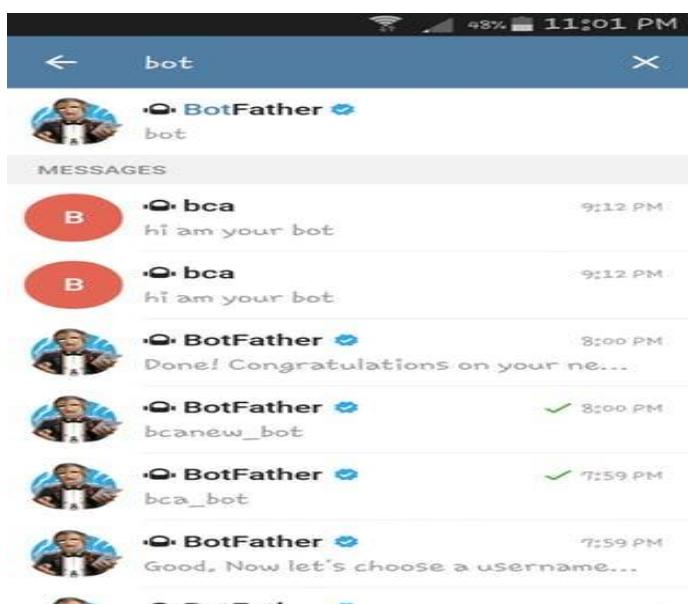
4 Controlling Raspberry Pi with Telegram.

Step 1: Open Telegram app in your system

or mobile

1.1 Open Telegram app in your system or mobile\

1.2 Start "BotFather"



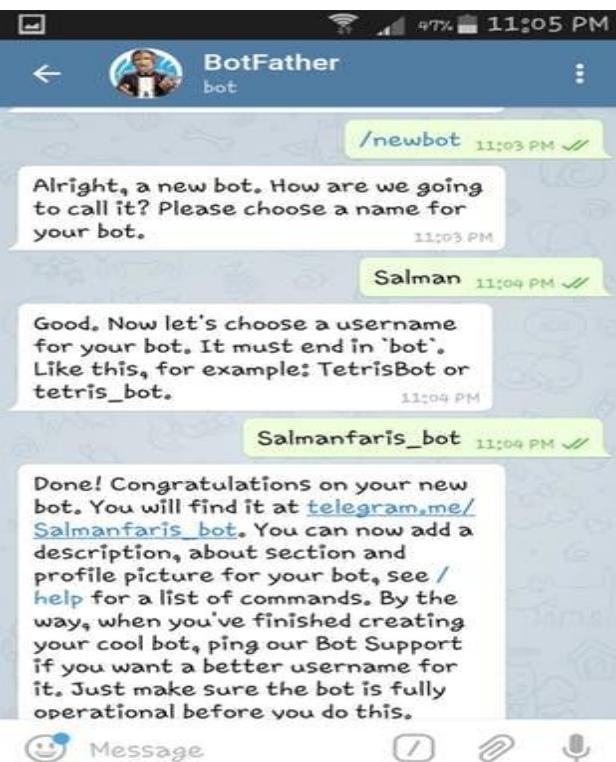
1.3 Open "BotFather"



1.4 Start "BotFather"



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')
```

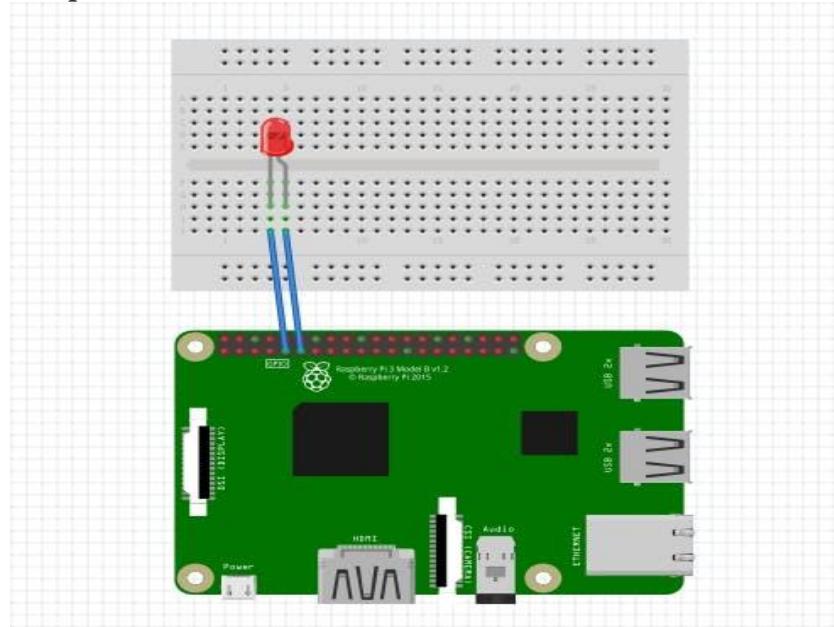
Note: 1.6 for more details

4.3 Run the Code

```
python telegrambot.py
```

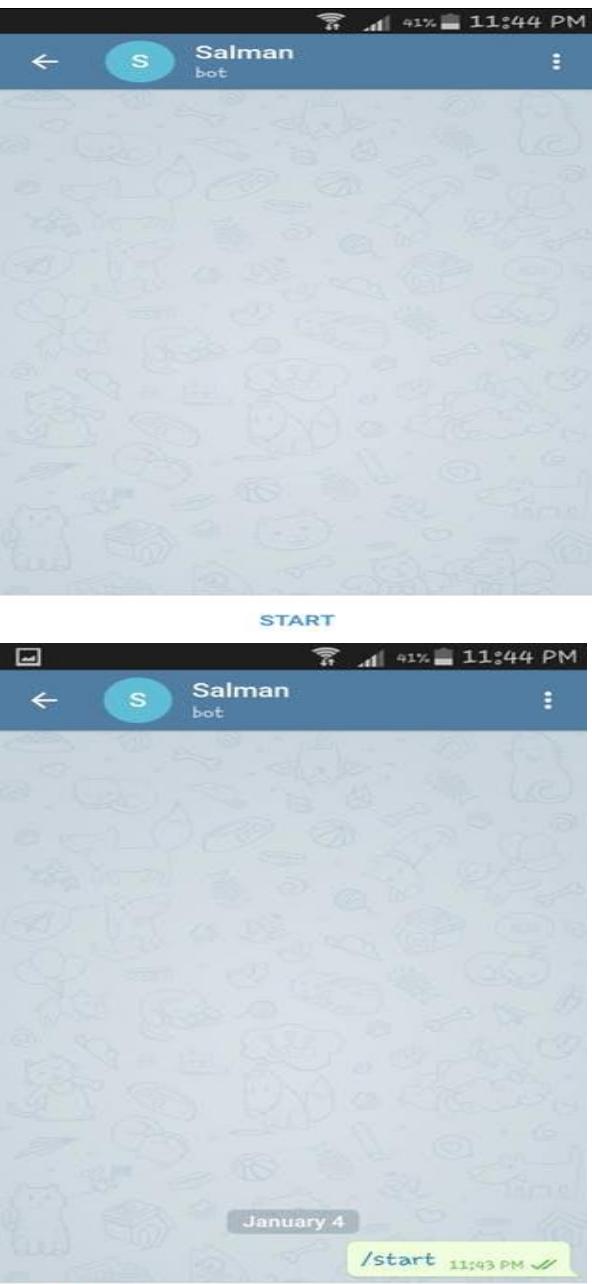
All set, now time to connect the Pi and LED.

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	<u>Setting up Wireless Access Point using Raspberry Pi</u>
	<p>Required Components:</p> <p>The following components will be needed to set up a raspberry pi as a wireless access point:</p> <ol style="list-style-type: none"> 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)
	<p>Steps for Setting up Raspberry Pi as Wireless Access Point:</p> <p>Step 1: Update the Pi</p> <p>As usual, we update the raspberry pi to ensure we have the latest version of everything. This is done using;</p>
	<pre>sudo apt-get update</pre>
	<p>followed by;</p>
	<pre>sudo apt-get upgrade</pre>
	<p>With the update done, reboot your pi to effect changes.</p>
	<p>Step 2: Install “dnsmasq” and “hostapd”</p> <p>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;</p>
	<pre>sudo apt-get install dnsmasq</pre>
	<p>followed by;</p>
	<pre>sudo apt-get install hostapd</pre>
	<p>or you could combine it by running;</p>
	<pre>sudo apt-get install dnsmasq hostapd</pre>
	<p>Step 3: Stop the software from Running</p>

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

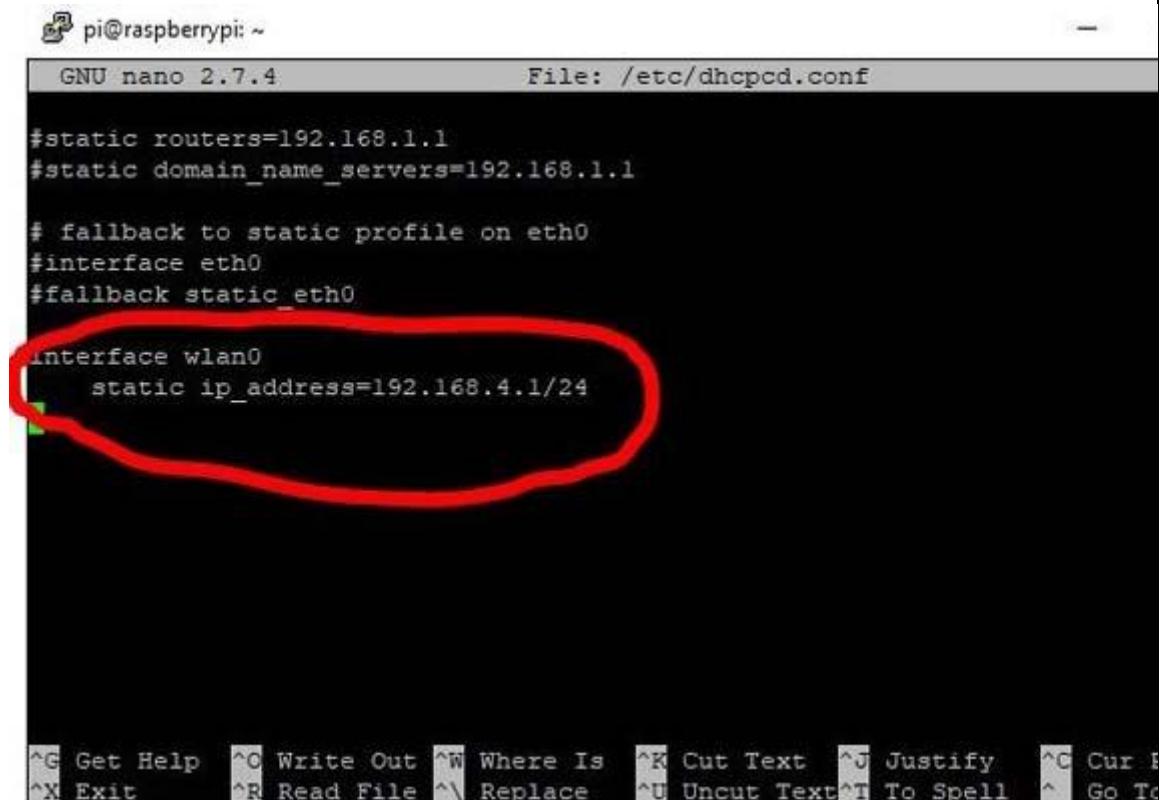
Confirm the *wlan* port on which the wireless device being used is connected. For my Pi, the wireless is on wlan0. **Setting up the Raspberry Pi to act as a server** requires us to assign a static IP address to the wireless port. This can be done by editing the *dhpcd* config file. To edit the configuration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



```
pi@raspberrypi: ~  
GNU nano 2.7.4 File: /etc/dhcpcd.conf  
  
#static routers=192.168.1.1  
#static domain_name_servers=192.168.1.1  
  
# fallback to static profile on eth0  
#interface eth0  
#fallback static_eth0  
  
interface wlan0  
    static ip_address=192.168.4.1/24
```

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 *dhpcd* service to effect the changes made to the configuration using;

```
sudo service dhpcd restart
```

Step 5: Configure the *dhpcd* server

With a static IP address now configured for the Raspberry Pi wlan, the next thing is for us to configure the *dhpcd* 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 one 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 wireless
```

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

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

```
pi@raspberrypi: ~
GNU nano 2.7.4          File: /etc/dnsmasq.conf

interface=wlan0      # Use the require wireless interface - usually wlan0
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h

[ Read 2 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur I
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell ^G Go To
```

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 encryption 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.

A screenshot of a terminal window titled "pi@raspberrypi: ~". The window shows the contents of the file "/etc/hostapd/hostapd.conf" using the "nano" text editor. The configuration file contains the following settings:

```
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=emmanuel  
wpa_key_mgmt=WPA-PSK  
wpa_pairwise=TKIP  
rsn_pairwise=CCMP
```

At the bottom of the terminal window, there is a menu bar with various keyboard shortcuts for navigating the editor. The menu items include: [Read 14 lines], ^G Get Help, ^C Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur E, ^X Exit, ^R Read File, ^\ Replace, ^U Uncut Text, ^T To Spell, ^ Go To.

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.

```
pi@raspberrypi: ~
GNU nano 2.7.4          File: /etc/default/hostapd

Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about alternative
# methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configuration
# file and hostapd will be started during system boot. An example configuration
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"

# Additional daemon options to be appended to hostapd command:-
# -d      show more debug messages (-dd for even more)
# -K      include key data in debug messages
# -t      include timestamps in some debug messages
#
# Note that -B (daemon mode) and -P (pidfile) options are automatically
# configured by the init.d script and must not be added to DAEMON_OPTS.

[ Read 21 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit    ^R Read File ^N Replace ^U Uncut Text ^T To Spell ^L Go To Line
```

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

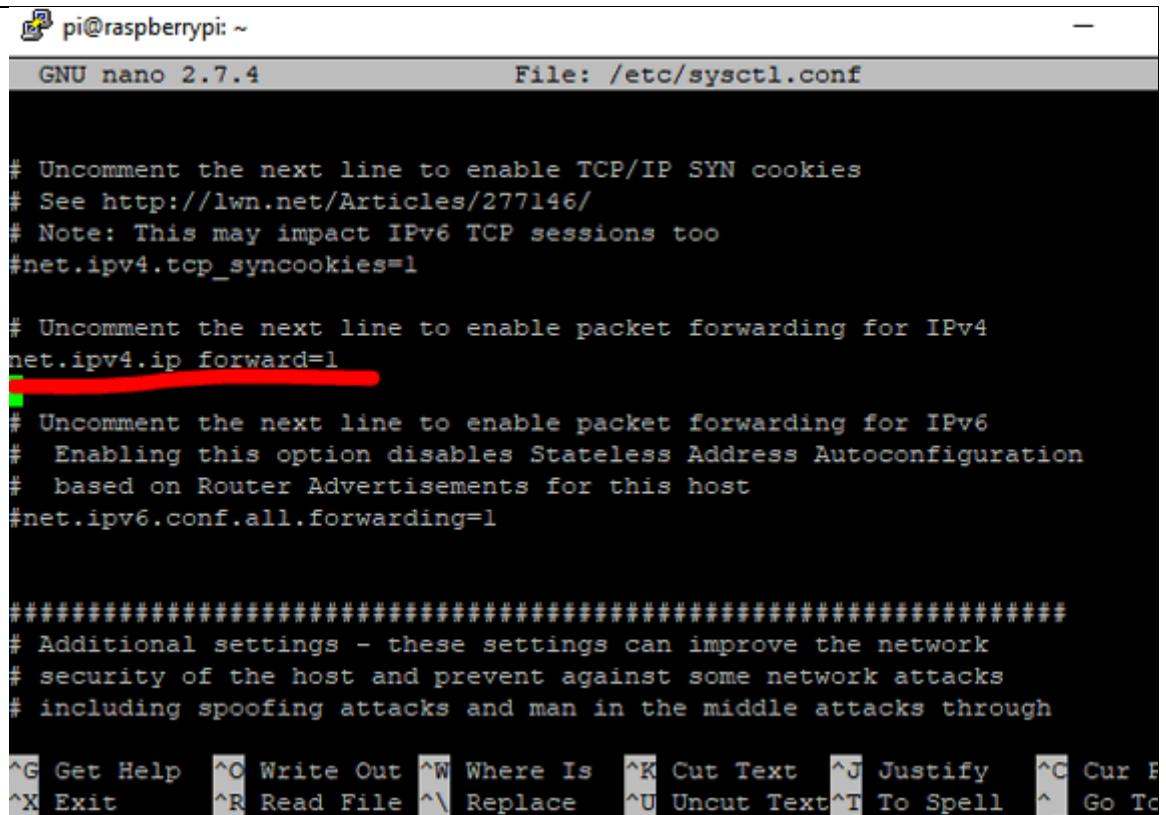
Step 8: Routing and masquerade for outbound traffic

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)



```
pi@raspberrypi: ~
GNU nano 2.7.4          File: /etc/sysctl.conf

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_synccookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur P
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^ Go To
```

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
```

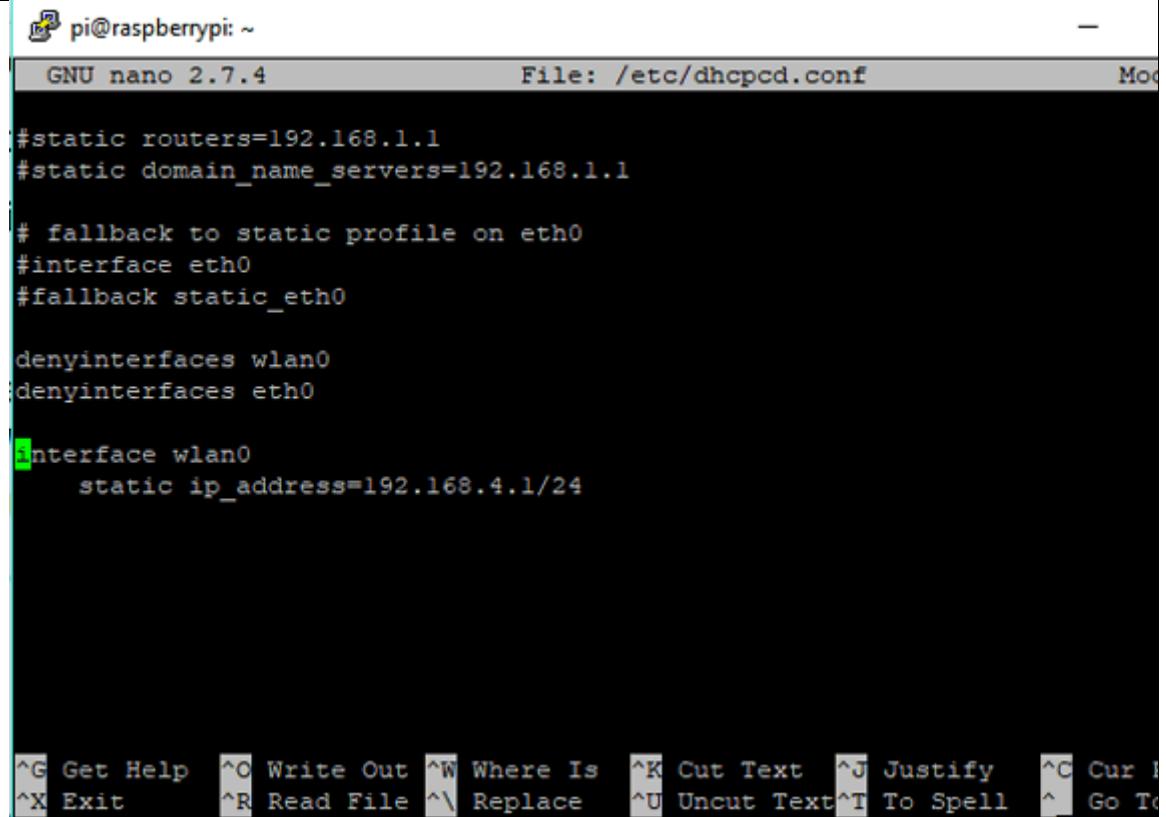
Next, we stop *hostapd* so as to configure the software.

```
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
```



```
pi@raspberrypi: ~
GNU nano 2.7.4          File: /etc/dhcpd.conf      Mod

#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

denyinterfaces wlan0
denyinterfaces eth0

interface wlan0
    static ip_address=192.168.4.1/24

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur I
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^ Go To
```

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 the 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.

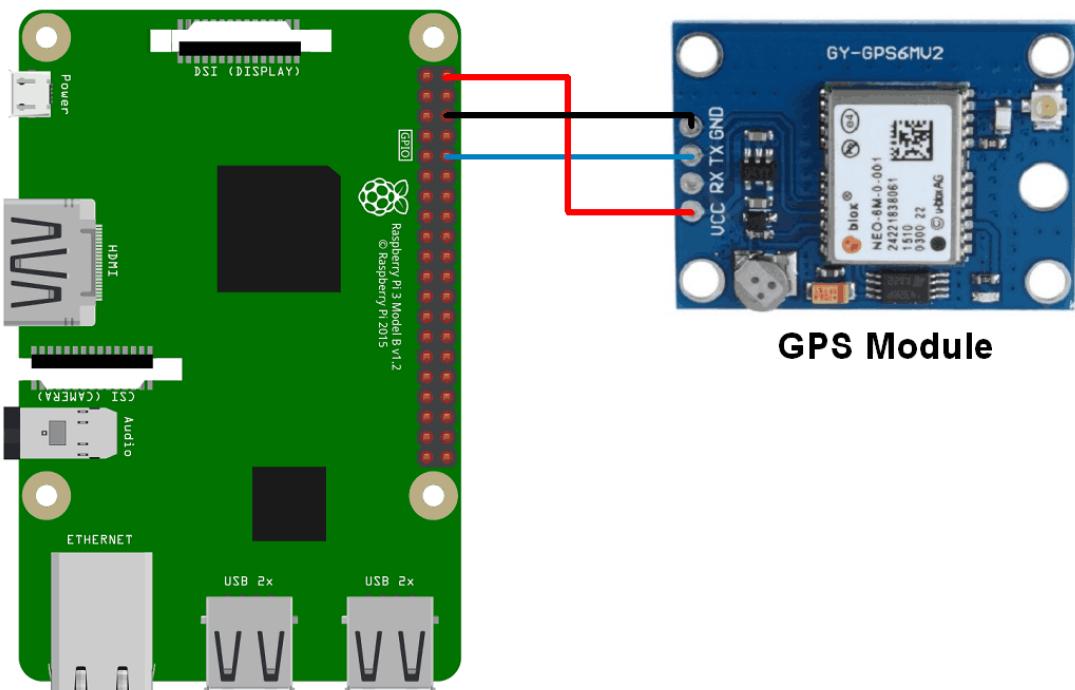
```
interface=wlan0
bridge=br0
ssid=piNetwork
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=mcctest1
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

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.



GPS Module

```
sudo nano /boot/config.txt
#####
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:

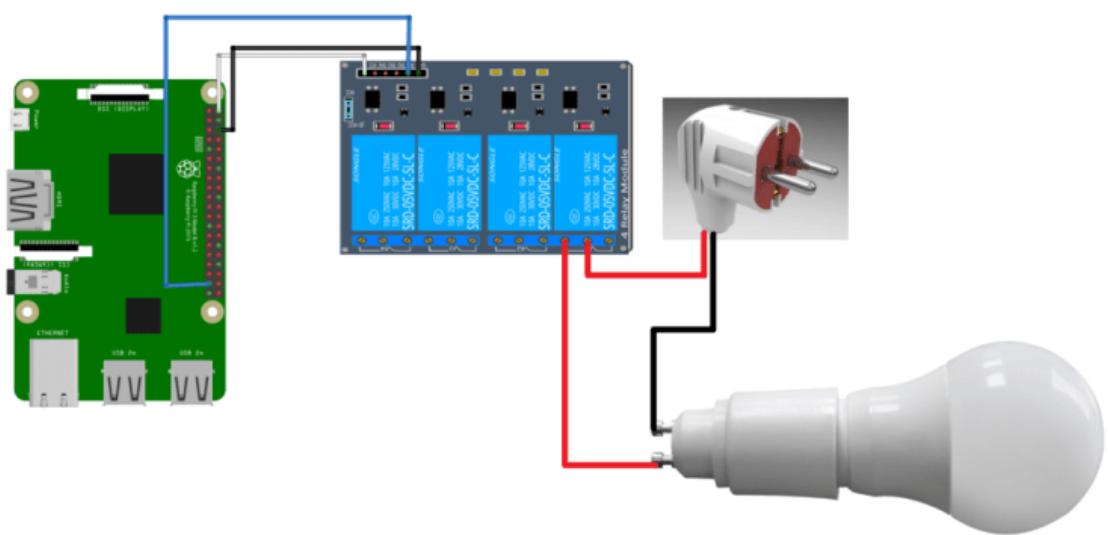
```
import time
import serial
import string
import pynmea2
import RPi.GPIO as gpio
gpio.setmode(gpio.BCM)
port = "/dev/ttyAMA0" # the serial port to which the pi is connected.
```

```
#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.
	 <p>To capture image:</p> <pre>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()</pre> <p>To capture video:</p> <pre>import picamera from time import sleep camera = picamera.PiCamera() camera.resolution = (640, 480)</pre>

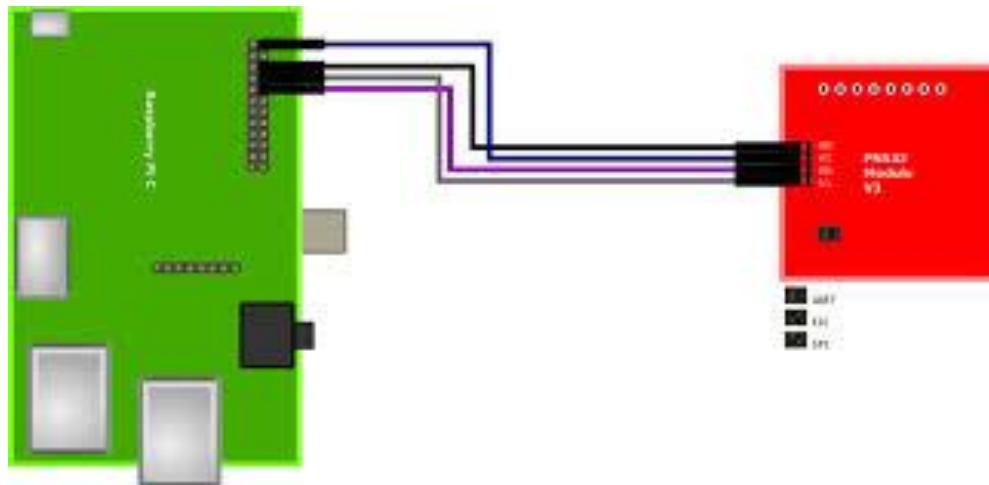
```
print()  
#start recording using pi camera  
camera.start_recording("/home/pi/demo.h264")  
#wait for video to record  
camera.wait_recording(20)  
#stop recording  
camera.stop_recording()  
camera.close()  
print("video recording stopped")
```

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	SEL
0	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:~ $ 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 modulation
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.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 appear the following error

```
pi@raspberrypi:~ $ sudo nano /etc/nfc/libnfc.conf
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 modify 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.

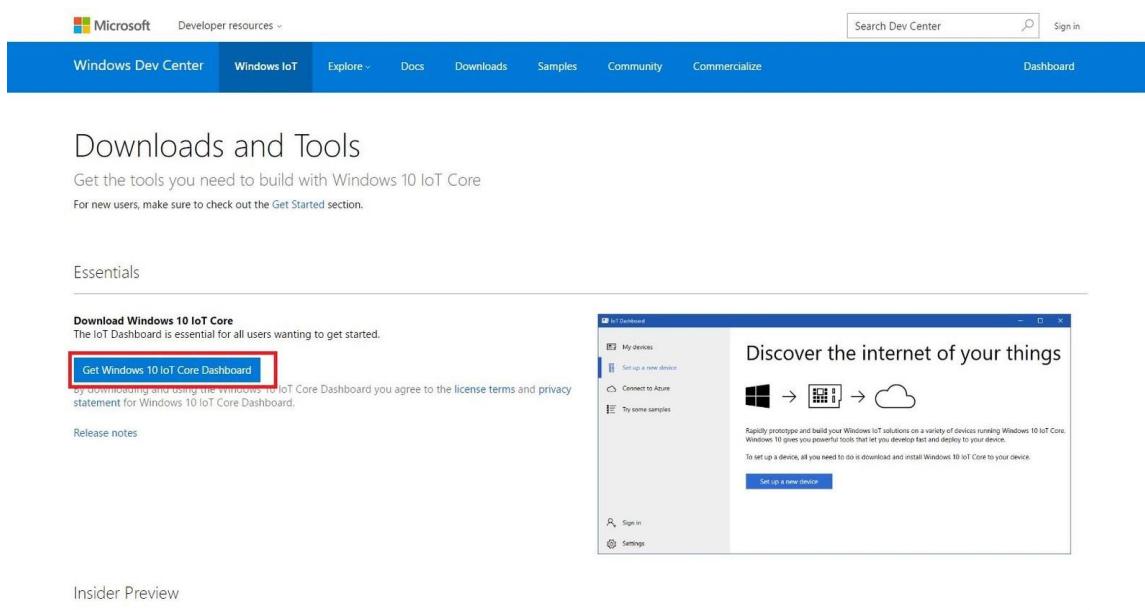
To get up and running you need a few bits and pieces:

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.



The screenshot shows the Microsoft Windows Dev Center homepage. The 'Downloads and Tools' section is highlighted. A red box surrounds the 'Get Windows 10 IoT Core Dashboard' button. Below it, there's a note about accepting license terms and privacy statements. Further down, there's a 'Release notes' link. To the right, a separate window titled 'IoT Dashboard' is displayed, showing a sidebar with 'My devices', 'Set up a new device', 'Connect to Azure', and 'Try some samples'. The main area says 'Discover the internet of things' with a flow diagram from a computer to a building icon to a cloud. It also includes a note about rapidly prototyping IoT solutions and a 'Set up a new device' button.

Install the application and open it.

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

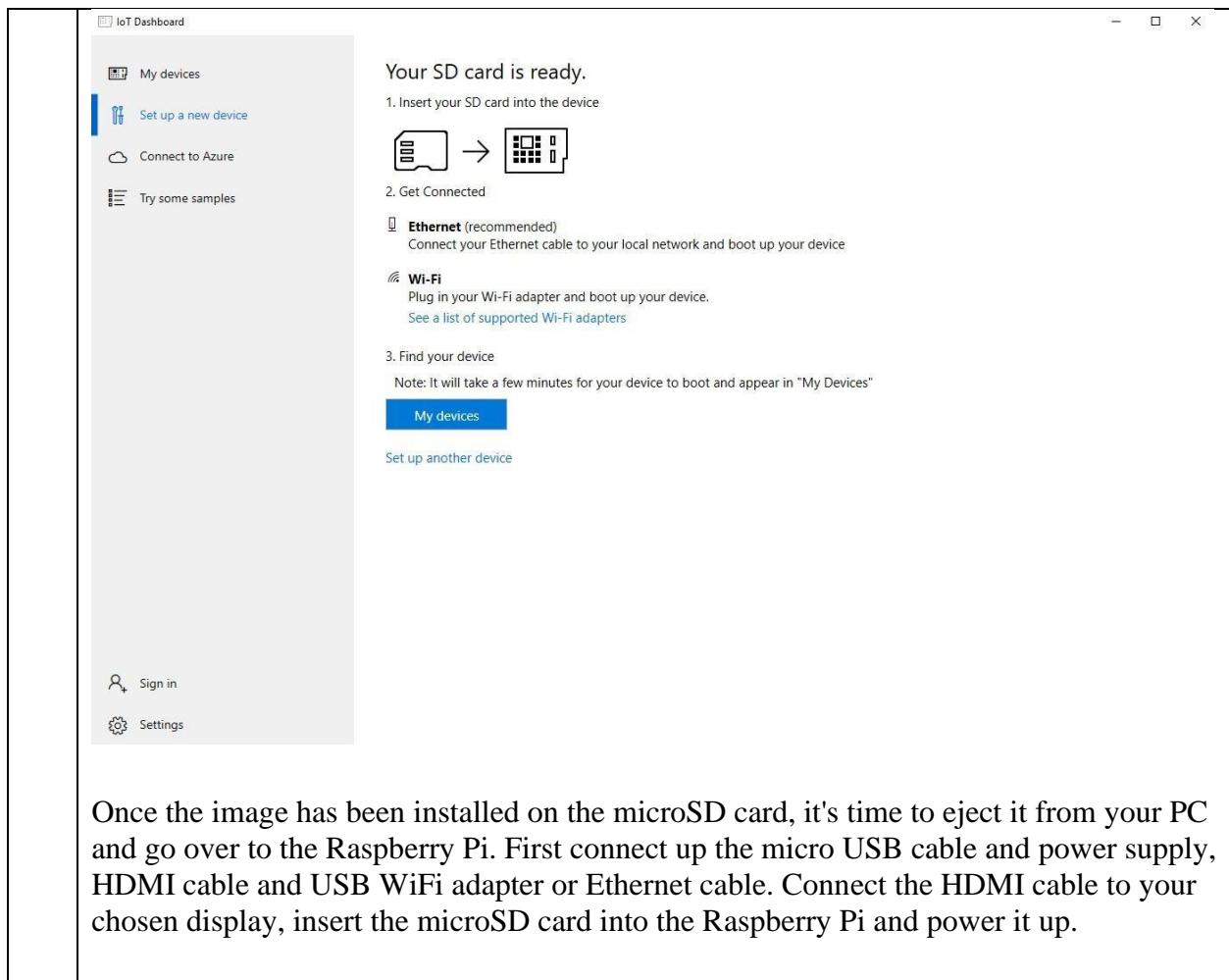
Select the options as shown in the image below. Make sure you select the correct drive for your microSD card and give your device a name and admin password.

The screenshot shows the Microsoft IoT Dashboard interface. On the left, there's a sidebar with options like 'My devices', 'Set up a new device' (which is highlighted with a red box), 'Connect to Azure', and 'Try some samples'. The main area is titled 'Set up a new device' and contains instructions: 'First, let's get Windows 10 IoT Core on your device.' It has several configuration fields: 'Device type' (Raspberry Pi 2 & 3), 'OS Build' (Windows 10 IoT Core), 'Drive' (E: 14Gb [SDHC Card]), 'Device name' (RaspberryPi3), 'New Administrator password' (*****), and 'Confirm Administrator password' (*****). To the right, there's a section for Wi-Fi network selection with a checked checkbox for 'Wi-Fi Network Connection' and a dropdown list showing 'TP-LINK_C587'. A note below says 'Only 2.4 Ghz WiFi networks that have already been connected to will appear in this list'. At the bottom, there's a link to 'View software license terms', a checked checkbox for 'I accept the software license terms', and a blue 'Download and install' button.

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.



Deep learning

Sr.No	Practical name	Pg.No
1	Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow	110
2	Solving XOR problem using deep feed forward network.	111
3	Implementing deep neural network for performing binary classification task.	113
4	a) Aim: Using deep feed forward network with two hidden layers for performing multiclass classification and predicting the class. b) Aim: Using a deep feed forward network with two hidden layers for performing classification and predicting the probability of class. c) Aim: Using a deep feed forward network with two hidden layers for performing linear regression and predicting values.	115 117 118
5	a) Evaluating feed forward deep network for regression using KFold cross validation. b) Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.	119 120
6	Implementing regularization to avoid overfitting in binary classification.	125
7	Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.	130
8	Performing encoding and decoding of images using deep autoencoder.	133
9	Implementation of convolutional neural network to predict numbers from number images	136
10	Denoising of images using autoencoder.	139

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".format(e_matrix_A))  
  
eigen_values_A,eigen_vectors_A=tf.linalg.eigh(e_matrix_A)  
  
print("Eigen Vectors:\n{ }\nEigen Values:\n{ }\n".format(eigen_vectors_A,eigen_values_A))
```

OUTPUT:

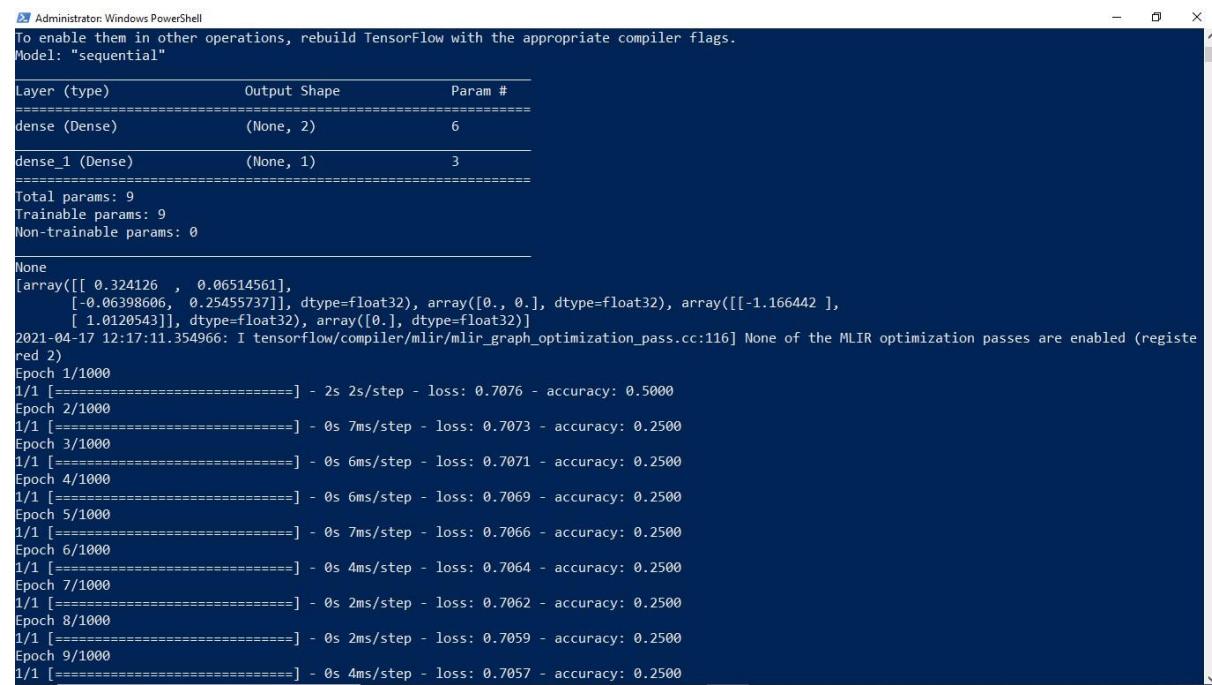
```
tf.Tensor(  
[[1 2 3]  
 [4 5 6]], shape=(2, 3), dtype=int32)  
tf.Tensor(  
[[ 7  8]  
 [ 9 10]  
 [11 12]], shape=(3, 2), dtype=int32)  
Product: tf.Tensor(  
[[ 58  64]  
 [139 154]], shape=(2, 2), dtype=int32)  
Matrix A:  
[[7.791751  6.3527837]  
 [6.8659496 5.229142 ]]  
  
Eigen Vectors:  
[[-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=['accuracy'])  
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  
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.  
Model: "sequential"  


| Layer (type)    | Output Shape | Param # |
|-----------------|--------------|---------|
| dense (Dense)   | (None, 2)    | 6       |
| dense_1 (Dense) | (None, 1)    | 3       |

  
Total params: 9  
Trainable params: 9  
Non-trainable params: 0  
  
None  
[array([[ 0.324126 ,  0.06514561],  
       [-0.06398606,  0.25455737]], dtype=float32), array([0., 0.], dtype=float32), array([-1.166442 ,  
       [ 1.0120543]], dtype=float32), array([0.], dtype=float32)]  
2021-04-17 12:17:11.354966: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)  
Epoch 1/1000  
1/1 [=====] - 2s 2ms/step - loss: 0.7076 - accuracy: 0.5000  
Epoch 2/1000  
1/1 [=====] - 0s 7ms/step - loss: 0.7073 - accuracy: 0.2500  
Epoch 3/1000  
1/1 [=====] - 0s 6ms/step - loss: 0.7071 - accuracy: 0.2500  
Epoch 4/1000  
1/1 [=====] - 0s 6ms/step - loss: 0.7069 - accuracy: 0.2500  
Epoch 5/1000  
1/1 [=====] - 0s 7ms/step - loss: 0.7066 - accuracy: 0.2500  
Epoch 6/1000  
1/1 [=====] - 0s 4ms/step - loss: 0.7064 - accuracy: 0.2500  
Epoch 7/1000  
1/1 [=====] - 0s 2ms/step - loss: 0.7062 - accuracy: 0.2500  
Epoch 8/1000  
1/1 [=====] - 0s 2ms/step - loss: 0.7059 - accuracy: 0.2500  
Epoch 9/1000  
1/1 [=====] - 0s 4ms/step - loss: 0.7057 - accuracy: 0.2500
```

```
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  
>>>
```

```
Administrator: Windows PowerShell  
>>> dataset=loadtxt('pima-indians-diabetes.csv',delimiter=',')  
>>> dataset  
array([[ 6. , 148. , 72. , ... , 0.627, 50. , 1. ],  
 [ 1. , 85. , 66. , ... , 0.351, 31. , 0. ],  
 [ 8. , 183. , 64. , ... , 0.672, 32. , 1. ],  
 ...,  
 [ 5. , 121. , 72. , ... , 0.245, 30. , 0. ],  
 [ 1. , 126. , 60. , ... , 0.349, 47. , 1. ],  
 [ 1. , 93. , 70. , ... , 0.315, 23. , 0. ]])  
>>> X=dataset[:,0:8]  
>>> Y=dataset[:,8]  
>>> X  
array([[ 6. , 148. , 72. , ... , 33.6 , 0.627, 50. ],  
 [ 1. , 85. , 66. , ... , 26.6 , 0.351, 31. ],  
 [ 8. , 183. , 64. , ... , 23.3 , 0.672, 32. ],  
 ...,  
 [ 5. , 121. , 72. , ... , 26.2 , 0.245, 30. ],  
 [ 1. , 126. , 60. , ... , 30.1 , 0.349, 47. ],  
 [ 1. , 93. , 70. , ... , 30.4 , 0.315, 23. ]])  
>>>  
>>> Y  
array([1., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 1.,  
 1., 0., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 0.,  
 0., 0., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0.,  
 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0.,  
 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1.,  
 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1.,
```

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)
```

```
>>> 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 (registered 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:

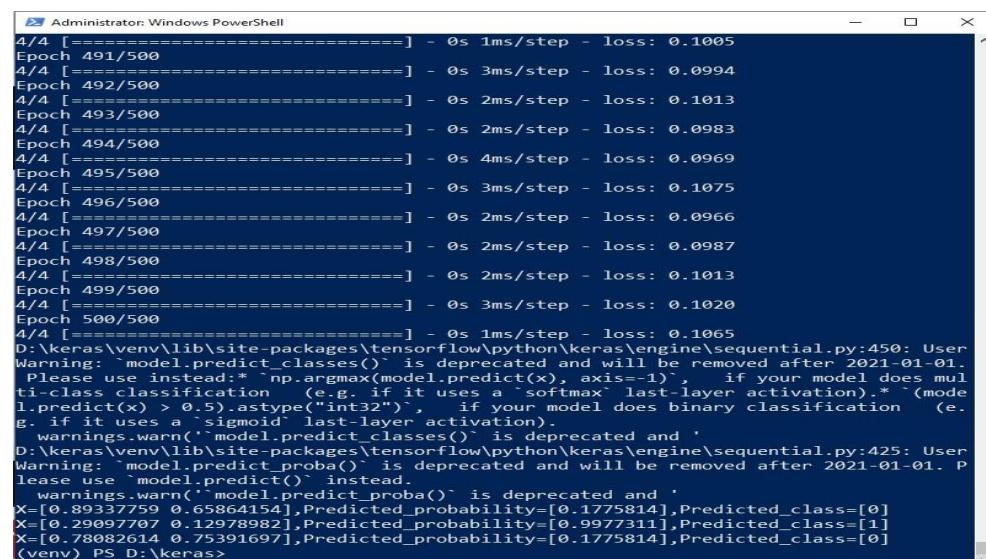
```
Administrator: Windows PowerShell
4/4 [=====] - 0s 2ms/step - loss: 0.6935
Epoch 488/500
4/4 [=====] - 0s 2ms/step - loss: 0.6927
Epoch 489/500
4/4 [=====] - 0s 3ms/step - loss: 0.6931
Epoch 490/500
4/4 [=====] - 0s 3ms/step - loss: 0.6928
Epoch 491/500
4/4 [=====] - 0s 2ms/step - loss: 0.6938
Epoch 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: User
Warning: `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 mul
ti-class classification (e.g. if it uses a `softmax` last-layer activation). * `(mode
l.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:



```
Administrator: Windows PowerShell
4/4 [=====] - 0s 1ms/step - loss: 0.1005
Epoch 491/500
4/4 [=====] - 0s 3ms/step - loss: 0.0994
Epoch 492/500
4/4 [=====] - 0s 2ms/step - loss: 0.1013
Epoch 493/500
4/4 [=====] - 0s 2ms/step - loss: 0.0983
Epoch 494/500
4/4 [=====] - 0s 4ms/step - loss: 0.0969
Epoch 495/500
4/4 [=====] - 0s 3ms/step - loss: 0.1075
Epoch 496/500
4/4 [=====] - 0s 2ms/step - loss: 0.0966
Epoch 497/500
4/4 [=====] - 0s 2ms/step - loss: 0.0987
Epoch 498/500
4/4 [=====] - 0s 2ms/step - loss: 0.1013
Epoch 499/500
4/4 [=====] - 0s 3ms/step - loss: 0.1020
Epoch 500/500
4/4 [=====] - 0s 1ms/step - loss: 0.1065
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 '
D:\keras\venv\lib\site-packages\tensorflow\python\keras\engine\sequential.py:425: UserWarning: `model.predict_proba()` is deprecated and will be removed after 2021-01-01. Please use `model.predict()` instead.
  warnings.warn(`model.predict_proba()` is deprecated and '
X=[0.89337759 0.65864154],Predicted_probability=[0.1775814],Predicted_class=[0]
X=[0.29097707 0.12978982],Predicted_probability=[0.9977311],Predicted_class=[1]
X=[0.78082614 0.75391697],Predicted_probability=[0.1775814],Predicted_class=[0]
(venv) PS D:\keras>
```

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,noisy=0.1,random_state=1)  
scalar,scalarY=MinMaxScaler(),MinMaxScaler()
```

```
scalar.fit(X)  
scalarY.fit(Y.reshape(100,1))  
X=scalar.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,noisy=0.1,random_state=1)  
Xnew=scalar.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	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	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.]
[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)

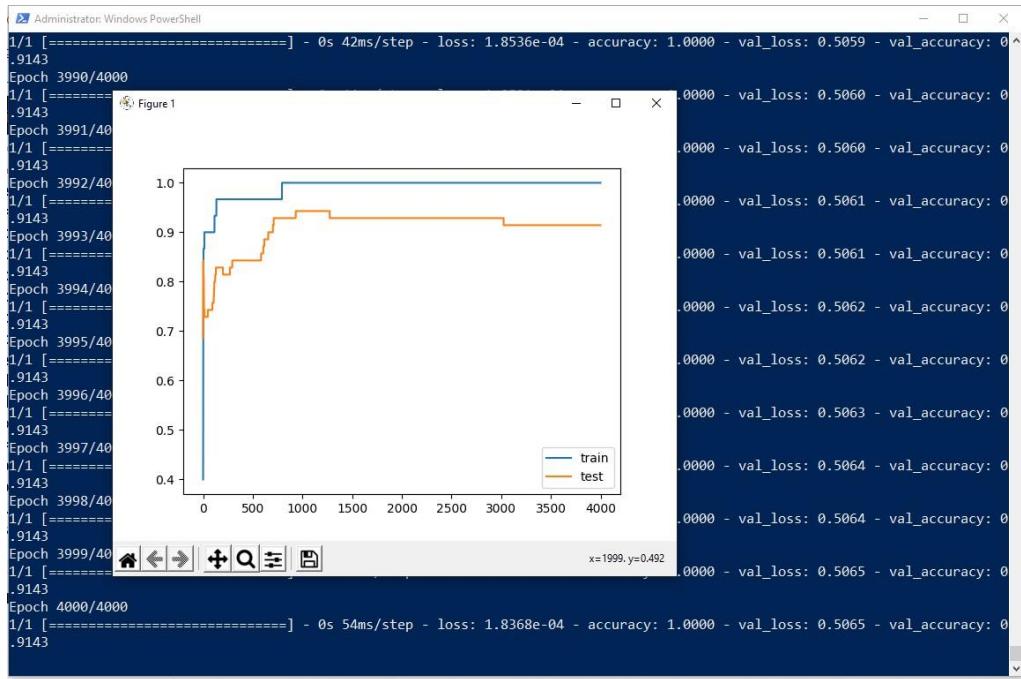
```
model.add(Dense(10,input_dim=4,activation='relu'))  
3/3 [=====] - 0s 999us/step - loss: 0.1436 - accuracy: 1.0000  
Baseline: 98.67% (2.67%)
```

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,noisy=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=['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()
```

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,noisy=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'))

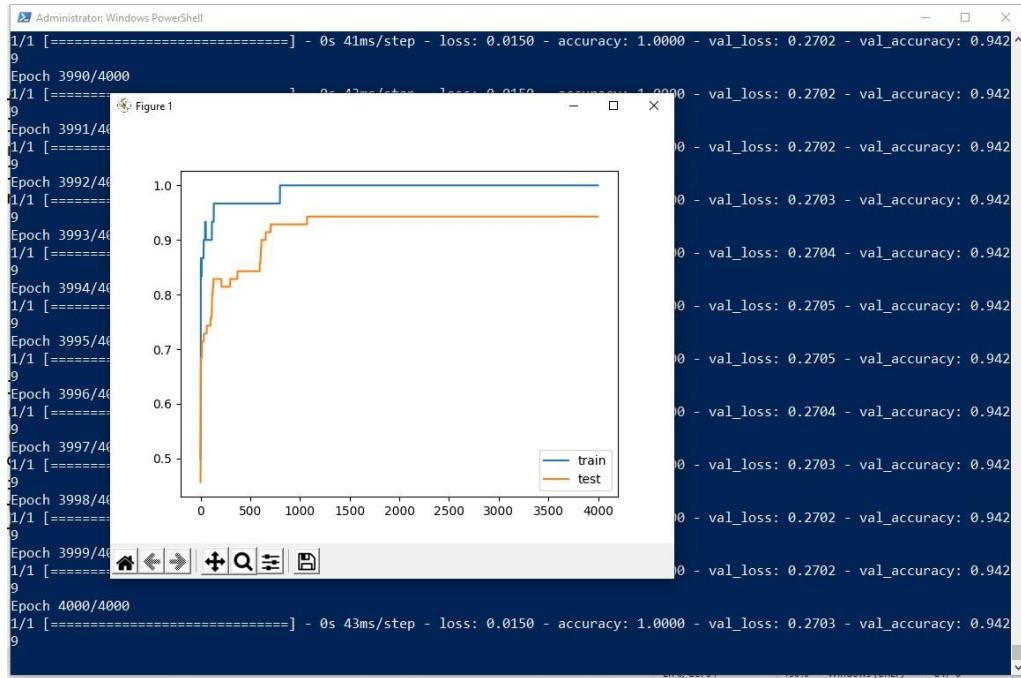
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

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

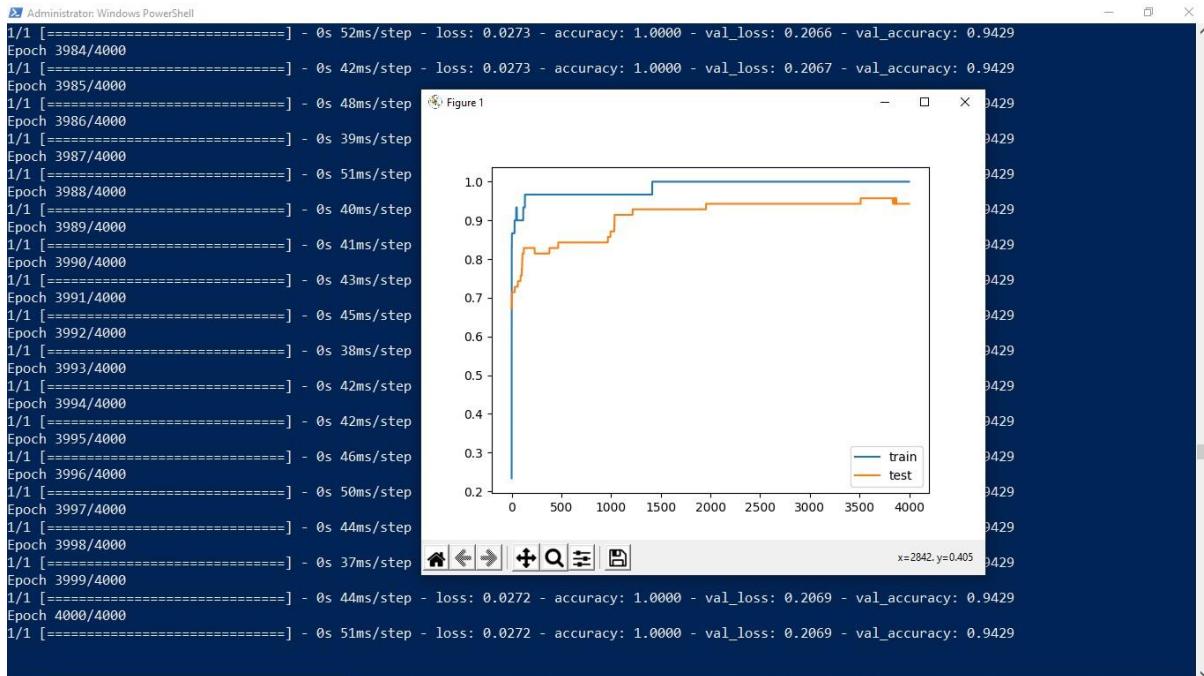
pyplot.plot(history.history['accuracy'],label='train') pyplot.plot(history.history['val_accuracy'],label='test')

pyplot.legend() pyplot.show()

```



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

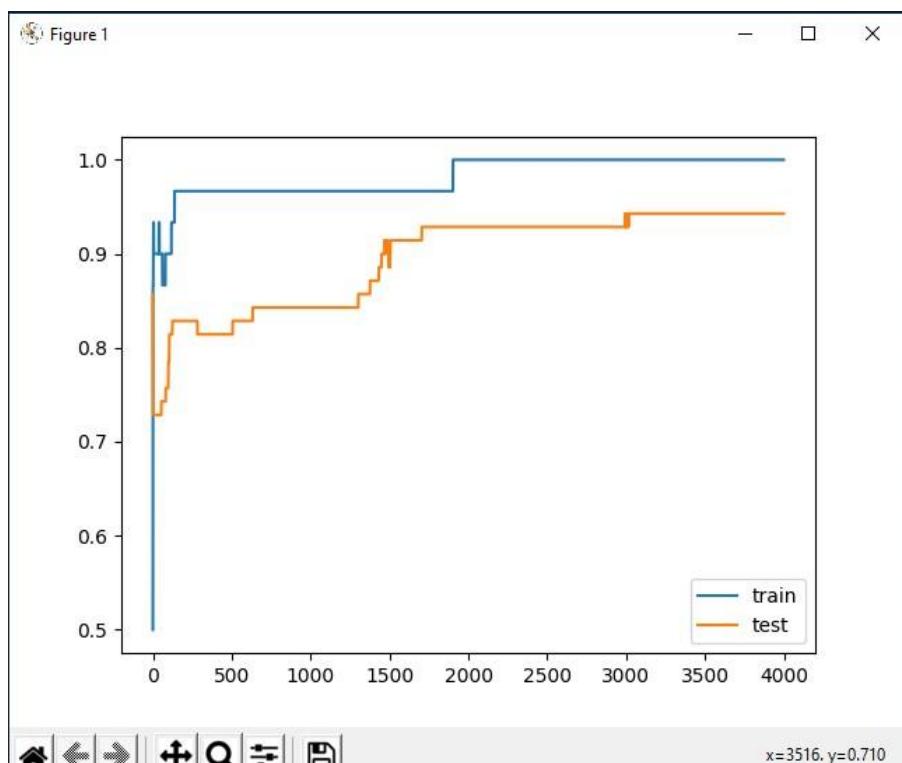


By applying l1 and l2 regularizer we can observe the following changes in accuracy of both training and testing data. The changes in code are also highlighted.

```
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,noisy=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=['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()
```

OUTPUT:



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.csv')
#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

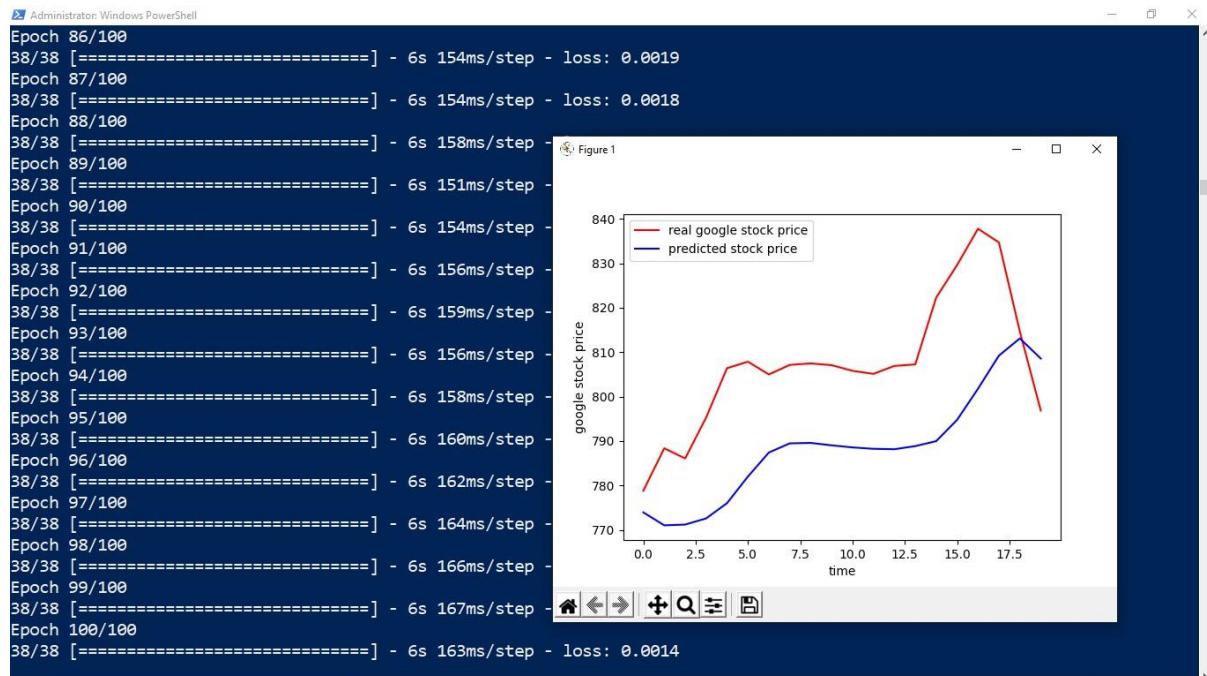
dataset_total=pd.concat((dataset_train['Open'],dataset_test['Open']),axis=0)
inputs=dataset_total[len(dataset_total)-len(dataset_test)-60:].values
inputs=inputs.reshape(-1,1)
inputs=sc.transform(inputs)
X_test=[]
for i in range(60,80):
    X_test.append(inputs[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

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:



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)
```

```

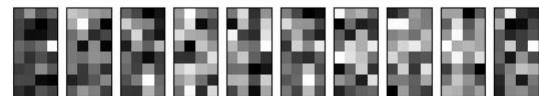
#train autoencoder with training dataset
autoencoder.fit(X_train,X_train,
epochs=50,           batch_size=256,
shuffle=True,
validation_data=(X_test,X_test))
encoded_imgs=encoder.predict(X_test)
decoded_imgs=decoder.predict(encoded_im
gs) import matplotlib.pyplot as plt n = 10 #
How many digits we will 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
235/235 [=====]
Figure 1
Epoch 37/50
235/235 [=====]
Epoch 38/50
235/235 [=====]
Epoch 39/50
235/235 [=====]
Epoch 40/50
235/235 [=====]
Epoch 41/50
235/235 [=====]
Epoch 42/50
235/235 [=====]
Epoch 43/50
235/235 [=====]
Epoch 44/50
235/235 [=====]
Epoch 45/50
235/235 [=====]
Epoch 46/50
235/235 [=====]
Epoch 47/50
235/235 [=====] - 3s 14ms/step - loss: 0.0929 - val_loss: 0.0915
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
```

7 2 1 0 4 1 4 9 5 9



7 2 1 0 4 1 4 9 5 9

Practical No:9

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

```
from keras.datasets import mnist
keras.utils import to_categorical
keras.models import Sequential
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',metric
s=['accuracy'])

#train

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

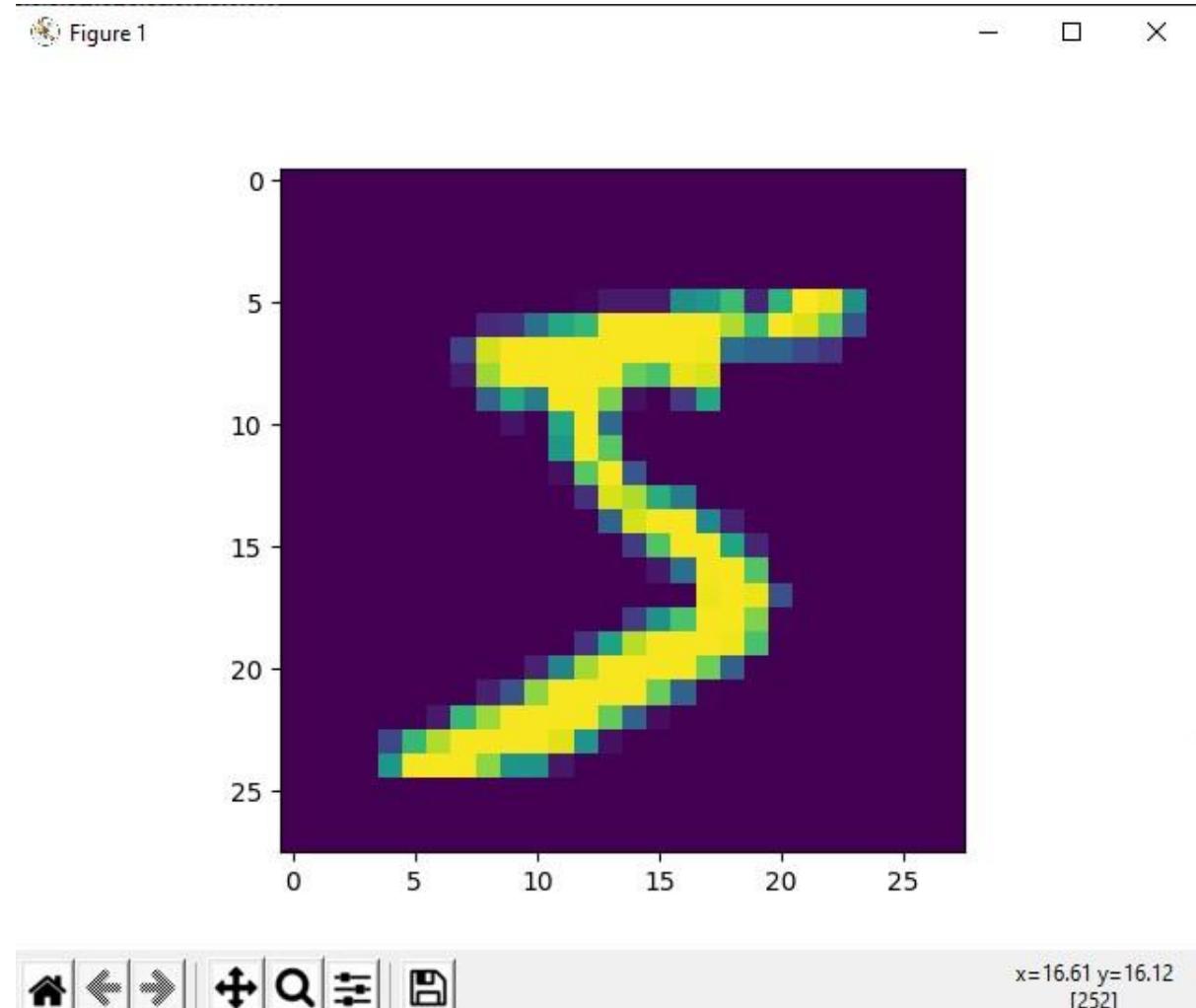
print(model.predict(X_test[:4]))

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

print(Y_test[:4])

```

OUTPUT:



```

(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. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [1. 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:

