

# BLOCKCHAIN

## INDEX

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

## Practical 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(se
            lf._private_key)

    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER'))
            ).decode('ascii')
Dinesh = Client()
print (Dinesh.identity)
```

**Output:**



30819f300d06092a864886f70d0101050003818d0030818902818100eea39d3e40f737cff2050d40515a4833c80cdd9073d88a3629aef45ee77793c78197eb0f8bbf0688c0672e22ee74e691bd53668

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

**Code:**

```
class Transaction:  
    def __init__(self,  
                 sender,  
                 recipient,  
                 value):  
        self.sender =  
        sender  
        s  
        e  
        l  
        f  
        .  
        r  
        e  
        c  
        i  
        p  
        i  
        e  
        n  
        t  
        =  
        r  
        e  
        c  
        i  
        p  
        i  
        e  
        n  
        t  
        s  
        e  
        l  
        f  
        .  
        v  
        a  
        l  
        u  
        e  
        =  
        v  
        a  
        l
```

```

u           e
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:**

0f18c12abcf2904299162ce5c39689df19a9bf583689ec58182cac42617da5b1282d467b6a92ffd358f33052abbeefd502d1bbfcfcde293962e8ecec7b9f6a249df1bfcacf537a19404913e73079fc58c1f

### **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: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a
-----
value: 15.0
-----
time: 2022-07-30 17:35:06.809777
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a72cfb527dff9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2b
-----
value: 6.0
-----
time: 2022-07-30 17:35:06.811866
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a81
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d578
-----
value: 2.0
-----
time: 2022-07-30 17:35:06.814247
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a81
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d578
-----
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, previous_hash):
        self.previous_hash = previous_hash
        self.timestamp = time.time()
        self.data = []
        self.nonce = 0
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        sha = hashlib.sha256()
        hashable_data = str(self.previous_hash).encode() + str(self.timestamp).encode() + str(self.data).encode() + str(self.nonce).encode()
        sha.update(hashable_data)
        return sha.hexdigest()

    def mine(self, target):
        while self.hash[:len(target)] != target:
            self.nonce += 1
            self.hash = self.calculate_hash()
        print("Block mined with nonce: " + str(self.nonce))

    def __str__(self):
        return f"Block Hash: {self.hash}\nPrevious Hash: {self.previous_hash}\nTimestamp: {self.timestamp}\nData: {self.data}\nNonce: {self.nonce}\nHash: {self.hash}
```

```
c          o
t          c
i          k
o          -
n          h
s          a
=
[          s
]          h
s          =
e          "
e          "
l          s
f          e
.
p          l
r          f
e          .
v          N
i          o
o          n
u          c
s          e
-
b          =
l          "
          "
last_block_hash = ""

Dinesh = Client()

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

block0 = Block()

block0.previous_block_hash = None
Nonce = None

block0.verified_transactions.append (t0)

digest = hash (block0)
last_block_hash = digest

TPCoins = []

def dump_blockchain (self):
    print ("Number of blocks in the chain: " + str(len (self)))
```

```

for x in range (len(TPCoins)):
    block_temp = TPCoins[x]
    print ("block # " + str(x))
    for transaction in block_temp.verified_transactions:
        display_transaction (transaction)
    print ('- -----')
    print ('=====') 

TPCoins.append (block0)
dump_blockchain(TPCoins)

```

**Output:**

---

```

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



**1e) Create a mining function and test it.**

**Code:**

```

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

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

mine ("test message", 2)

```

**Output:**

```
after 238 iterations found nonce: 1124711db6185591392c6a06c24a3c2ebbaeca647fb8374824f939ada27c09e9
after 353 iterations found nonce: 11e0a4b57bb76496ecc6ab5a3c126165bc9dbbaf80f664e7e192a422360c3884
after 419 iterations found nonce: 11280dfd9ab05b3dbfa869990153732941408faa4a3b0832819b161f52321c08
after 511 iterations found nonce: 1150944bd7ea429acd052da390b148f27eb881686e0f8bfcd77f833af878e2e
after 822 iterations found nonce: 110e61d41d94b48f6dfcb6f43f9ceafe2ab7168456dab088e05126d43d0366ef
after 924 iterations found nonce: 11147bf32bafeee4505b5cc40c3b227366d8a41ab3bd740e437c74400b7a8127
'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: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa

```

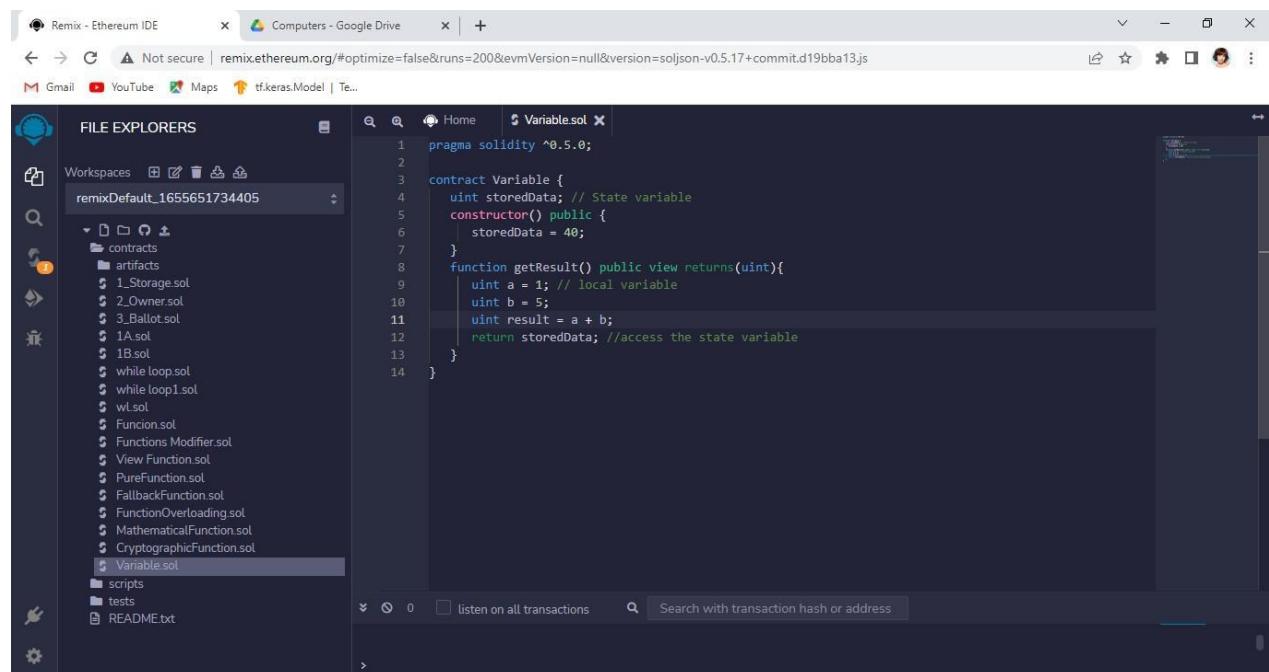
## Practical 2

**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 Ethereum IDE (Remix) interface. The left sidebar displays a file explorer with 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, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, and Variable.sol. The "Variable.sol" file is currently selected and open in the main editor area. The code in the editor is as follows:

```
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:**

### Operators:

**Arithmetic Operator**

**Code:**

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files: wLsol, Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, 1A.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, Libraries.sol, Assembly.sol, Events.sol, ErrorHandling.sol, and ArithmeticOperator.sol. The "ArithmeticOperator.sol" file is selected. The main editor area displays the Solidity code for the "ArithmeticOperator" contract. The code defines a public variable "a" (uint16) with a value of 50, and a public variable "b" (uint16) with a value of 20. It includes functions for addition, subtraction, multiplication, division, modulus, negation, and increment/decrement operations. The bottom status bar shows transaction logs and debug buttons.

```
pragma solidity ^0.5.0;
contract ArithmeticOperator {
    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:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. This sidebar lists various function names: a, b, dec, diff, div, inc, mod, mul, and sum, each associated with a placeholder input field. The main editor area shows the same Solidity code as the previous screenshot. The bottom status bar shows transaction logs for calls to the "mul" and "sum" functions, along with debug buttons.

```
pragma solidity ^0.5.0;
contract ArithmeticOperator {
    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:

FILE EXPLORER

- Funcion.sol
- Functions Modifier.sol
- View Function.sol
- PureFunction.sol
- FallbackFunction.sol
- FunctionOverloading.sol
- MathematicalFunction.sol
- CryptographicFunction.sol
- Variable.sol
- BitwiseOperator.sol
- 1A.sol
- AssignmentOperator.sol
- ArithmeticOperator.sol
- WithdrawalPattern.sol
- RestrictedAccess.sol
- Contracts.sol
- Inheritance.sol
- Constructors.sol
- AbstractConstructor.sol
- Interfaces.sol
- Libraries.sol
- Assembly.sol
- Events.sol
- ErrorHandling.sol
- ArithmeticOperator.sol
- RelationalOperator.sol
- scripts
- tests
- README.txt

RelationalOperator.sol

```
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;
}
```

ContractDefinition RelationalOperator → 1 reference(s) ▾

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.leseq() data: 0xbc2...958e7 Debug

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.noteq() data: 0x6a8...a6ab5 Debug

## Output:

DEPLOY & RUN TRANSACTIONS

| Op    | Value         | Type | Output         |
|-------|---------------|------|----------------|
| a     |               |      |                |
| b     | 0: uint16: 70 |      |                |
| eq    | 0: uint16: 10 |      | 0: bool: false |
| gtr   |               |      | 0: bool: true  |
| gtreq |               |      | 0: bool: true  |
| les   |               |      | 0: bool: false |
| leseq |               |      | 0: bool: false |
| noteq |               |      | 0: bool: true  |

RelationalOperator.sol

```
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;
}
```

ContractDefinition RelationalOperator → 1 reference(s) ▾

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.leseq() data: 0xbc2...958e7 Debug

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.noteq() data: 0x6a8...a6ab5 Debug

## Logical Operator

### Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files. In the center, the code editor displays the file `LogicalOperator.sol` with the following content:

```
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);
    }
}
```

Below the code editor, the ContractDefinition RelationalOperator section shows 1 reference(s). The transaction history pane shows a single call to the `Logic` function with parameters `a: true`, `b: false`. The output of the call is `[call] from: 0x5B38D0a6a701c568545dCfcB03FcB875f56beddC4 to: LogicalOperator.Logic(bool,bool) data: 0xc94...00000`.

## Output:

The screenshot shows the Remix Ethereum IDE interface with the DEPLOY & RUN TRANSACTIONS sidebar open. The Deploy button is highlighted. The code editor shows the same `LogicalOperator.sol` file as before. The Logic section of the sidebar shows inputs `a: true` and `b: false`, with a `call` button. The transaction history pane shows the same call to the `Logic` function with the same output.

The screenshot shows the Remix Ethereum IDE interface. On the left is the FILE EXPLORER sidebar, which lists various Solidity files under the contracts folder, including 1\_Storage.sol, 2\_Owner.sol, 3\_Ballot.sol, while\_loop.sol, while\_loop1.sol, wLsol, 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, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, and Libraries.sol. The BitwiseOperator.sol file is currently selected and open in the main code editor window. The code defines a contract named BitwiseOperator with several public functions that perform bit operations on uint16 variables a and b.

```
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. This sidebar displays the deployed contract address (BITWISEOPERATOR AT 0x...), along with the names and types of the public functions: a (uint16:20), and (uint16:16), b (uint16:50), leftshift (uint16:0), not (uint16:65515), or (uint16:54), rightshift (uint16:0), and xor (uint16:38). The main code editor window shows the same BitwiseOperator contract code as in the previous screenshot. At the bottom of the interface, a transaction log is visible, indicating a successful call to the constructor of the LogicalOperator contract.

```
[vm] from: 0x5B3...eddC4 to: LogicalOperator.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x5df...62867
```

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main editor area displays the "AssignmentOperator.sol" contract code:

```

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 = 50;
    uint16 public assignment_mod = 32;

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

```

Below the code, a tooltip indicates "ContractDefinition BitwiseOperator" and "1 reference(s)". The bottom status bar shows a transaction log: "[vm] From: 0x5B3...edc4 to: AssignmentOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xdal...5c92c".

## Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. It displays the "ASSIGNMENTOPERATOR AT 0x9D" contract, which has been deployed. The sidebar lists the following functions:

- getResult (orange button)
- assignment
- assignment\_a...
- o: uint16: 20
- assignment\_d...
- o: uint16: 60
- assignment\_m...
- o: uint16: 5
- assignment\_m...
- o: uint16: 12
- assignment\_m...
- o: uint16: 100
- assignment\_s...
- o: uint16: 30

The main editor area shows the same "AssignmentOperator.sol" code as the previous screenshot. A tooltip below the code indicates "ContractDefinition BitwiseOperator" and "1 reference(s)". The bottom status bar shows a transaction log: "[call] from: 0x5B38D0a6a701c568545dCfcB03FcB875f56beddC4 to: AssignmentOperator.assignment\_sub() data: 0x656...c8bd4 hash: 0x656...c8bd4".

## Loops

### While Loop

#### Code:

```
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:

```
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;
    }
}
```

## Dowhile loop

### Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files such as 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, LogicalOperator.sol, Dowhileloop.sol, scripts, tests, and README.txt. The main code editor window displays the content of Dowhileloop.sol:

```
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;
    }
}
```

At the bottom of the code editor, there is a status bar showing transaction details: [vm] from: 0x583...edd4 to: Dowhileloop.loop() 0xd91...39138 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x3d7...90655. A "Debug" button is also present in the status bar.

## Output:

The screenshot shows the Remix Ethereum IDE interface with the DEBUGGER tab selected. The left sidebar displays the Solidity Locals and Solidity State. The Solidity State section shows the variable data as an array of uint256 with length 8, containing values from 0 to 7. The main code editor window shows the same Solidity code as before. At the bottom, the status bar shows the same transaction details and a "Debug" button.

## For loop

### Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files such as 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, LogicalOperator.sol, Dowhileloop.sol, ForLoop.sol, scripts, tests, and README.txt. The main central area displays the Solidity code for the ForLoop contract:

```
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;
    }
}
```

Below the code editor, the status bar shows: ContractDefinition ForLoop → 1 reference(s) ▾ ▾, 0 listen on all transactions, Search with transaction hash or address, [vm] from: 0x583...eddC4 to: ForLoop.loop() 0xf8e...9f8e8 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x724...0add1, and a Debug button.

## Output:

The screenshot shows the Remix Ethereum IDE interface with the DEBUGGER tab selected. The left sidebar includes the DEBUGGER tab, Stop debugging button, Function Stack (showing 0: loop()), Solidity Locals, and Solidity State (showing data: uint[256] with length: 4, containing values 0: 0 uint256, 1: 1 uint256, 2: 2 uint256, 3: 3 uint256). The main central area displays the same Solidity code for the ForLoop contract. The status bar at the bottom shows: ContractDefinition ForLoop → 1 reference(s) ▾ ▾, 0 listen on all transactions, Search with transaction hash or address, [vm] from: 0x583...eddC4 to: ForLoop.loop() 0xf8e...9f8e8 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x724...0add1, and a Debug button.

## Decision making

### If statement

#### Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main editor window contains the code for the `IfStatement` contract:

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

Below the editor, the "TRANSACTIONS" section shows two entries:

- [vm] from: 0x583...edd4 to: ForLoop.loop() 0xf8e...9f8e value: 0 wei data: 0xa92...10cb logs: 0 hash: 0x724...0add1 Debug
- [vm] from: 0x583...edd4 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 displays the "DEBUGGER CONFIGURATION" settings, which include a checked checkbox for "Use generated sources (Solidity >= v0.7.2)" and a "Stop debugging" button.

The main editor window shows the same `IfStatement` contract code as before. Below the editor, the "ASSEMBLY" section displays the assembly code corresponding to the current state:

```
0xc29922ad2898edc14f0eece0a3019...
Stop debugging
```

The assembly code includes the following lines:

- 0: 0077 JUMPDEST
- 078 PUSH1 00
- 080 PUSH1 14
- 082 PUSH1 00
- 084 SLOAD
- 085 LT

If...else

statement Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files such as 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, Interfaces.sol, Libraries.sol, Assembly.sol, Events.sol, ErrorHandling.sol, ArithmeticOperator.sol, RelationalOperator.sol, LogicalOperator.sol, DoWhileLoop.sol, ForLoop.sol, IfStatement.sol, and IfElse.sol. The IfElse.sol file is currently selected. The main central area displays the Solidity code for the IfElse contract:

```
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;
    }
}
```

Below the code editor, the status bar indicates "Execution cost: 22744 gas" and "FunctionDefinition decision\_making".

## 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)" and a "Stop debugging" button. The central area displays the same Solidity code as the previous screenshot. Below the code editor, the status bar indicates "ContractDefinition IfElse" and "1 reference(s)". The bottom section shows the Solidity State and Function Stack. The Function Stack shows a single entry: "0: decision\_making()". The Solidity State shows the variable "i: 20 uint256" and "even: true bool". The assembly code at the bottom is:

```
004 RETURN
005 JUMPDEST
006 PUSH1 00
008 DUP1
009 PUSH1 02
071 PUSH1 00
```

## If...else if...else statement

### Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files, including 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, LogicalOperator.sol, DowhileLoop.sol, ForLoop.sol, IfStatement.sol, and IfElse.sol. The file IfElse.sol is currently selected. The main editor area displays the Solidity code for the IfElseIf contract:

```
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 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)" and a deployment address "0xc3983f1150fa2158e0e5b0075b29c...". The central editor area shows the same Solidity code as before. Below the code, the "ContractDefinition IfElseIf" section indicates "1 reference(s)". The bottom pane displays the EVM assembly code for the current state:

```
i: 10 iint256
result: equal to 10 string
178 JUMPDEST
179 PUSH1 60
182 PUSH1 0a
184 PUSH1 00
186 SLOAD
```

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity source files, including `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`, `LogicalOperator.sol`, `Dowhileloop.sol`, `ForLoop.sol`, `IfStatement.sol`, `IfElse.sol`, and `IfElseIf.sol`. The file `String.sol` is currently selected. The main editor area displays the Solidity code for the `StringsExample` contract:

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

## Output:

The screenshot shows the Remix Ethereum IDE with the DEBUGGER configuration open. The configuration panel includes a checkbox for "Use generated sources (Solidity >= v0.7.2)" and a dropdown menu showing the address `0x02b5a78ec919c54377899e8881ef...`. Below this, there are buttons for "Stop debugging" and a step control slider. The main editor area shows the same `StringsExample` contract code as before. The Solidity State pane at the bottom shows the state of the variable `text` with the value `Hello World`. The assembly code pane shows the EVM assembly for the `setText()` function call, including instructions like `JUMPDEST`, `PUSH1 60`, `PUSH1 40`, `MLOAD`, `DUP1`, and `PUSH1 40`.

The screenshot shows the Remix Ethereum IDE interface. In the top left, there's a 'FILE EXPLORER' sidebar listing various Solidity files like Variable.sol, BitwiseOperator.sol, etc. The main central area displays the code for 'Array.sol'. Below the code editor, the 'ContractDefinition Types' section shows a list of transactions:

- 0: |vm| from: 0x583...edd4 to: ArrayExample.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x625...00b09
- call to ArrayExample.testArray
- CALL [call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: ArrayExample.testArray() data: 0x228...3bffd

At the bottom right, there are two 'Debug' buttons. The system tray at the bottom shows the date and time as 3:34 PM, 7/2/2022.

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

## Output:

This screenshot shows the Remix Ethereum IDE with the 'DEBUGGER' section active on the left. It includes a 'DEBUGGER CONFIGURATION' panel with a checked checkbox for 'Use generated sources (Solidity >= v0.7.2)' and a transaction ID '0x735733dec9b850d092d87c6aea4...'. The main code editor area shows the same 'Array.sol' code as before. The bottom section displays the execution environment and output:

- EXECUTION COST:** 244/20 gas used
- input:** 0x738...ed52a
- decoded input:** ()
- decoded output:** { "0": "uint256[]": [10, 20, 30, 40, 50], "1": "int256[]": [-60, 70, -80, 90, -100, -120, 140] }
- logs:** []

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files such as BitwiseOperator.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, LogicalOperator.sol, DowhileLoop.sol, ForLoop.sol, IfStatement.sol, IfElse.sol, IfElseif.sol, String.sol, and Array.sol. Below these, a folder structure for 'scripts', 'tests', and 'README.txt' is visible. The main central area displays the Solidity code for 'Enum.sol'. The code defines a contract named 'Enum' with an enum 'FreshJuiceSize' containing values 'SMALL', 'MEDIUM', and 'LARGE'. It includes functions 'setLarge()', 'getChoice()', and 'getDefaultChoice()'. A tooltip for 'ContractDefinition Enum' indicates there is 1 reference(s). The bottom status bar shows the date and time as 7/24/2022 3:41 PM.

```
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);
    }
}
```

## Output:

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. On the left, the 'Deploy' section is active, showing options to 'Publish to IPFS' or 'At Address'. Below this, a 'Transactions recorded' section shows 20 transactions. Under 'Deployed Contracts', the contract 'ENUM AT 0X332.D4B6D (MEMO)' is listed with its address. To the right, the same Solidity code for 'Enum.sol' is displayed. The bottom status bar shows the date and time as 7/24/2022 3:43 PM.

```
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 screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files such as 1A.sol, AssignmentOperator.sol, ArithmeticOperator.sol, etc. The main editor area displays 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 with the "DEPLOY & RUN TRANSACTIONS" sidebar open. The "Deploy" button is highlighted. The sidebar also includes options to "Publish to IPFS" or "At Address". The main editor area shows the same Solidity code as before, but now it includes a deployment address and interaction buttons:

```
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', 4);
    }
    function getBookId() public view returns (uint) {
        return book.book_id;
    }
}
```

The "Transactions recorded" section shows 24 recorded transactions. The "Deployed Contracts" section shows a deployed contract at address 0x93F...C96CC with a balance of 0: uint256: 4. Interaction buttons for "setBook" and "getBookId" are visible.

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity files: 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, LogicalOperator.sol, Dowhileloop.sol, ForLoop.sol, IfStatement.sol, IfElse.sol, IfElseif.sol, String.sol, Array.sol, Enum.sol, Struct.sol, and Mapping.sol. The Mapping.sol file is currently selected. The main editor area displays the Solidity code for the Mapping contract:

```
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;
    }
}
```

## Output:

The screenshot shows the Remix Ethereum IDE interface with the DEBUGGER tab selected. The left sidebar shows the Function Stack, Solidity Locals, and Solidity State. The Solidity State section displays the state of the mapping\_example contract, showing a mapping from address to student. A transaction has been sent to the adding\_values() function, resulting in a new entry in the student\_result array. The transaction logs show the creation of the student object and its addition to the array. The bottom part of the interface shows the assembly code for the current state:

```
1 JUMPDEST
2 PUSH1 01
3 DUP2
4 DUP2
5 SLOAD
6 DUP2
```

```

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:

Deployed Contracts

- SPECIALVARIABLE AT 0X540...C75

| Function           | Value         |
|--------------------|---------------|
| setRollNO          | 7             |
| whatIsMyRollNumber | 0: uint256: 7 |

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

### Function:

#### **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;
    }
}

```

## Output:

Transactions recorded: 18

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.

getResult

0: uint256: product 220  
1: uint256: sum 31

Low level interactions  
CALLDATA  
Transact

## Functions Modifiers

### 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:

```
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;
    }
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'FILE EXPLORERS' panel displays a workspace named 'remixDefault\_1655651/34405'. Inside this workspace, there are several contracts and scripts listed under the 'contracts' folder, including 1\_Storage.sol, 2\_Owner.sol, 3\_Ballot.sol, 1A.sol, 1B.sol, white\_loop.sol, while\_loop1.sol, wl.sol, Function.sol, Functions.Modifier.sol, View.Function.sol (which is currently selected), PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, and CryptographicFunction.sol. Below the contracts are 'scripts' and 'tests' folders, along with a README.txt file. The main central area is a code editor showing the following Solidity code:

```
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 with the 'DEPLOY & RUN TRANSACTIONS' panel active. This panel includes fields for deploying contracts at an address or loading one from an address. It also displays a list of 'Transactions recorded' with 21 items, noting that all transactions (deployed contracts and function executions) can be saved and replayed in another environment. Below this is a 'Deployed Contracts' section showing a single entry: 'VIEWFUNCTION AT 0xE23L4157A ()'. Under this entry, the 'getResult' function is listed with its parameters: '0: uint256 product160' and '1: uint256 sum26'. At the bottom of the panel, there are sections for 'Low level interactions' and 'CALLDATA', with a 'Transaction' button.

The screenshot shows the Remix Ethereum IDE interface. In the top navigation bar, there are tabs for "Remix - Ethereum IDE" and "Computers - Google Drive". Below the tabs, a URL bar displays "Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". The main workspace contains several tabs: "Funcion.sol", "Functions Modifier.sol", "View Function.sol", "PureFunction.sol" (which is currently active), "FallbackFunction.sol", and "FunctionOverload".

The left sidebar is titled "FILE EXPLORERS" and shows a file tree for a workspace named "remixDefault\_1655651734405". The "contracts" folder contains multiple Solidity files like "Storage.sol", "Owner.sol", "Ballot.sol", "A.sol", "B.sol", "whileloop1.sol", "wl.sol", "Fucion.sol", "Functions Modifier.sol", "View Function.sol", "PureFunction.sol" (selected), "FallbackFunction.sol", "FunctionOverloading.sol", "MathematicalFunction.sol", and "CryptographicFunction.sol". There are also "artifacts", "scripts", and "tests" folders, along with a "README.txt" file.

The central code editor area displays the following Solidity code:

```
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;
    }
}
```

## Output:

The screenshot shows the same Remix Ethereum IDE interface, but the left sidebar is now titled "DEPLOY & RUN TRANSACTIONS".

The "Transactions recorded" section indicates 22 transactions. It provides instructions for saving and replaying transactions. The "Deployed Contracts" section shows a deployed contract named "PUREFUNCTION AT 0X1C9...2B4BD ()". Under this contract, the "getResult" function is listed with its outputs: "0: uint256: product 20" and "1: uint256: sum 12".

The code editor area remains the same, displaying the Solidity code for the "PureFunction" contract.

The bottom status bar shows a transaction log: "[vm] from: 0x583...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xe0e9...5a4d6". A "Debug" button is visible at the bottom right.

```
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:

```
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));
}
```

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays the 'FILE EXPLORERS' panel with a workspace named 'remixDefault\_1655651734405'. Inside this workspace, there are several contracts and scripts, including '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', and 'CryptographicFunction.sol'. The right side of the interface features a code editor with the file '1.sol' open, containing 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);
    }
}
```

## Output:

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' panel selected. This panel includes a message about deploying contracts and a list of deployed contracts under 'FUNCTIONOVERLOADING AT 0X5A8'. Below this, there are two buttons labeled 'callSumWithT...' and 'getSum'. The right side of the interface features a code editor with the file '1.sol' open, displaying the same Solidity code as the previous screenshot.

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar includes tabs for Remix - Ethereum IDE, Computers - Google Drive, and a link to a solidity file. Below the tabs, there are browser-like controls for back, forward, search, and refresh.

The main area is divided into several panes:

- FILE EXPLORERS** pane on the left, titled "remixDefault\_1655651734405". It lists various contracts and scripts under the "contracts" folder, including 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 a README.txt file.
- Code Editor** pane on the right containing the Solidity code for the `MathematicalFunction` contract:

```
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);
    }
}
```

At the bottom of the interface are buttons for "listen on all transactions", a search bar, and a transaction history section.

## Output:

This screenshot shows the same Remix Ethereum IDE interface, but with a different active panel: **DEPLOY & RUN TRANSACTIONS**.

The left sidebar displays a list of recorded transactions and a note about replaying them in another environment. The "Deployed Contracts" section shows a single contract named `MATHEMATICALFUNCTION` at address `0x40f...`. Under this contract, two functions are listed: `callAddMod` and `callMulMod`, each with its corresponding output value.

The right side of the interface remains the same as the first screenshot, showing the code editor with the `MathematicalFunction.sol` code.

```
pragma solidity ^0.5.0;

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

## Output:

```
pragma solidity ^0.5.0;

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

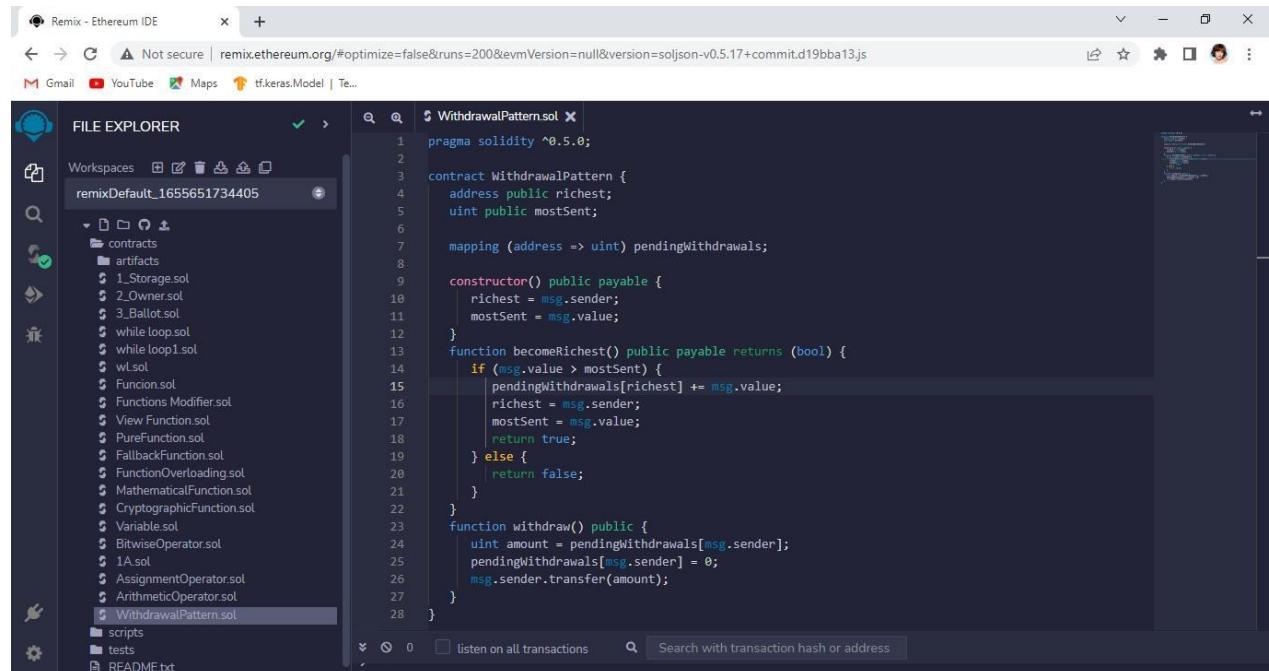
# Practical 3

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

## 3a) Withdrawal Pattern, Restricted Access.

### Withdrawal Pattern

Code:



```
pragma solidity ^0.5.0;

contract WithdrawalPattern {
    address public richest;
    uint public mostSent;

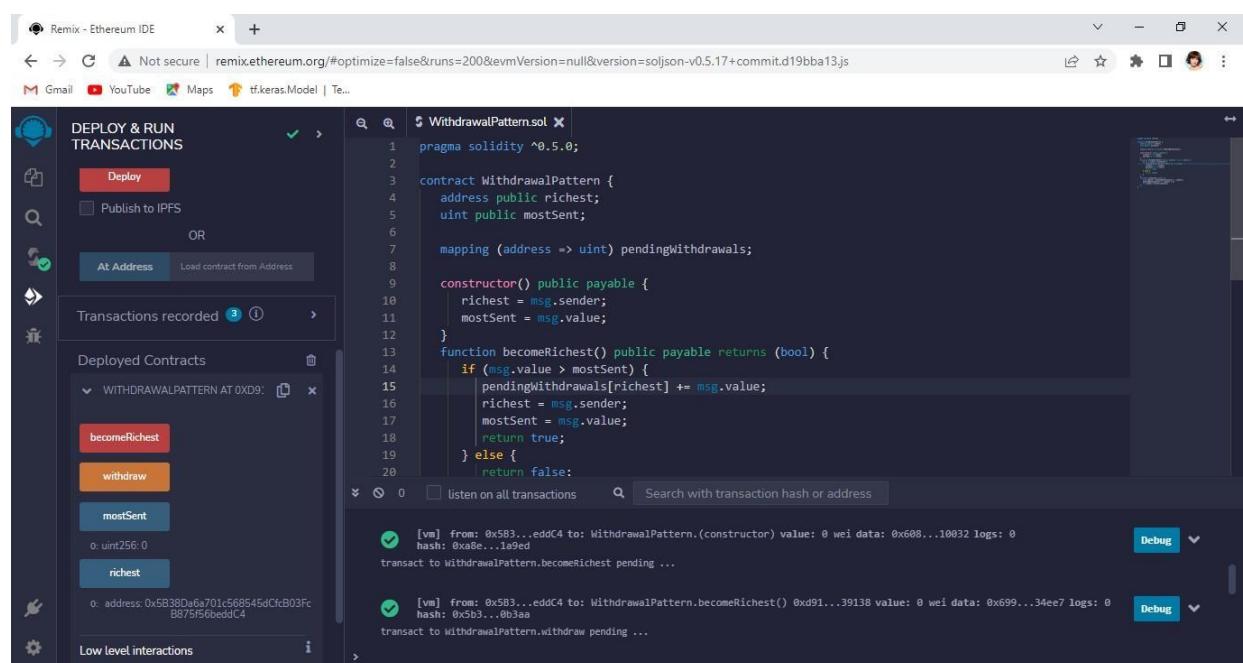
    mapping (address => uint) pendingWithdrawals;

    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);
    }
}
```

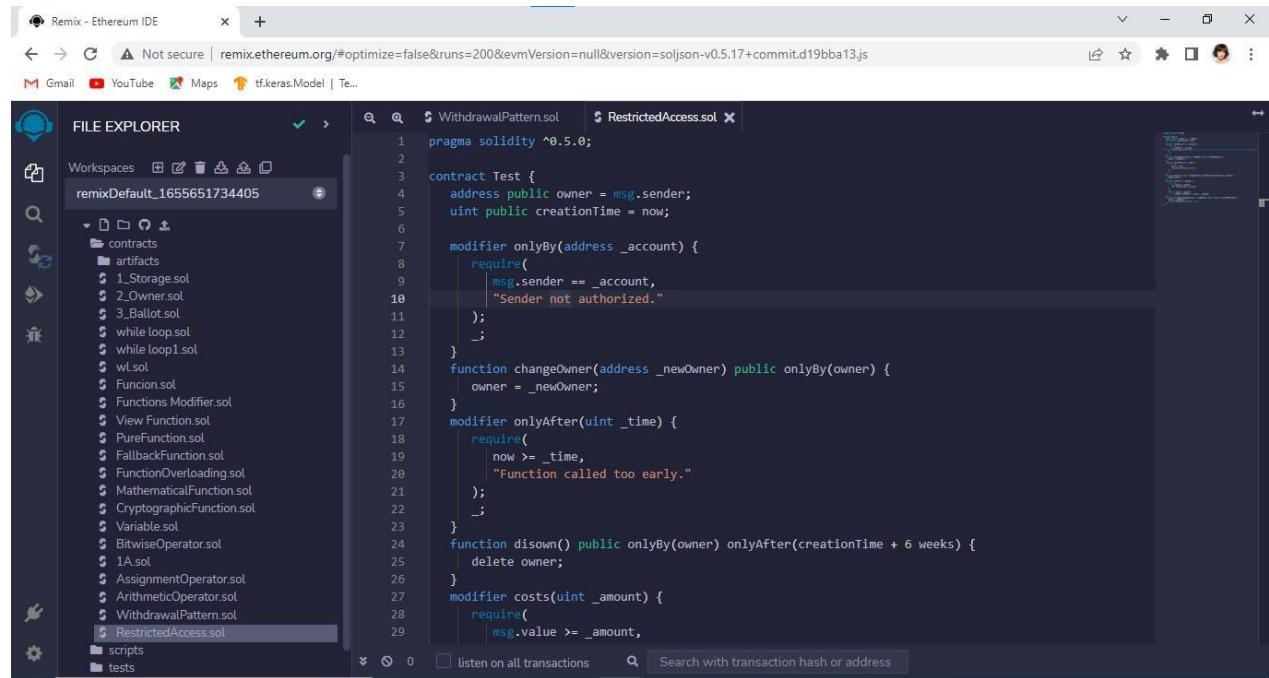
Output:



The screenshot shows the Remix Ethereum IDE interface with the "WithdrawalPattern.sol" file loaded. The left sidebar has a "DEPLOY & RUN TRANSACTIONS" section with a "Deploy" button and a "Transactions recorded" list. The list shows two transactions: one for deploying the contract and another for calling the "becomeRichest" function. The right side shows the Solidity code for the "WithdrawalPattern" contract, which includes a constructor, a "becomeRichest" function that updates the "richest" and "mostSent" variables, and a "withdraw" function that sends the pending withdrawal amount back to the sender.

## Restricted Access

**Code:**



The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'remixDefault\_1655651734405' workspace, with 'contracts' expanded to show several .sol files. The right pane contains the Solidity code for the 'RestrictedAccess.sol' contract. The code includes pragmas, imports, and a main contract definition with modifiers like 'onlyBy' and 'onlyAfter'.

```
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."
        );
    }

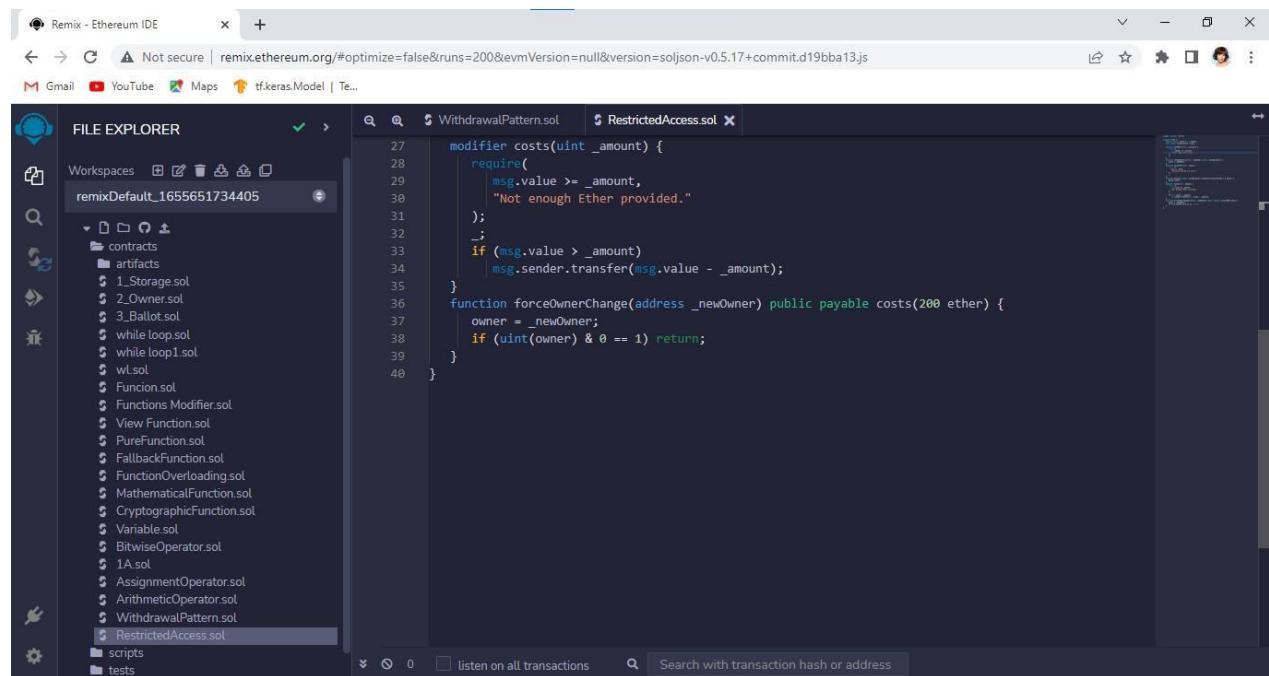
    function changeOwner(address _newOwner) public onlyBy(owner) {
        owner = _newOwner;
    }

    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,
            "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;
}
```



The screenshot shows the continuation of the Remix Ethereum IDE interface. The left sidebar remains the same. The right pane continues the Solidity code for the 'RestrictedAccess.sol' contract, focusing on the 'costs' modifier and the 'forceOwnerChange' function.

```
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. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar displays a deployed contract named 'TEST AT 0XD7A...F71B (MEMORY)'. It lists several functions: 'changeOwner' (orange button), 'disown' (orange button), 'forceOwnerCh...' (red button), 'creationTime' (blue button), 'owner' (blue button), and 'o: uint256: 1658594164' (blue button). The main workspace shows the Solidity code for the 'RestrictedAccess.sol' contract. The code includes a modifier 'costs' that checks if the provided value is greater than or equal to the required amount, and a function 'forceOwnerChange' that changes the owner if the caller has the correct permissions.

```
25 |     delete owner;
26 |
27 |     modifier costs(uint _amount) {
28 |         require(
29 |             msg.value >= _amount,
30 |             "Not enough Ether provided."
31 |         );
32 |
33 |         if (msg.value > _amount)
34 |             msg.sender.transfer(msg.value - _amount);
35 |
36 |     function forceOwnerChange(address _newOwner) public payable costs(200 ether) {
37 |         owner = _newOwner;
38 |         if (uint(owner) & 0 == 1) return;
39 |     }
40 | }
```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options like "Publish to IPFS" and "At Address". The main area displays two tabs: "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, the "Transactions recorded" section shows two recent transactions:

- A successful transaction to "TEST AT 0xC0D6...99DF9 (MEMORY)" with the "changeOwner" function, hash: 0xD6a42782d230D7c13A74ddc5d, value: 0 wei.
- A pending transaction to "WithdrawalPattern.becomeRichest" with the "becomeRichest" function, hash: 0x5b3...003aa, value: 0 wei.

At the bottom right, there are "Debug" buttons for each transaction.

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

## Contracts

## Code:

The screenshot shows the Remix - 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-v0.5.1+commit.d19bba13.js". The left sidebar is titled "FILE EXPLORER" and shows a workspace named "remixDefault\_1655651734405". Inside this workspace, there are several contracts and artifacts, including "1\_Storage.sol", "2\_Owner.sol", "3\_Ballot.sol", "while loop.sol", "while loop1.sol", "wl.sol", "Function.sol", "Functions Modifiers.sol", "View Function.sol", "PureFunction.sol", "FallbackFunction.sol", "FunctionOverloading.sol", "MathematicalFunction.sol", "CryptographicFunction.sol", "Variable.sol", "BitwiseOperator.sol", "IA.sol", "AssignmentOperator.sol", "ArithmeticOperator.sol", "WthdrawalPattern.sol", "RestrictedAccess.sol", and "Contracts.sol". The main editor area contains the Solidity code for "WthdrawalPattern.sol".

```
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. The top bar displays the title "Remix - Ethereum IDE" and the URL "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.1+commit.d19bba13.js". The left sidebar is titled "FILE EXPLORER" and shows a workspace named "remixDefault\_1655651734405". Inside this workspace, there are several contracts and artifacts, including "1\_Storage.sol", "2\_Owner.sol", "3\_Ballot.sol", "while loop.sol", "while loop1.sol", "wl.sol", "Function.sol", "Functions Modifiers.sol", "View Function.sol", "PureFunction.sol", "FallbackFunction.sol", "FunctionOverloading.sol", "MathematicalFunction.sol", "CryptographicFunction.sol", "Variable.sol", "BitwiseOperator.sol", "IA.sol", "AssignmentOperator.sol", "ArithmeticOperator.sol", "WthdrawalPattern.sol", "RestrictedAccess.sol", and "Contracts.sol". The main editor area contains the Solidity code for "Contracts.sol".

```
//Derived Contract
contract E is C {
    uint private result;
    C private c;

    constructor() public {
        c = new C();
    }
    function getComputedResult() public {
        result = compute(3, 5);
    }
    function getResult() public view returns(uint) { return result; }
    function getData() public view returns(uint) { return c.info(); }
}
```

**Output:**

```

//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(); }
}

```

```

//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. The top bar displays the title "Remix - Ethereum IDE" and a warning about the connection being "Not secure". Below the title is a navigation bar with links to "Gmail", "YouTube", "Maps", and "tf.keras.Model". The main area is divided into several panes: a "FILE EXPLORER" on the left listing various Solidity files, a central "CODE EDITOR" pane containing the WithdrawalPattern.sol code, and a "CONTRACTS" pane on the right showing the deployed contract details.

```
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 getComputedResult() public {
        result = compute(3, 5);
    }

    function getResult() public view returns(uint) { return result; }
    function getData() public view returns(uint) { return c.info(); }
}
```

This screenshot shows the same Remix IDE interface as the first one, but with changes made to the WithdrawalPattern.sol code. Specifically, the constructor for the derived contract E now initializes the variable 'c' instead of 'result'. The rest of the code remains the same.

```
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 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. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is open, showing deployment options like 'Deploy' and 'At Address'. Below it, the 'Transactions recorded' section lists 24 entries for the deployed contract C. The main panel displays the Solidity code for the WithdrawalPattern.sol contract. The code defines a contract C with a private state variable 'data' and a public state variable 'info'. It includes a constructor setting 'info' to 20, a private function 'increment' returning the value plus one, a public function 'updateData' setting 'data' to the input 'a', and a view function 'getData' returning 'data'. A derived contract E is also defined, inheriting from C and adding a private state variable 'result'.

```
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();
    }
}
```

This screenshot shows the same Remix IDE session after a transaction has been executed. The 'Transactions recorded' section now shows 25 entries. In the 'updateData' section of the sidebar, there is a new entry with the value '17' and a 'transact' button. The Solidity code remains identical to the first screenshot, but the deployment address for contract C is now shown as 0x5E1\_4EFF5 (MEMORY).

```
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();
    }
}
```

## Constructors

**Code:**

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files and artifacts. In the center, a code editor displays the following Solidity 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 with the DEPLOY & RUN TRANSACTIONS tab selected. The CONTRACT section shows the deployed contract "constructorExample". The Transactions recorded section shows a recent transaction for "getValue". The Deployed Contracts section shows the deployed contract at address 0x1C. The Low level interactions section shows a call to "getValue" with the result "0: string: This is a example of Constructor".

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files and scripts. In the center, a code editor window displays 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;
    }
}
```

At the bottom of the code editor, transaction logs are visible:

- [call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: Test.getResult() data: 0xde2...92789

## Output:

The screenshot shows the Remix Ethereum IDE interface with the DEPLOY & RUN TRANSACTIONS tab selected. On the left, the sidebar shows the deployed contract TEST AT 0xD91\_39138 (MEMORY). In the center, the code editor window shows the same Solidity code as before. Transaction logs at the bottom indicate a successful deployment and a call to the getResult() function:

- [vm] from: 0x583...eddC4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x2a3...4b9ff call to Test.getResult
- [call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: Test.getResult() data: 0xde2...92789

The screenshot shows the Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several contracts and scripts. In the center, a code editor window displays 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;
    }
}
```

At the bottom of the code editor, the output pane shows a transaction log:

```
[vm] from: 0x5B3...eddC4 to: Test.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x137...54395
call to Test.getResult
```

## Output:

The screenshot shows the Ethereum IDE interface with the DEPLOY & RUN TRANSACTIONS sidebar open. Under the CONTRACT section, the contract "Test - contracts/Interfaces.sol" is selected, and the "Deploy" button is highlighted. The code editor window shows the same Solidity code as before.

The output pane at the bottom shows two transaction logs:

```
[vm] from: 0x5B3...eddC4 to: Test.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x137...54395
call to Test.getResult
```

```
[call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: Test.getResult() data: 0xde2...92789
call to Test.getResult
```

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

### Libraries

#### Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity files: 2\_Owner.sol, 3\_Ballot.sol, while\_loop.sol, while\_loop1.sol, wlSol, 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, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, and Libraries.sol. The Libraries.sol file is currently selected and open in the main code 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 or -1 if it's not found. 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 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 selected. It displays the deployed contract Libraries at address 0x35B...D5EE3. The contract has a single function, isValuePresent, which returns the value 4. The interface also shows the deployed contract's low-level interactions and a transaction history.

Deployed Contracts

- LIBRARIES AT 0x35B...D5EE3 (MEM)
  - isValuePresent
    - o: uint256: 3

Low level interactions

CALLDATA

Transact

## Assembly

### Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity files: 3\_Ballot.sol, while\_loop.sol, while\_loop1.sol, wLsol, 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. The main editor area displays the Solidity code for the `Assembly` contract:

```
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` sidebar open. The CONTRACT dropdown is set to `Assembly - contracts/Assembly.sol`. The Deploy button is highlighted. Below it, there are options to `Publish to IPFS` or `At Address` (which is selected). The Transactions recorded section shows one entry: `ASSEMBLY AT 0xD91_39138 (MEN)`. The Low level interactions section shows a call to the `add` function with parameters `a: uint256: b 45`. The main editor area shows the same Solidity code as the previous screenshot. The status bar at the bottom indicates a successful deployment and a failed call to the `add` function due to an error in encoding arguments.

The screenshot shows the Remix Ethereum IDE interface. In the top left, there's a browser bar with the URL [Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.4.26+commit.4563c3fc.js&language=Solidity](https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.4.26+commit.4563c3fc.js&language=Solidity). The main area is titled "FILE EXPLORER" and lists several Solidity files. One file, "Events.sol", is selected and shown in the code editor. The code defines a contract "Events" with a state variable "value" and a function "getValue" that emits an event "Increment". Below the code editor, the "ContractDefinition" section shows "eventExample" with 1 reference(s). The bottom right shows two transaction logs: one from the VM and one from a call to the contract.

```
// 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;
    }
}
```

## Output:

This screenshot shows the "DEPLOY & RUN TRANSACTIONS" interface in the Remix Ethereum IDE. On the left, there are options to "Deploy" or "Publish to IPFS" or "At Address". The "At Address" option is selected. The "Transactions recorded" section shows a recent transaction for the contract "EVENTS AT 0x583...9FB8E (MEMO)". Below it, the "Deployed Contracts" section shows the "Events" contract with its address. A "getValue" function call is being interacted with, with inputs "a: 500" and "b: 300" and a "value" output of "0: uint256: 800". The right side of the screen is identical to the previous screenshot, showing the Solidity code and transaction logs.

```
// 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;
    }
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity files, with `ErrorHandling.sol` currently selected. The main workspace displays the Solidity code for `ErrorHandling`. The code includes two functions: `checkInput` and `Odd`. The `checkInput` function requires that the input is between 0 and 255. The `Odd` function returns true if the input is odd. At the bottom, the transaction history shows the deployment of the contract and a call to its constructor.

```
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 Remix Ethereum IDE interface with the `DEPLOY & RUN TRANSACTIONS` sidebar open. It displays the deployed contract `ERRORHANDLING AT 0x907...B5E`. The `checkInput` function is called with the value `243`, and the result is `0: string: Input is Uint8`. The `Odd` function is also called with the same value, and the result is `0: bool: true`.

```
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;
    }
}
```

## Practical 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$ cryptogen 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:
      ca01:
        image: hyperledger/fabric-ca:$IMAGE_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/ca.org1.example.com-cert.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 -e org1.example.com -p 7054 -t'

```

**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:
      ca01:
        image: hyperledger/fabric-ca:$IMAGE_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/ca.org2.example.com-cert.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 -e org2.example.com -p 7054 -t'

```

### 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
33

```

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      - peer0.org1.example.com:/var/hyperledger/production
51    ports:
52      - 7051:7051
53      - 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-clt.yaml -f docker-compose-couch.yaml up -d 2>&1
Creating orderer.example.com ...
Creating couchdb1 ...
Creating couchdb3 ...
Creating couchdb2 ...
Creating orderer.example.com
Creating couchdb0 ...
Creating couchdb2
Creating couchdb1
Creating couchdb3
Creating couchdb0 ... done
Creating 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 ...
Creating peer0.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<br>PORTS | IMAGE  | NAMES                                     | COMMAND  | CREATED      | STATUS      |
|-----------------------|--|---|--|--------------|-------------|
| badfe5ea8a92          | hyperledger/fabric-peer:latest   | peer node start<br>peer1.org1.example.com | "peer node start"  | 4 hours ago  | Up 3 hours  |
| 50d9fa8f5dfb          | hyperledger/fabric-peer:latest   | peer node start<br>peer0.org2.example.com | "peer node start"  | 4 hours ago  | Up 3 hours  |
| ee4b501b7d00          | hyperledger/fabric-peer:latest   | peer node start<br>peer0.org1.example.com | "peer node start"  | 4 hours ago  | Up 3 hours  |
| 32f0f4426a22          | hyperledger/fabric-peer:latest   | peer node start<br>peer1.org2.example.com | "peer node start"  | 4 hours ago  | Up 3 hours  |
| ec8f9df17258          | hyperledger/fabric-couchdb<br>4369/tcp, 9100/tcp, 0.0.0.0:8984->5984/tcp | couchdb3                                  | "tini -- /docker-entrypoint-initdb.d/couchdb.sh & couchdb -p 5984" | 4 hours ago  | Up 4 hours  |
| 2b4be64efcf6          | hyperledger/fabric-couchdb<br>4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp | couchdb2                                  | "tini -- /docker-entrypoint-initdb.d/couchdb.sh & couchdb -p 5984" | 4 hours ago  | Up 4 hours  |
| f922625a027e          | hyperledger/fabric-couchdb<br>4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp | couchdb0                                  | "tini -- /docker-entrypoint-initdb.d/couchdb.sh & couchdb -p 5984" | 4 hours ago  | Up 4 hours  |
| 8de729e174b0          | hyperledger/fabric-couchdb<br>4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp | couchdb1                                  | "tini -- /docker-entrypoint-initdb.d/couchdb.sh & couchdb -p 5984" | 4 hours ago  | Up 4 hours  |
| 18830d7ccf5e          | hyperledger/fabric-tools:latest  | cli                                       | "/bin/bash"  | 11 hours ago | Up 11 hours |
| c7ef15868cd5          | hyperledger/fabric-orderer:latest<br>0.0.0.0:7050->7050/tcp              | orderer                                   | "orderer"  | 11 hours ago | Up 11 hours |
| orderer.example.com   |  |   |  |              |             |

Proposed network setup is complete, our network have -

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

#### 4. Creating channel

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

#### 5. Joining first peer where channel is created —

```
docker exec -e

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

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

orderer For Fetching –

```
docker exec -e

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

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

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

For joining channel –

```
docker exec -e

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

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

### Install chainCode on every machine

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

PeerAdmin@multi_org1
```

### Start chaincode on one machine

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

PeerAdmin@multi_org1
```

### Conclusion

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

## Practical 5

**Aim:** Demonstrate the running of the blockchain node.

**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(se
            lf._private_key)

    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER'))
            .decode('ascii')
Dinesh = Client()
print (Dinesh.identity)
```

**Output:**

```
D:\ 30819f300d06092a864886f70d0101050003818d0030818902818100eea39d3e40f737cff2050d40515a4833c80cdd9073d88a3629aef45ee77793c78197eb0f8bbf0688c0672e22ee74e691bd53668
```

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

**Code:**

```
class Transaction:  
    def __init__(self,  
                 sender,  
                 recipient,  
                 value):  
        self.sender =  
        sender  
        s  
        e  
        l  
        f  
        .  
        r  
        e  
        c  
        i  
        p  
        i  
        e  
        n  
        t  
        =  
        r  
        e  
        c  
        i  
        p  
        i  
        e  
        n  
        t  
        s  
        e  
        l  
        f  
        .  
        v  
        a  
        l  
        u  
        e  
        =  
        v  
        a  
        l  
        u  
        e  
        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:**



```
0f18c12abcf2904299162ce5c39689df19a9bf583689ec58182cac42617da5b1282d467b6a92ffd358f33052abbeefd502d1bbfcfcde293962e8ecec7b9f6a249df1bfcacf537a19404913e73079fc58c1f
```

**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: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a
-----
value: 15.0
-----
time: 2022-07-30 17:35:06.809777
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a72cfb527dff9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2b
-----
value: 6.0
-----
time: 2022-07-30 17:35:06.811866
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a81
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d57
-----
value: 2.0
-----
time: 2022-07-30 17:35:06.814247
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a81
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d57
-----
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.transactions = []
```

```
c          o
t          c
i          k
o          -
n          h
s          a
=
[          s
]          h
s          =
e          "
e          "
l          s
f          e
.
p          l
r          f
e          .
v          N
i          o
o          n
u          c
s          e
-
b          =
l          "
          "
last_block_hash = ""

Dinesh = Client()

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

block0 = Block()

block0.previous_block_hash = None
Nonce = None

block0.verified_transactions.append (t0)

digest = hash (block0)
last_block_hash = digest

TPCoins = []

def dump_blockchain (self):
    print ("Number of blocks in the chain: " + str(len (self)))
```

```

for x in range (len(TPCoins)):
    block_temp = TPCoins[x]
    print ("block # " + str(x))
    for transaction in block_temp.verified_transactions:
        display_transaction (transaction)
    print ('- -----')
    print ('=====') 

TPCoins.append (block0)
dump_blockchain(TPCoins)

```

**Output:**

---

```

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



**1e) Create a mining function and test it.**

**Code:**

```

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

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

mine ("test message", 2)

```

**Output:**

```
after 238 iterations found nonce: 1124711db6185591392c6a06c24a3c2ebbaeca647fb8374824f939ada27c09e9
after 353 iterations found nonce: 11e0a4b57bb76496ecc6ab5a3c126165bc9dbbaf80f664e7e192a422360c3884
after 419 iterations found nonce: 11280dfd9ab05b3dbfa869990153732941408faa4a3b0832819b161f52321c08
after 511 iterations found nonce: 1150944bd7ea429acd052da390b148f27eb881686e0f8bfcdf77f833af878e2e
after 822 iterations found nonce: 110e61d41d94b48f6dfcb6f43f9ceafe2ab7168456dab088e05126d43d0366ef
after 924 iterations found nonce: 11147bf32bafeee4505b5cc40c3b227366d8a41ab3bd740e437c74400b7a8127
'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: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa

```

## **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 left sidebar displays a file explorer with various Solidity files and scripts. The main editor window shows the `Variable.sol` contract:

```
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:**

## Operators:

### Arithmetic Operator

**Code:**

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with various Solidity files and scripts. The main editor window shows the `ArithmeticOperator.sol` contract:

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

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

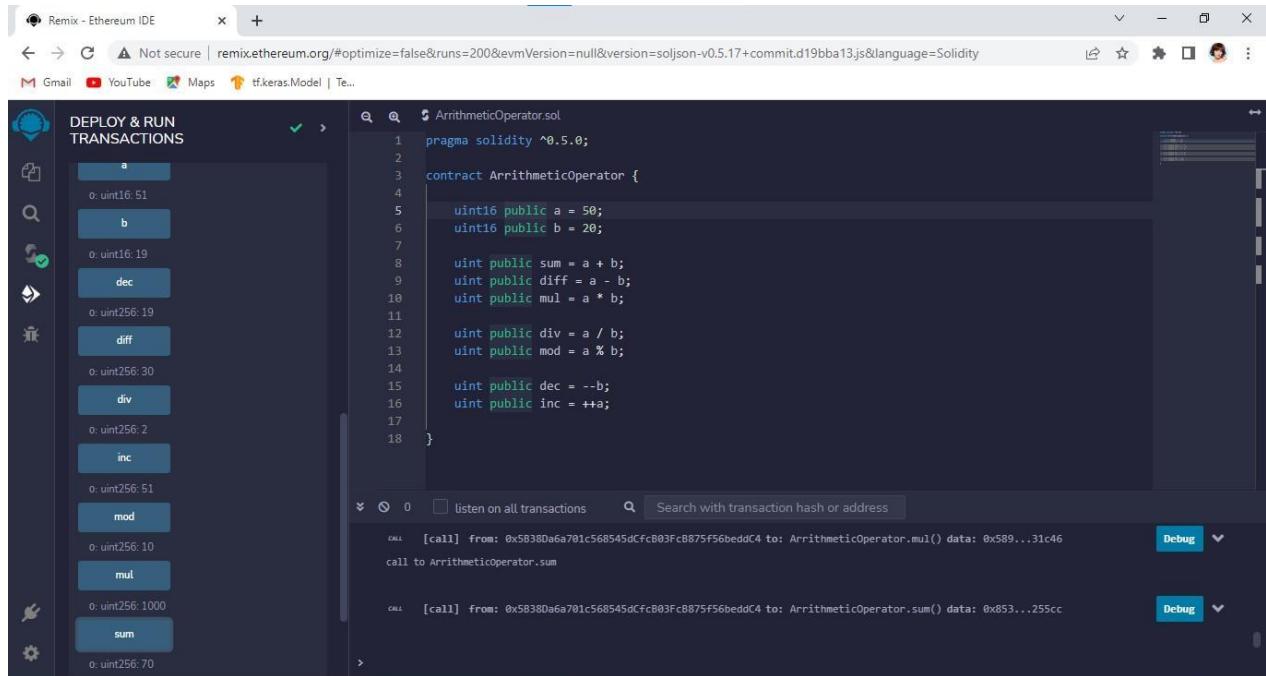
    uint public div = a / b;
    uint public mod = a % b;

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

At the bottom, two deployment logs are shown:

- [vm] from: 0x5B3...eddC4 to: SolidityTest.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x95b...f24a5 creation of Arithmeticoperator pending...
- [vm] from: 0x5B3...eddC4 to: ArithmeticOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x6fa...e48ac

## Output:



The screenshot shows the Ethereum IDE interface with the title "Remix - Ethereum IDE". The left sidebar has a "DEPLOY & RUN TRANSACTIONS" section containing buttons for various arithmetic operations: add, sub, mul, div, mod, inc, dec, and sum. The main code editor displays the Solidity code for the `ArithmeticOperator` contract:

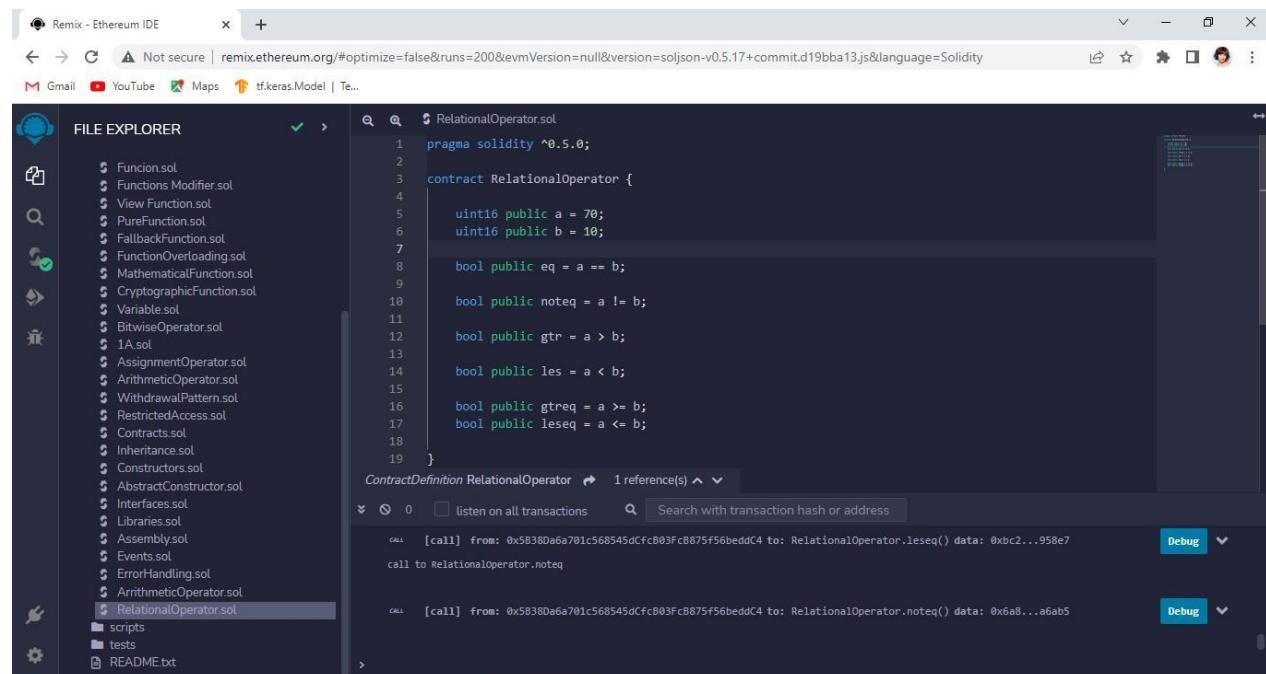
```
pragma solidity ^0.5.0;
contract ArithmeticOperator {
    uint16 public a = 50;
    uint16 public b = 20;
    uint public sum = a + b;
    uint public diff = a - b;
    uint public mul = a * b;
    uint public div = a / b;
    uint public mod = a % b;
    uint public dec = --b;
    uint public inc = ++a;
}
```

The bottom pane shows transaction logs:

- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ArithmeticOperator.mul() data: 0x589...31c46 call to ArithmeticOperator.mul
- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ArithmeticOperator.sum() data: 0x853...255cc call to ArithmeticOperator.sum

## Relational

### Operator Code:



The screenshot shows the Ethereum IDE interface with the title "Remix - Ethereum IDE". The left sidebar has a "FILE EXPLORER" section listing various Solidity files. The main code editor displays the Solidity code for the `RelationalOperator` contract:

```
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 bottom pane shows transaction logs:

- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.leseq() data: 0xbc2...958e7 call to RelationalOperator.leseq
- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.noteq() data: 0xa8...a6ab5 call to RelationalOperator.noteq

## Output:

```

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;
}

```

## Logical Operator

### Code:

```

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. On the left, there's a sidebar with options like 'DEPLOY & RUN TRANSACTIONS' and 'Deploy'. Below that is a section for 'Transactions recorded' and 'Deployed Contracts'. A specific contract named 'LOGICALOPERATOR AT 0XB27...' is selected. In the main area, the code for 'LogicalOperator.sol' is displayed:

```

1 pragma solidity ^0.5.0;
2
3 // Creating a contract
4 contract LogicalOperator{
5
6     function Logic(
7         bool a, bool b) public view returns(
8             bool, bool, bool){
9
10        // Logical AND operator
11        bool and = a&&b;
12
13        // Logical OR operator
14        bool or = a||b;
15
16        // Logical NOT operator
17        bool not = !a;
18
19    }
20

```

Below the code, there's a 'Logic' interaction panel with inputs 'a: true' and 'b: false', and a 'call' button. The results show: 0: bool: false, 1: bool: true, 2: bool: false. At the bottom, there's a transaction log entry:

CALL [call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: LogicalOperator.Logic(bool,bool) data: 0xc94...00000

## Bitwise Operator

**Code:**

The screenshot shows the Remix Ethereum IDE interface with two open files: 'LogicalOperator.sol' and 'BitwiseOperator.sol'. The 'FILE EXPLORER' sidebar on the left lists various contracts and artifacts. The 'BitwiseOperator.sol' file contains the following code:

```

1 pragma solidity ^0.5.0;
2
3 contract BitwiseOperator {
4
5     uint16 public a = 20;
6     uint16 public b = 50;
7
8     uint16 public and = a & b;
9
10    uint16 public or = a | b;
11
12    uint16 public xor = a ^ b;
13
14    uint16 public leftshift = a << b;
15
16    uint16 public rightshift = a >> b;
17
18    uint16 public not = ~a;
19
20 }
21
22
23

```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" which lists various logical operators like and, or, xor, etc., each with their respective EVM opcodes. The main code editor contains the BitwiseOperator.sol contract:

```
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;
}
```

The status bar at the bottom indicates a successful deployment with the message "[vm] from: 0x583...eddC4 to: LogicalOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x5df...62867".

## Assignment Operator

Code:

The screenshot shows the Remix Ethereum IDE interface with the AssignmentOperator.sol contract selected in the file explorer. The code defines a contract with several assignment operations:

```
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 = 50;
    uint16 public assignment_mod = 32;

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

The status bar at the bottom indicates a successful deployment with the message "[vm] from: 0x583...eddC4 to: AssignmentOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xdal...5c92c".

```

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 = 50;
    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:

```

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;
    }
}

```

```

1 pragma solidity ^0.5.0;
2
3 contract Types {
4
5     uint[] data;
6     uint8 j = 0;
7
8     function loop(
9         ) public returns(uint[] memory){
10        while(j < 5) {
11            j++;
12            data.push(j);
13        }
14        return data;
15    }
16
17 }
18

```

## Dowhile loop

### Code:

```

1 pragma solidity ^0.5.0;
2
3 contract Dowhileloop {
4
5     uint[] data;
6     uint8 j = 0;
7
8     function loop(
9         ) public returns(uint[] memory){
10        do{
11            j++;
12            data.push(j);
13        }while(j < 8);
14        return data;
15    }
16
17 }
18

```

### Output:

```
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;
    }
}
```

## For loop

Code:

```
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 screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a 'DEBUGGER' panel with buttons for 'Stop debugging' and navigation controls, along with sections for 'Function Stack' and 'Solidity Locals'. The main code editor displays the Solidity code for a 'ForLoop' contract:

```

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 bottom status bar shows transaction logs and a 'Debug' button.

## Decision making

### If statement

#### Code:

The screenshot shows the Remix Ethereum IDE interface with a 'FILE EXPLORER' sidebar listing various Solidity files. The main code editor displays the Solidity code for an 'IfStatement' contract:

```

pragma solidity ^0.5.0;

contract IfStatement {
    uint i = 20;

    function decision_making()
        public returns(bool)
    {
        if(i<20){
            return true;
        }
    }
}

```

The bottom status bar shows transaction logs and a 'Debug' button.

```

pragma solidity ^0.5.0;
contract IfStatement {
    uint i = 20;
    function decision_making() public returns(bool){
        if(i>0){
            return true;
        }
    }
}

```

## If...else

### statement 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(i2 == 0){
            even = true;
        } else{
            even = false;
        }
        return even;
    }
}

```

### Output:

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEBUGGER' panel is open, showing a function stack with 'decision\_making()' at the top. The main code editor contains 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 EVM stack trace on the right shows the execution flow with assembly instructions like JUMPDEST, PUSH1, and DUP1.

## If...else if...else statement

Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' panel open on the left, displaying a list of Solidity files. The current file being edited is 'IfElseIf.sol'.

```
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;
    }
}
```

The code implements an if...else if...else conditional statement to determine the result based on the value of 'i'.

The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and a URL "Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity". The sidebar on the left contains icons for file operations, search, and settings. The main area is divided into several panes: a "DEBUGGER" pane on the far left with sections for "DEBUGGER CONFIGURATION" (checkbox for "Use generated sources (Solidity >= v0.7.2)" checked), "Function Stack" (showing "o. decision\_making()"), "Solidity Locals" (empty), and "Solidity State" (showing "i:10 uint256" and "result: equal to 10 string"). The central pane displays the Solidity code for the "IfElseIf" contract:

```
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;
    }
}
```

Below the code, a "ContractDefinition IfElseIf" section shows "1 reference(s)". The bottom pane shows the assembly code:

```
0x00: RETURN
179 JUMPIST
180 PUSH1 60
182 PUSH1 08
184 PUSH1 00
196 SLOAD
```

## String

### Code:

The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and a URL "Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity". The sidebar on the left contains icons for file operations, search, and settings. The main area is divided into several panes: a "FILE EXPLORER" pane on the far left listing various Solidity files like CryptographicFunction.sol, Variable.sol, etc., and a "String.sol" file selected. The central pane displays the Solidity code for the "StringsExample" contract:

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

Below the code, a "ContractDefinition StringsExample" section shows "1 reference(s)". The bottom pane shows the assembly code:

```
0x00: RETURN
179 JUMPIST
180 PUSH1 60
182 PUSH1 08
184 PUSH1 00
196 SLOAD
```

```

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

```

## Array

### Code:

```

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

```

Remix - Ethereum IDE

Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity

DEBUGGER

DEBUGGER CONFIGURATION

Use generated sources (Solidity >= v0.7.2)

0x735733decb9b850d092d87c6aea4...

Stop debugging

Function Stack

0: dynamic\_array()

Solidity Locals

Solidity State

data: uint256[]

data1: int256[]

Assembly

214 RETURNDATASIZE

215 JUMPDEST

216 PUSH1 60

218 DUP1

219 PUSH1 40

221 MLOAD

222 MSTORE

223 POP

224 RETUREN

225 STOP

input 0x738...ed52a

decoded input {}

decoded output { "0": "uint256[]: [10, 20, 30, 40, 50]", "1": "int256[]: [-60, 70, -80, 90, -100, -120, 140]" }

logs []

## Enums

### Code:

Remix - Ethereum IDE

Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity

FILE EXPLORER

BitwiseOperator.sol

IA.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

LogicalOperator.sol

Dowhileloop.sol

ForLoop.sol

IfElse.sol

IfElseIf.sol

String.sol

Array.sol

Enum.sol

scripts

tests

README.txt

ContractDefinition Enum 1 reference(s) ▾

input 0 listen on all transactions Search with transaction hash or address

```
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 screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with options like 'DEPLOY & RUN TRANSACTIONS' (with a 'Deploy' button), 'Transactions recorded' (20), 'Deployed Contracts' (e.g., 'ENUM AT 0X332...D4B6D (MEMORY)'), and 'Low level interactions' (e.g., 'setLarge', 'getChoice', 'getDefaultCho...'). The main area displays the Solidity code for 'Enum.sol':

```
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);
    }
}
```

## Struct

### Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' sidebar open, displaying various Solidity files. The file 'Struct.sol' is currently selected. The main code editor shows the Solidity code for 'Struct.sol':

```
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;
    }
}
```

```
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', 4);
    }

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

## Mapping

### 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[0xDEE7796E89C82C36BAdd1375076f39069FafE252];

        student.name = "John";
        student.subject = "Chemistry";
        student.marks = 88;
        student_result.push(0xDEE7796E89C82C36BAdd1375076f39069FafE252) -1;
    }
}
```

```
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 {
        student memory student;
        student.name = "John";
        student.subject = "Chemistry";
        student.marks = 88;
        result[student] = student;
        student_result.push(student);
    }
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left is the FILE EXPLORER sidebar with various Solidity files listed. The main editor area displays a Solidity contract named `Mapping.sol`. The code defines a struct `student` and a mapping `result` from `address` to `student`. It includes a constructor and a public function `adding_values()` that adds a new entry to the mapping and pushes it to an array `student_result`. The right side of the interface features a DEBUGGER sidebar with sections for `Function Stack` and `Solidity Locals`. At the bottom, transaction logs are displayed, showing calls to the `adding_values()` function and their corresponding results.

## 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];
    }
}
```

This screenshot shows the Remix Ethereum IDE interface again. The FILE EXPLORER sidebar on the left shows multiple Solidity files, including `ArithmeticOperator.sol`, `WithdrawalPattern.sol`, and `SpecialVariable.sol`. The main editor area contains a Solidity contract named `SpecialVariable.sol`. The code defines a mapping `rollNo` from `address` to `uint`. It includes a public function `setRollNO` that updates the mapping with the sender's address and a value, and a public view function `whatIsMyRollNumber` that returns the value associated with the sender's address. The interface is similar to the first screenshot, with a DEBUGGER sidebar and transaction logs at the bottom.

```

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];
    }
}

```

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

### Function:

#### 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;
    }
}

```

```

pragma solidity ^0.5.0;

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

```

## Functions Modifiers

**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. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options like "Publish to IPFS" and "At Address". Below it, a section says "Transactions recorded 20" with a note about replaying transactions. In the center, under "Deployed Contracts", there's a list item "OWNER AT 0X5E1\_4EFF5(MEMORY)". On the right, the main code editor displays the Solidity code for the "Owner" contract:

```
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;
    }
}
```

## View function

### Code:

The screenshot shows the Remix Ethereum IDE with the "FILE EXPLORERS" sidebar open. The "Workspaces" section shows a workspace named "remixDefault\_1655651734405" containing several contracts like "1\_Secure.sol", "2\_Owner.sol", etc. The "Contracts" folder is expanded, showing files such as "View Function.sol", "PureFunction.sol", etc. The "View Function.sol" file is currently selected and displayed in the main code editor. The code for the "ViewFunction" contract is as follows:

```
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;
    }
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with various icons for deployment, running transactions, and interacting with contracts. The main area displays a Solidity code editor with the following content:

```
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 editor, there's a section titled "Deployed Contracts" which lists "VIEWFUNCTION AT 0xE25...4157A". Under this, it shows the "getResult" function with its parameters: "u: uint256 product 160" and "i: uint256 sum 26". There's also a "Low level interactions" section with a "CALLDATA" button and a "Tosact" button.

## Pure function

Code:

The screenshot shows the Remix Ethereum IDE interface with the "FILE EXPLORERS" sidebar open. The "Workspaces" section shows a workspace named "remixDefault\_1655651734405". In the file list, the "PureFunction.sol" file is selected and highlighted. The main code editor area contains the following Solidity code:

```
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 "Welcome to Remix 0.24.1" message is visible at the bottom of the interface.

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with icons for file operations like Open, Save, and Deploy. The main area has tabs for different Solidity files: Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol (which is currently selected), FallbackFunction.sol, and FunctionOverload. The PureFunction.sol tab shows the following Solidity code:

```

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;
    }
}

```

Below the code, the "Deployed Contracts" section shows a deployed contract named "PUREFUNCTION AT 0x1C9...2B4BD". It lists a single function call: "getResult" with parameters "0: uint256; product 20" and "1: uint256; sum 12". The "Low level interactions" section includes a "CALLDATA" button and a "Transact" button. At the bottom, a transaction log is displayed: "[vm] from: 0x583...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xe9...6a4d6". A "Debug" button is also present.

## Fallback function

**Code:**

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a "FILE EXPLORERS" panel with a tree view of workspace files, including contracts like 1\_Artifacts.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 (selected), FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, scripts, tests, and README.txt. The main workspace shows the FallbackFunction.sol code:

```

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));
    }
}

```

At the bottom, a transaction log is shown: "[vm] from: 0x583...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xe9...6a4d6". A "Debug" button is also present.

The screenshot shows the Remix Ethereum IDE interface. The central area displays Solidity code for function overloading:

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

The left sidebar shows the "Deploy & Run Transactions" section with tabs for "At Address" and "Load contract from Address". It displays a list of recorded transactions and deployed contracts, including "callSink" and "callTest".

## Function Overloading

**Code:**

The screenshot shows the Remix Ethereum IDE interface with the "FILE EXPLORERS" sidebar open. The "remixDefault\_1655651734405" workspace is selected, showing a list of files including "FunctionOverloading.sol". The main code editor window displays the following Solidity code for function overloading:

```
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 screenshot shows the Ethereum IDE interface. On the left, there's a sidebar with icons for deploying contracts, running transactions, and other tools. The main area displays a Solidity code editor for a file named `FunctionOverloading.sol`. The code defines a contract with three functions: `getSum` with two parameters, `callSumWithTwoArguments` with two parameters, and `callSumWithThreeArguments` with three parameters. All functions return a uint. Below the code editor, there's a section for "Deployed Contracts" showing two entries: `callSumWithT...` and `callSumWithT...`, each with a dropdown menu showing their respective signatures. At the bottom, there's a "Low level interactions" section with a "TRANSACTION" button.

```
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);
    }
}
```

## Mathematical Function

Code:

The screenshot shows the Ethereum IDE interface with a "FILE EXPLORERS" sidebar on the left. It lists several workspace files, including `remixDefault_1655651734405`, which contains multiple contracts like `1_Storage.sol`, `2_Owner.sol`, `3_Ballot.sol`, and `MathematicalFunction.sol`. The `MathematicalFunction.sol` file is currently selected and shown in the main code editor. The code defines a contract with two functions: `callAddMod` and `callMulMod`, both returning uint values by calling `addmod` and `mulmod` respectively with arguments 14, 15, and 13.

```
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);
    }
}
```

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's a message about SSL security and a list of recent files. The main workspace is titled 'DEPLOY & RUN TRANSACTIONS'. On the left, a sidebar displays 'Transactions recorded' (25) and 'Deployed Contracts' (MATHEMATICALFUNCTION AT 0X40). The central area contains the Solidity code for the 'MathematicalFunction' contract:

```
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);
    }
}
```

## 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's a message about SSL security and a list of recent files. The main workspace is titled 'FILE EXPLORERS'. On the left, a sidebar shows a file tree with workspaces and contracts like 'remixDefault\_1655651734405', '1.Storage.sol', '2\_Owner.sol', etc. The central area contains the Solidity code for the 'CryptographicFunction' contract:

```
pragma solidity ^0.5.0;

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

```

pragma solidity ^0.5.0;

contract CryptographicFunction {
    function callKecak256() public pure returns(bytes32 result){
        return keccak256("ABC");
    }
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with icons for deploying contracts, running transactions, and viewing logs. The main area has tabs for different Solidity files: PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, and CryptographicFunction.sol. The CryptographicFunction.sol tab is active, displaying the provided Solidity code. Below the code, the 'Transactions recorded' section shows one transaction: 'callKecak256' with a result of '0xe1629b9dd060bb30c7908346f6af189c16773f6148d3366701fbfaa35d54f3c8'. There are also sections for 'Deployed Contracts' and 'Low level interactions'.

### 3a) Withdrawal Pattern, Restricted Access.

#### Withdrawal Pattern

**Code:**

```

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);
    }
}

```

This screenshot shows the Remix Ethereum IDE with the 'WithdrawalPattern.sol' file selected in the 'FILE EXPLORER' sidebar. The code defines a contract with a constructor that sets the initial richest address and mostSent value. It includes a 'becomeRichest' function that updates the richest address and mostSent value if a higher value is sent. It also includes a 'withdraw' function that allows the current richest address to withdraw their pending withdrawal amount.

```

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);
        msg.sender.transfer(pendingWithdrawals[msg.sender]);
        delete pendingWithdrawals[msg.sender];
    }

    function mostSent() public view returns (uint) {
        return mostSent;
    }

    function richest() public view returns (address) {
        return richest;
    }
}

```

## 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, the FILE EXPLORER panel displays a workspace named "remixDefault\_1655651734405". Inside this workspace, there are several contracts and scripts. The "contracts" folder contains files like Storage.sol, Owner.sol, Ballot.sol, while loop.sol, while loop1.sol, wlSol, 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, and RestrictedAccess.sol. The "scripts" and "tests" folders are also visible. The central area shows two tabs: "WithdrawalPattern.sol" and "RestrictedAccess.sol". The "RestrictedAccess.sol" tab is active, displaying the following Solidity code:

```
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" panel open. This panel allows users to interact with deployed contracts. In the "Deployed Contracts" section, there is a single entry: "TEST AT 0xD7A...F71B (MEMORY)". Below this, under "Low level interactions", there are several buttons: "changeOwner" (orange), "disown" (orange), "forceOwnerCh..." (red), "creationTime" (blue), and "owner" (blue). The "owner" button is currently selected, showing its address: 0x6838Da6a701c568545dCfkB03Fc8875f56beddC4. The right side of the interface shows the same Solidity code as the previous screenshot.

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar for "DEPLOY & RUN TRANSACTIONS" with options like "Publish to IPFS" and "At Address". Below it is a "Deployed Contracts" section showing a deployed contract named "TEST AT 0XCD6...". The main area contains two tabs: "WithdrawalPattern.sol" and "RestrictedAccess.sol". The "WithdrawalPattern.sol" tab is active, displaying 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 transaction logs and a "Debug" button.

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

#### Contracts

##### Code:

The screenshot shows the Remix Ethereum IDE interface with the "FILE EXPLORER" sidebar open, displaying a workspace named "remixDefault\_1655651734405". Inside the workspace, several contracts are listed in the tree view, including "L\_Storage.sol", "2\_Owner.sol", "3\_Ballot.sol", "while\_loop1.sol", "whileloop1.sol", "w1.sol", "Function.sol", "Functions.Modifiers.sol", "View.Function.sol", "PureFunction.sol", "FallbackFunction.sol", "FunctionOverloading.sol", "MathematicalFunction.sol", "CryptographicFunction.sol", "Variable.sol", "RithwiseOperator.sol", "I1.sol", "AssignmentOperator.sol", "ArithmeticOperator.sol", "WithdrawalPattern.sol", "RestrictedAccess.sol", and "Contracts.sol". The "Contracts.sol" file is currently selected and its code is visible in the main editor area:

```

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 is the 'FILE EXPLORER' sidebar, which lists several Solidity files under 'Workspaces'. The main area is the code editor, displaying the 'WithdrawalPattern.sol' file. The code defines a derived contract 'C' that inherits from 'Contract E'. It includes a private result variable, a constructor that initializes it to zero, and two public view functions: 'getComputedResult' and 'getResult'.

```
//Derived Contract
contract E is C {
    uint private result;
}
contract C is E {
    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 'DEPLOY & RUN TRANSACTIONS' tab in the Remix interface. It displays the deployed contract 'C' at address 0x5FD...9D88D (MEMORY). The 'Transactions recorded' section shows a log entry for the deployment of 'C'. The 'Low level interactions' section shows the contract's methods: 'updateData', 'getData', and 'info'. The 'info' method is currently selected, showing its value as 0.

Deployed Contracts

- C AT 0x5FD...9D88D (MEMORY)

Transactions recorded

- [call] from: 0x58380ada701c56b545dCfc03Fc8875f56bedd4 to: Test.owner() data: 0x8da...5cb5b creation of C pending...

Low level interactions

- updateData
- getData
- info

info

0

```

//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(); }
}

//Contract C
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; }
}

```

## Inheritance

### Code:

```

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();
    }
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'FILE EXPLORER' sidebar displays a workspace named 'remixDefault\_1655651734405'. Inside this workspace, there is a folder 'contracts' containing several Solidity source files: 1\_Storage.sol, 2\_Owner.sol, 3\_Ballot.sol, while\_loop.sol, while\_if1.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 main central area is the code editor, which currently displays the content of the 'Inheritance.sol' file. The code defines a derived contract E that inherits from C, with various access modifiers and functions.

```
//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' tab selected. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar includes options for 'Deploy' (with a 'Publish to IPFS' checkbox), 'At Address' (selected), and 'Load contract from Address'. Below these are sections for 'Transactions recorded' (24) and 'Deployed Contracts'. Under 'Deployed Contracts', a contract named 'C AT 0X5E1\_4EFF5 (MEMORY)' is listed, with its address and storage slot (0x5e14eff5) shown. A dropdown menu for this contract reveals three functions: 'updateData' (with parameter 'a' of type uint256), 'getData' (with parameter 'a' of type uint256), and 'info' (with parameter 'a' of type uint256). The main central area is the code editor, which displays the Solidity code for the 'Inheritance.sol' contract, including the definition of contract C with its state variables and functions, and the derived contract E.

```
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 getComputedResult() public {
        result = compute(3, 5);
    }
    function getResult() public view returns(uint) { return result; }
    function getData() public view returns(uint) { 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();
    }
}

```

## 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 with options for 'DEPLOY & RUN TRANSACTIONS' and 'CONTRACT'. Under 'CONTRACT', 'constructorExample - contracts/Cons...' is selected. Below it, there are buttons for 'Deploy' and 'Publish to IPFS'. The main area shows the Solidity code for 'Constructor.sol':

```

pragma solidity ^0.5.0;

contract constructorExample {
    string str;

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

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

```

Below the code, the 'ContractDefinition C' section shows a pending transaction for the constructor. The transaction details are:

- [vm] from: 0x583...edd4 to: constructorExample.(constructor) value: 0 wei data: 0x600...10032 logs: 0
- call to constructorExample.getValue

At the bottom, there are two 'Debug' buttons.

## Abstract Contracts

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'FILE EXPLORER' sidebar lists several Solidity files under 'CONTRACTS' and other folders like 'artifacts', 'scripts', and 'tests'. The current file is 'AbstractConstructor.sol' in the 'Home' tab. The code defines an abstract contract 'abstractConstructor' and a derived contract 'Test' that implements it:

```

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;
    }
}

```

At the bottom, a transaction call is shown:

- [call] from: 0x583...a701c568545dCfcB03Fc8875f56beddC4 to: Test.getResult() data: 0xde2...92789

```

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 Remix interface shows the deployment of the contract at address `TEST AT 0xD91...39138 (MEMORY)`. A transaction record is visible in the bottom right:

- `[vm] from: 0x583...eddC4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x2a3...4b9ff`
- `[call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: Test.getResult() data: 0xde2...92789`

## Interfaces

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 Remix interface shows the deployment of the contract at address `TEST AT 0xD91...39138 (MEMORY)`. A transaction record is visible in the bottom right:

- `[vm] from: 0x583...eddC4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x137...54395`
- `[call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: Test.getResult() data: 0x137...54395`

```

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;
    }
}

```

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

#### Libraries

#### Code:

```

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;
    }
}

```

```

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;
    }
}

```

## Assembly

### Code:

```

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:

```

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)
        }
    }
}

```

The Remix interface shows the deployment status of the contract, with a Deploy button highlighted. The Transactions recorded section shows a single transaction for the constructor call. The Deployed Contracts section lists the deployed contract ASSEMBLY AT 0xD91...39138. The Low level interactions section shows a call to the add function with parameters a=17 and b=45.

## Events

### 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;
    }
}

ContractDefinition eventExample 1 reference(s)

```

The Remix interface shows the deployment status of the contract, with a Deploy button highlighted. The Transactions recorded section shows a transaction for the constructor call. The Deployed Contracts section lists the deployed contract Events. The Low level interactions section shows a call to the getValue function with parameters \_a=1 and \_b=2.

```

// 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;
    }
}

ContractDefinition eventExample 1 reference(s)

```

## Error Handling

Code:

```

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;
    }
}

```

Remix - Ethereum IDE

Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13js&language=Solidity

Gmail YouTube Maps tf.keras.Model Te...

DEPLOY & RUN TRANSACTIONS

Deploy

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded

Deployed Contracts

ERRORHANDLING AT 0X90d7...B5B

checkinput

\_input: 243 call

0: string: Input is Uint8

Odd

\_input: 243 call

0: bool: true

Low level interactions

ErrorHandling.sol

```
pragma solidity ^0.5.0;

contract ErrorHandling {

    function checkInput(uint _input) public view returns(string memory){
        require(_input >= 0, "invalid uint8");
        require(_input <= 255, "invalid uint8");

        return "Input is Uint8";
    }

    function Odd(uint _input) public view returns(bool){
        require(_input % 2 != 0);
        return true;
    }
}
```

listen on all transactions Search with transaction hash or address

call [call] from: 0x5838Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.checkInput(uint256) data: 0x021...000F3 Debug

call to ErrorHandling.Odd

call [call] from: 0x5838Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.Odd(uint256) data: 0x922...000F3 Debug

## Practical 6

**Aim: Demonstrate the use of Bitcoin Core API.**

**Code:**

```
from hashlib import sha256
MAX_NONCE = 100000000000

def SHA256(text):
    return sha256(text.encode("ascii")).hexdigest()

def mine(block_number, transactions, previous_hash, prefix_zeros):
    prefix_str = '0'*prefix_zeros
    for nonce in range(MAX_NONCE):
        text = str(block_number) + transactions + previous_hash + str(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 has after trying {MAX_NONCE} times")

if __name__=='__main__':
    transactions='''Dhaval->Bhavin->20,
Mando->Cara->45
'''

    difficulty=4 # try changing this to higher number and you will see it will take more time for mining as difficulty increases
    import time
    start = time.time()
    print("start mining")
    new_hash = mine(5,transactions,'0000000xa036944e29568d0cff17edbe038f81208fecf9a66be9a2b8321c6ec7', difficulty)
    total_time = str((time.time() - start))
    print(f"end mining. Mining took: {total_time} seconds")
    print(new_hash)
```

```
start mining
Yay! Successfully mined bitcoins with nonce value:2425
end mining. Mining took: 0.011358022689819336 seconds
0000de957fbfdfc77582e0d0b20c53d2d1d83d8bb8cf3693521f672bf2a6021
```

## Practical 7

**Aim:** Create your own blockchain and demonstrate its use.

**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(se
```

```
lf._private_key)
    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER'
)).decode('ascii')
Dinesh = Client()
```

```
print (Dinesh.identity)
```

**Output:**

```
30819f300d06092a864886f70d010101050003818d0030818902818100eea39d3e40f737cff2050d40515a4833c80cdd9073d88a3629aef45ee77793c78197eb0f8bbf0688c0672e22ee74e691bd53668
```

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

**Code:**

```
class Transaction:  
    def __init__(self,  
                 sender,  
                 recipient,  
                 value):  
        self.sender =  
        sender  
        s  
        e  
        l  
        f  
        .  
        r  
        e  
        c  
        i  
        p  
        i  
        e  
        n  
        t  
        =  
        r  
        e  
        c  
        i  
        p  
        i  
        e  
        n  
        t  
        s  
        e  
        l  
        f  
        .  
        v  
        a
```

```

l                               a
u                               l
e                               u
=                               e
v                               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)

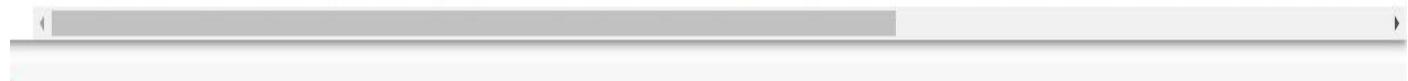
signature = t.sign_transaction()

print (signature)

```

**Output:**

```
0f18c12abcf2904299162ce5c39689df19a9bf583689ec58182cac42617da5b1282d467b6a92ffd358f33052abbeef502d1bbfcfcde293962e8ecec7b9f6a249df1bfc537a19404913e73079fc58c1f
```



### 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: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a
-----
value: 15.0
-----
time: 2022-07-30 17:35:06.809777
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbec4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a72cfb527dff9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2b
-----
value: 6.0
-----
time: 2022-07-30 17:35:06.811866
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a81
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d578
-----
value: 2.0
-----
time: 2022-07-30 17:35:06.814247
-----

sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a81
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d578
-----
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, previous_hash):
        self.previous_hash = previous_hash
        self.timestamp = time.time()
        self.data = []
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        sha = hashlib.sha256()
        hashable_data = str(self.previous_hash).encode() + str(self.timestamp).encode() + str(self.data).encode()
        sha.update(hashable_data)
        return sha.hexdigest()

    def add_transaction(self, transaction):
        self.data.append(transaction)

    def __str__(self):
        return f"Block Hash: {self.hash}\nPrevious Hash: {self.previous_hash}\nTimestamp: {self.timestamp}\nData: {self.data}\nHash: {self.hash}"
```

```
c          o
t          c
i          k
o          -
n          h
s          a
=
[          s
]          h
s          =
e          "
e          "
l          s
f          e
.
p          l
r          f
e          .
v          N
i          o
o          n
u          c
s          e
-
b          =
l          "
          "
last_block_hash = ""

Dinesh = Client()

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

block0 = Block()

block0.previous_block_hash = None
Nonce = None

block0.verified_transactions.append (t0)

digest = hash (block0)
last_block_hash = digest

TPCoins = []

def dump_blockchain (self):
    print ("Number of blocks in the chain: " + str(len (self)))
```

```

for x in range (len(TPCoins)):
    block_temp = TPCoins[x]
    print ("block # " + str(x))
    for transaction in block_temp.verified_transactions:
        display_transaction (transaction)
    print ('- -----')
    print ('=====') 

TPCoins.append (block0)
dump_blockchain(TPCoins)

```

**Output:**

---

```

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



**1e) Create a mining function and test it.**

**Code:**

```

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

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

mine ("test message", 2)

```

**Output:**

```
after 238 iterations found nonce: 1124711db6185591392c6a06c24a3c2ebbaeca647fb8374824f939ada27c09e9
after 353 iterations found nonce: 11e0a4b57bb76496ecc6ab5a3c126165bc9dbbaf80f664e7e192a422360c3884
after 419 iterations found nonce: 11280dfd9ab05b3dbfa869990153732941408faa4a3b0832819b161f52321c08
after 511 iterations found nonce: 1150944bd7ea429acd052da390b148f27eb881686e0f8bfcdf77f833af878e2e
after 822 iterations found nonce: 110e61d41d94b48f6dfcb6f43f9ceafe2ab7168456dab088e05126d43d0366ef
after 924 iterations found nonce: 11147bf32bafeee4505b5cc40c3b227366d8a41ab3bd740e437c74400b7a8127
'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: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539ccb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa

```

## **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 left sidebar displays a file explorer with various Solidity files and scripts. The main editor window shows the code for `Variable.sol`:

```
pragma solidity ^0.5.0;
contract Variable {
    uint storedData; // State variable
    constructor() public {
        storedData = 40;
    }
    function getResult() public view returns(uint){
        uint a = 1; // local variable
        uint b = 5;
        uint result = a + b;
        return storedData; //access the state variable
    }
}
```

**Output:**

## Operators:

### Arithmetic Operator

**Code:**

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with various Solidity files and scripts. The main editor window shows the code for `ArithmeticOperator.sol`:

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

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

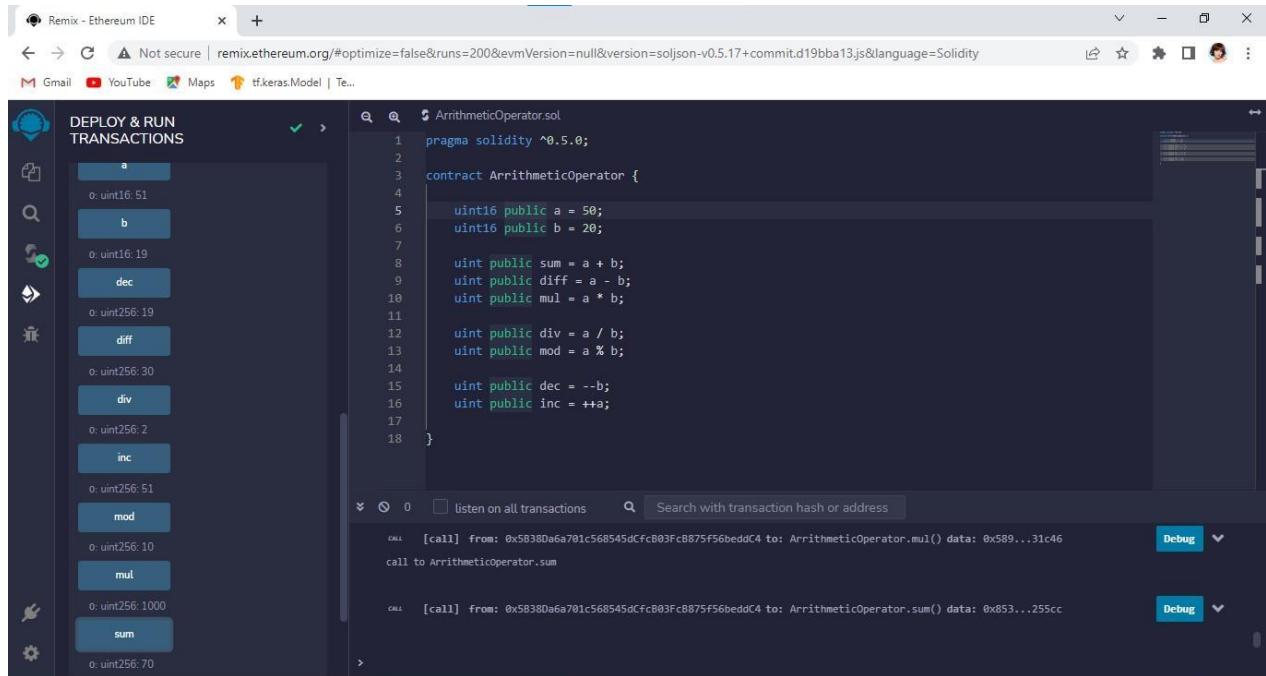
    uint public div = a / b;
    uint public mod = a % b;

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

At the bottom, the terminal output shows deployment logs:

```
[vm] from: 0x5B3...eddC4 to: SolidityTest.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x95b...f24a5 creation of ArithmetricOperator pending...
[vm] from: 0x5B3...eddC4 to: ArithmetricOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x6fa...e48ac
```

## Output:



The screenshot shows the Ethereum IDE interface with the title "Remix - Ethereum IDE". The left sidebar has a "DEPLOY & RUN TRANSACTIONS" section containing buttons for various arithmetic operations: add, sub, mul, div, mod, inc, dec, and sum. The main code editor displays the Solidity code for the `ArithmeticOperator` contract:

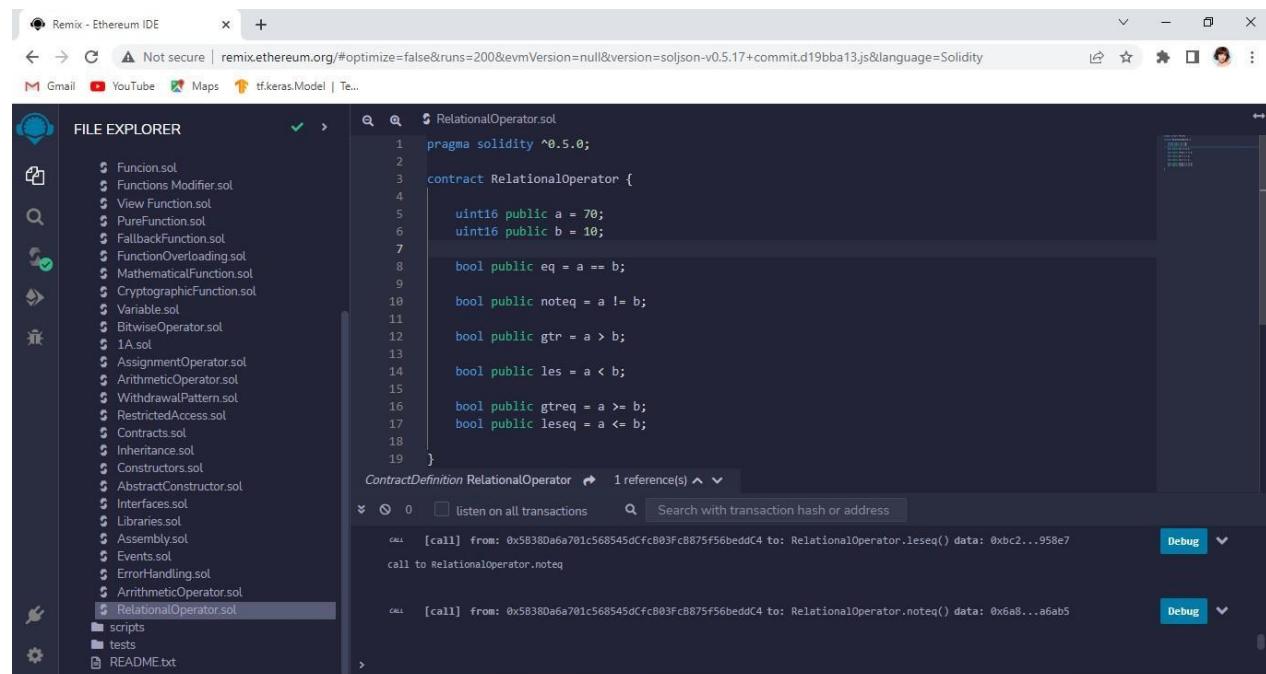
```
pragma solidity ^0.5.0;
contract ArithmeticOperator {
    uint16 public a = 50;
    uint16 public b = 20;
    uint public sum = a + b;
    uint public diff = a - b;
    uint public mul = a * b;
    uint public div = a / b;
    uint public mod = a % b;
    uint public dec = --b;
    uint public inc = ++a;
}
```

The bottom pane shows transaction logs:

- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ArithmeticOperator.mul() data: 0x589...31c46 call to ArithmeticOperator.mul
- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ArithmeticOperator.sum() data: 0x853...255cc call to ArithmeticOperator.sum

## Relational

### Operator Code:



The screenshot shows the Ethereum IDE interface with the title "Remix - Ethereum IDE". The left sidebar has a "FILE EXPLORER" section listing various Solidity files. The main code editor displays the Solidity code for the `RelationalOperator` contract:

```
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 bottom pane shows transaction logs:

- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.leseq() data: 0xbc2...958e7 call to RelationalOperator.leseq
- CALL [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.noteq() data: 0xa8...a6ab5 call to RelationalOperator.noteq

## Output:

```
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;
}
```

## Logical Operator

### Code:

```
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. On the left, there's a sidebar with options like 'DEPLOY & RUN TRANSACTIONS' (with a 'Deploy' button), 'Publish to IPFS', and 'At Address'. Below that is a 'Transactions recorded' section with 12 items. Under 'Deployed Contracts', there's a section for 'LOGICALOPERATOR AT 0XB27...' which includes a 'Logic' section with inputs 'a: true' and 'b: false', and a 'call' button. At the bottom, there's a 'Low level interactions' section. The main right panel displays the Solidity code for 'LogicalOperator.sol'.

```

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. The left sidebar has a 'FILE EXPLORER' section with various Solidity files listed, including 'LogicalOperator.sol' and 'BitwiseOperator.sol'. The main right panel shows the Solidity code for 'BitwiseOperator.sol'. It defines a contract with several bitwise operations: and, or, xor, leftshift, rightshift, and not. Below the code, there's a transaction history and a search bar.

```

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;
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" which lists various logical operators like and, or, xor, etc., each with their respective hex values. The main code editor contains the "BitwiseOperator.sol" file:

```
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;
}
```

The status bar at the bottom indicates a successful deployment with the message "[vm] from: 0x583...eddC4 to: LogicalOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x5df...62867".

## Assignment Operator

Code:

The screenshot shows the Remix Ethereum IDE interface with the "FILE EXPLORER" sidebar open, displaying multiple Solidity files. The "AssignmentOperator.sol" file is selected in the sidebar.

The code editor contains the "AssignmentOperator.sol" file:

```
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 = 50;
    uint16 public assignment_mod = 32;

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

The status bar at the bottom indicates a successful deployment with the message "[vm] from: 0x583...eddC4 to: AssignmentOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xdal...5c92c".

```

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 = 50;
    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:

```

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;
    }
}

```

```

1 pragma solidity ^0.5.0;
2
3 contract Types {
4
5     uint[] data;
6     uint8 j = 0;
7
8     function loop(
9         ) public returns(uint[] memory){
10        while(j < 5) {
11            j++;
12            data.push(j);
13        }
14        return data;
15    }
16
17 }
18

```

The screenshot shows the Remix IDE interface with the 'wl.sol' file open. The code defines a contract 'Types' with a single function 'loop'. This function pushes integers from 0 to 4 onto a dynamic array 'data'. The debugger sidebar on the left shows the 'Function Stack' and 'Solidity State'. The 'Solidity State' panel displays the memory state at each step of the loop, showing the array 'data' growing from length 0 to 4, and the variable 'j' increasing from 0 to 4. The assembly code pane at the bottom shows the EVM assembly for the current step.

## Dowhile loop

### Code:

```

1 pragma solidity ^0.5.0;
2
3 contract Dowhileloop {
4
5     uint[] data;
6     uint8 j = 0;
7
8     function loop(
9         ) public returns(uint[] memory){
10        do{
11            j++;
12            data.push(j);
13        }while(j < 8);
14        return data;
15    }
16
17 }
18

```

The screenshot shows the Remix IDE interface with the 'Dowhileloop.sol' file open. The code defines a contract 'Dowhileloop' with a single function 'loop'. This function uses a do-while loop to push integers from 0 to 7 onto a dynamic array 'data'. The debugger sidebar on the left shows the 'Function Stack' and 'Solidity State'. The 'Solidity State' panel displays the memory state at each step of the loop, showing the array 'data' growing from length 0 to 8, and the variable 'j' increasing from 0 to 8. The assembly code pane at the bottom shows the EVM assembly for the current step.

### Output:

```
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;
    }
}
```

## For loop

Code:

```
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 screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a 'DEBUGGER' panel with buttons for 'Stop debugging' and navigation controls, along with sections for 'Function Stack' and 'Solidity Locals'. The main code editor displays the Solidity source 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 bottom status bar shows transaction logs and a 'Debug' button.

## Decision making

### If statement

#### Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a 'FILE EXPLORER' panel listing various Solidity files. The main code editor displays the Solidity source code for 'IfStatement.sol':

```

pragma solidity ^0.5.0;

contract IfStatement {
    uint i = 20;

    function decision_making()
        public returns(bool)
    {
        if(i>20){
            return true;
        }
    }
}

```

The bottom status bar shows transaction logs and a 'Debug' button.

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEBUGGER' panel is open, showing a function stack with 'o.\_decision\_making()' and various Solidity locals and state variables. The main code editor contains the following Solidity code:

```
pragma solidity ^0.5.0;
contract IfStatement {
    uint i = 20;
    function decision_making() public returns(bool){
        if(i>0){
            return true;
        }
    }
}
```

The right side of the interface shows the EVM call trace with the following details:

- Input: 0x887...3af24
- decoded input: ()
- decoded output: { "0": "bool: false" }
- logs: []
- val: 0 wei

## If...else

### statement Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' panel open, displaying a list of Solidity files. The code editor on the right contains the following Solidity code for an 'IfElse' contract:

```
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 bottom status bar indicates an execution cost of 22744 gas.

### Output:

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with icons for file operations, a magnifying glass for search, and a gear for settings. The main area has tabs for 'IfStatement.sol' and 'IfElse.sol'. The code editor displays the 'IfElse.sol' file:

```
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 right side of the interface includes a 'DEBUGGER' panel with sections for 'FUNCTION STACK' (containing 'o: decision\_making()'), 'SOLIDITY LOCALS' (showing 'i: 20 uint256' and 'even: true bool'), and 'SOLIDITY STATE' (displaying assembly code). Below the code editor is a 'ContractDefinition IfElse' section with a reference count of '1 reference(s)'.

## If...else if...else statement

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a 'FILE EXPLORER' panel listing various Solidity files like MathematicalFunction.sol, CryptographicFunction.sol, etc. The main area has tabs for 'IfElseIf.sol' and 'IfElse.sol'. The code editor displays the 'IfElseIf.sol' file:

```
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;
    }
}
```

The right side of the interface includes a 'FILE EXPLORER' panel with sections for 'scripts', 'tests', and 'README.txt'. Below the code editor is a 'ContractDefinition IfElseIf' section with a reference count of '1 reference(s)'.

The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and a URL "Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity". The sidebar on the left contains icons for file operations, search, and settings. The main area is divided into several panes: a "DEBUGGER" pane on the far left with sections for "DEBUGGER CONFIGURATION" (checkbox for "Use generated sources (Solidity >= v0.7.2)" checked), "Function Stack" (showing "o. decision\_making()"), "Solidity Locals" (empty), and "Solidity State" (showing "i:10 uint256" and "result: equal to 10 string"). The central pane displays the Solidity code for the "IfElseIf" contract:

```
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;
    }
}
```

Below the code, a "ContractDefinition IfElseIf" section shows "1 reference(s)". The bottom pane shows the assembly code with highlighted lines 179-196 corresponding to the "else if(i == 10){ result = "equal to 10"; }" block.

## String

### Code:

The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and a URL "Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity". The sidebar on the left contains icons for file operations, search, and settings. The main area is divided into several panes: a "FILE EXPLORER" pane on the far left listing various Solidity files like CryptographicFunction.sol, Variable.sol, etc., and a "String.sol" file selected. The central pane displays the Solidity code for the "StringsExample" contract:

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

Below the code, a "ContractDefinition StringsExample" section shows "1 reference(s)". The bottom pane shows the assembly code with highlighted lines 179-196 corresponding to the "text = "Hello World";" assignment.

```

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

```

## Array

### Code:

```

// Creating a contract
contract ArrayExample {
    uint[] data
    int[] data1;
    function dynamic_array() public returns(
        uint[] memory, int[] memory){
        data
        data1
        return (data, data1);
    }
}

```

Remix - Ethereum IDE

Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity

DEBUGGER

DEBUGGER CONFIGURATION

Use generated sources (Solidity >= v0.7.2)

0x735733decb9b850d092d87c6aea4...

Stop debugging

Function Stack

0: dynamic\_array()

Solidity Locals

Solidity State

data: uint256[]

data1: int256[]

EVM CALL TRACE

214 RETURNDATASIZE

215 JUMPDEST

216 PUSH1 60

218 DUP1

219 PUSH1 40

221 MLOAD

222 MSTORE

223 POP

224 RETUREN

225 STOP

input 0x738...ed52a

decoded input {}

decoded output { "0": "uint256[]: [10, 20, 30, 40, 50]", "1": "int256[]: [-60, 70, -80, 90, -100, -120, 140]" }

logs []

## Enums

### Code:

Remix - Ethereum IDE

Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js&language=Solidity

FILE EXPLORER

BitwiseOperator.sol

IA.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

LogicalOperator.sol

Dowhileloop.sol

ForLoop.sol

IfElse.sol

IfElseIf.sol

String.sol

Array.sol

Enum.sol

scripts

tests

README.txt

ContractDefinition Enum 1 reference(s) ▾

ContractDefinition Enum 1 reference(s) ▾

input 0

Search with transaction hash or address

29°C Rain showers 3:41 PM 7/24/2022

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with options for 'DEPLOY & RUN TRANSACTIONS' (Deploy, Publish to IPFS), 'Transactions recorded' (20), 'Deployed Contracts' (ENUM AT 0X332...D4B6D (MEMORY)), and 'Low level interactions' (CALLDATA). The main area displays the Solidity code for 'Enum.sol':

```
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);
    }
}
```

Below the code, there are three buttons: 'setLarge', 'getChoice', and 'getDefaultCho...'. The status bar at the bottom right shows the date and time as 7/24/2022 3:43 PM.

## Struct

### Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' sidebar open, displaying various Solidity files like 'IA.sol', 'AssignmentOperator.sol', etc. The main code editor shows the Solidity code for 'Struct.sol':

```
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;
    }
}
```

The status bar at the bottom right shows the date and time as 7/24/2022 3:47 PM.

```
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', 4);
    }

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

## Mapping

### 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[0xDEE7796E89C82C36BAdd1375076f39069FaffE252];

        student.name = "John";
        student.subject = "Chemistry";
        student.marks = 88;
        student_result.push(0xDEE7796E89C82C36BAdd1375076f39069FaffE252) -1;
    }
}
```

```
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 {
        student memory student;
        student.name = "John";
        student.subject = "Chemistry";
        student.marks = 88;
        result[student] = student;
        student_result.push(student);
    }
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left is the 'DEBUGGER' sidebar with sections for 'Function Stack' and 'Solidity Locals'. The main area displays a Solidity contract named 'Mapping.sol'. The code defines a struct 'student' and a mapping 'result' from address to 'student'. It also contains a constructor-like function 'adding\_values()' that adds a new entry to the mapping and pushes it to an array 'student\_result'. Transaction logs at the bottom show two entries: one for the constructor call and another for the 'adding\_values()' function call.

## 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];
    }
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left is the 'FILE EXPLORER' sidebar listing several Solidity files: 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, LogicalOperator.sol, Dowhileloop.sol, ForLoop.sol, IfStatement.sol, IfElse.sol, IfSelf.sol, String.sol, Array.sol, Enum.sol, Struct.sol, Mapping.sol, and SpecialVariable.sol. The main area displays a Solidity contract named 'SpecialVariable.sol'. It contains a mapping from address to uint named 'rollNo' and two functions: 'setRollNO' which sets the value in the mapping, and 'whatIsMyRollNumber' which returns the value from the mapping.

```

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];
    }
}

```

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

### Function:

#### 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;
    }
}

```

```

pragma solidity ^0.5.0;

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

```

## Functions Modifiers

**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:**

```
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) {
            ;
        } else {
            revert();
        }
    }
    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;
        }
    }
}
```

## View function

### Code:

```
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;
    }
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with various icons for deployment, running transactions, and interacting with contracts. The main area displays a Solidity code editor with the following content:

```
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 editor, there's a section titled "Deployed Contracts" which lists "VIEWFUNCTION AT 0xE25...4157A". Under this, it shows the "getResult" function with its parameters: "u: uint256 product 160" and "i: uint256 sum 26". There's also a "Low level interactions" section with a "CALLDATA" button and a "Tosact" button.

## Pure function

Code:

The screenshot shows the Remix Ethereum IDE interface with the "FILE EXPLORERS" sidebar open. The "Workspaces" section shows a workspace named "remixDefault\_1655651734405". In the file list, the "PureFunction.sol" file is selected and highlighted. The main code editor area contains the following Solidity code:

```
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 "Welcome to Remix 0.24.1" message is visible at the bottom of the interface.

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with icons for file operations like Open, Save, and Deploy. The main area has tabs for different Solidity files: Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol (which is currently selected), FallbackFunction.sol, and FunctionOverload. The PureFunction.sol tab displays the following Solidity code:

```

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;
    }
}

```

Below the code, the "Deployed Contracts" section shows a deployed contract named "PUREFUNCTION AT 0x1C9...2B4BD". It lists a single function call: "getResult" with parameters "0: uint256; product 20" and "1: uint256; sum 12". The "Low level interactions" section includes a "CALLDATA" button and a "Transact" button. At the bottom, a transaction log is shown: "[vm] from: 0x583...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xe9...6a4d6". A "Debug" button is also present.

## Fallback function

**Code:**

The screenshot shows the Remix Ethereum IDE interface. The left sidebar shows a "FILE EXPLORERS" section with a workspace named "remixDefault\_1655651734405" containing contracts like 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 (selected), FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, scripts, tests, and README.txt. The main area has tabs for Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol (selected), and FunctionOverload. The FallbackFunction.sol tab displays the following Solidity code:

```

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));
    }
}

```

At the bottom, a transaction log is shown: "[vm] from: 0x583...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xe9...6a4d6". A "Debug" button is also present.

The screenshot shows the Remix Ethereum IDE interface. The central area displays Solidity code for function overloading:

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

The left sidebar shows the "Deploy & Run Transactions" section with tabs for "At Address" and "Load contract from Address". It displays a list of recorded transactions and deployed contracts, including "callSink" and "callTest".

## Function Overloading

**Code:**

The screenshot shows the Remix Ethereum IDE interface with the "FILE EXPLORERS" sidebar open. The "remixDefault\_1655651734405" workspace is selected, showing a list of files including "FunctionOverloading.sol". The main code editor window displays the following Solidity code for function overloading:

```
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 screenshot shows the Ethereum IDE interface. On the left, there's a sidebar with icons for deploying contracts, running transactions, and other tools. The main area displays a Solidity code editor for a file named `FunctionOverloading.sol`. The code defines a contract with three functions: `getSum` with two parameters, `callSumWithTwoArguments` with two parameters, and `callSumWithThreeArguments` with three parameters. All functions return a uint. Below the code editor, there's a section for "Deployed Contracts" showing two entries: `callSumWithT...` and `callSumWithT...`, each with a dropdown menu showing their respective signatures. At the bottom, there's a "Low level interactions" section with a "TRANSACTION" button.

```
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);
    }
}
```

## Mathematical Function

Code:

The screenshot shows the Ethereum IDE interface with a "FILE EXPLORERS" sidebar on the left. The sidebar lists several workspace files, including `remixDefault_1655651734405`, which contains sub-folders like `contracts` and `artifacts`, and various Solidity source files such as `1_A.sol`, `1B.sol`, `while.loop.sol`, `while.loop1.sol`, `wl.sol`, `Function.sol`, `Functions.Modifier.sol`, `View.Function.sol`, `PureFunction.sol`, `FallbackFunction.sol`, `FunctionOverloading.sol`, `MathematicalFunction.sol`, `CryptographicFunction.sol`, `scripts`, `tests`, and `README.txt`. The `MathematicalFunction.sol` file is currently selected and shown in the main code editor. The code defines a contract with two functions: `callAddMod` and `callMulMod`, both returning uint values by calling the `addmod` and `mulmod` library functions respectively.

```
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);
    }
}
```

The screenshot shows the Remix 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". The sidebar on the left is titled "DEPLOY & RUN TRANSACTIONS" and includes sections for "Transactions recorded" (25), "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.", and "Deployed Contracts" (MATHEMATICALFUNCTION AT 0x40). The main workspace contains the Solidity code for the MathematicalFunction contract:

```
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);
    }
}
```

## Cryptographic function

Code:

The screenshot shows the Remix 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". The sidebar on the left is titled "FILE EXPLORERS" and shows a workspace named "remixDefault\_1655651734405" containing several contracts like 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, and CryptographicFunction.sol. The main workspace contains the Solidity code for the CryptographicFunction contract:

```
pragma solidity ^0.5.0;

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

```

pragma solidity ^0.5.0;

contract CryptographicFunction {
    function callKecak256() public pure returns(bytes32 result){
        return keccak256("ABC");
    }
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with icons for deploying contracts, running transactions, and viewing logs. The main area has tabs for different Solidity files: PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, and CryptographicFunction.sol. The CryptographicFunction.sol tab is active, displaying the provided Solidity code. Below the code, the 'Transactions recorded' section shows one transaction: 'callKecak256' with a result of '0xe1629b9dd060bb30c7908346f6af189c16773f6148d3366701fbfaa35d54f3c8'. There are also sections for 'Deployed Contracts' and 'Low level interactions'.

### 3a) Withdrawal Pattern, Restricted Access.

#### Withdrawal Pattern

**Code:**

```

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);
    }
}

```

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' sidebar open. The 'WithdrawalPattern.sol' file is selected in the workspace. The code for the 'WithdrawalPattern' contract is displayed in the main editor. The contract has a constructor that sets the initial richest address and mostSent value. It includes a 'becomeRichest' function that updates the richest address and mostSent value if the caller sends more funds. It also includes a 'withdraw' function that allows the current richest address to withdraw their pending withdrawal amount.

```

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);
        msg.sender.transfer(pendingWithdrawals[msg.sender]);
        delete pendingWithdrawals[msg.sender];
    }

    function mostSent() public view returns (uint) {
        return mostSent;
    }

    function richest() public view returns (address) {
        return richest;
    }
}

```

## 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, the FILE EXPLORER panel displays a workspace named "remixDefault\_1655651734405". Inside this workspace, there are several contracts and scripts. The "contracts" folder contains files like Storage.sol, Owner.sol, Ballot.sol, while loop.sol, while loop1.sol, wlSol, 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, and RestrictedAccess.sol. The "scripts" and "tests" folders are also visible. On the right, the code editor shows two files: WithdrawalPattern.sol and RestrictedAccess.sol. The RestrictedAccess.sol file contains the following Solidity code:

```
27 modifier costs(uint _amount) {
28     require(
29         msg.value >= _amount,
30         "Not enough Ether provided."
31     );
32     if (msg.value > _amount)
33         msg.sender.transfer(msg.value - _amount);
34 }
35 function forceOwnerChange(address _newOwner) public payable costs(200 ether) {
36     owner = _newOwner;
37     if (uint(owner) & 0 == 1) return;
38 }
39 }
```

## Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" panel active. This panel allows users to interact with deployed contracts. In the "Deployed Contracts" section, there is a single entry: "TEST AT 0xD7A...F71B (MEMORY)". Below this, under "Low level interactions", there are several buttons: "changeOwner" (orange), "disown" (orange), "forceOwnerCh..." (red), "creationTime" (blue), and "owner" (blue). The "owner" button is currently selected, showing its address: 0x6838Da6a701c568545dCfkB03Fc8875f56beddC4. The code editor on the right shows the same Solidity code as the previous screenshot.

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar for 'DEPLOY & RUN TRANSACTIONS' with options like 'Publish to IPFS' and 'At Address'. Below it is a 'Deployed Contracts' section showing a deployed contract named 'TEST AT 0XCD6...'. The main area contains two tabs: 'WithdrawalPattern.sol' and 'RestrictedAccess.sol'. The 'WithdrawalPattern.sol' tab has the following 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."
        );
    }
}

```

The 'RestrictedAccess.sol' tab is also visible. At the bottom, there are transaction logs and a 'Debug' button.

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

#### Contracts

##### Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' sidebar open, displaying several workspace files. The 'Contracts.sol' file is currently selected and open in the code editor. The code for 'Contracts.sol' is as follows:

```

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 is the 'FILE EXPLORER' sidebar, which lists several Solidity files under 'Workspaces'. The main area is the code editor, showing the 'WithdrawalPattern.sol' file with the following code:

```
//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' tab selected. The left sidebar shows deployment options like 'Deploy' and 'Publish to IPFS'. The main area displays the deployed contract 'C' at address 0x5FD...9D88D (MEMORY). It shows interactions such as 'updateData' (with value 0), 'getData' (with value 0), and 'info' (with value 10). Below this, the 'Low level interactions' section shows a 'CALLDATA' entry with a 'Transact' button. The transaction history on the right shows two transactions:

- [call] from: 0x58380ada701c56b545dCfc03Fc8875f56bedd4 to: Test.owner() data: 0x8da...5cb5b creation of C pending...
- [vm] from: 0x583...eddc4 to: C.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0xd42...f1745 transact to C.updateData errored: Error encoding arguments: Error: invalid BigNumber string (argument="value", value="", code=INVALID\_ARGUMENT, version=b)

```

//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(); }
}

//Contract C
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; }
}

```

## Inheritance

### Code:

```

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();
    }
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'FILE EXPLORER' sidebar displays a workspace named 'remixDefault\_1655651734405'. Inside this workspace, there is a folder 'contracts' containing several Solidity source files: 1\_Storage.sol, 2\_Owner.sol, 3\_Ballot.sol, while\_loop.sol, while\_if1.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 main central area is the code editor, which currently displays the content of the 'Inheritance.sol' file. The code defines a derived contract E that inherits from C. It includes private state variables, a constructor, and several public functions like getComputedResult(), getResult(), and getdata(). The right side of the interface features a small preview window showing the deployed contracts.

```
//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:

This screenshot shows the 'DEPLOY & RUN TRANSACTIONS' tab in the Remix Ethereum IDE. On the left, there's a deployment interface with a 'Deploy' button and options to publish to IPFS or deploy at a specific address. Below this, the 'Transactions recorded' section shows 24 transactions. Under 'Deployed Contracts', it lists 'C AT 0X5E1\_4EFF5 (MEMORY)'. This contract has three methods listed: updateData(uint256 a), getData(), and info(). The 'Low level interactions' section shows the contract's address and a 'Transact' button. The central code editor area shows the full Solidity code for both contracts C and E, including their state variables, constructors, and various functions. The right side of the interface continues to show the deployed contract preview.

```
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 getComputedResult() public {
        result = compute(3, 5);
    }
    function getResult() public view returns(uint) { return result; }
    function getData() public view returns(uint) { 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();
    }
}

```

## 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 with options for 'DEPLOY & RUN TRANSACTIONS' and 'CONTRACT'. Under 'CONTRACT', 'constructorExample - contracts/Cons...' is selected. Below it, there are buttons for 'Deploy' and 'Publish to IPFS'. The main area shows the Solidity code for 'Constructor.sol':

```
pragma solidity ^0.5.0;

contract constructorExample {
    string str;

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

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

Below the code, the 'ContractDefinition C' section shows 4 reference(s). It includes a log entry: '[vm] from: 0x583...edd4 to: constructorExample.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0xe3c...1809c call to constructorExample.getValue'. There are also 'Debug' buttons for both the constructor and the function call.

## Abstract Contracts

Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' sidebar open, displaying various Solidity files. In the main editor area, 'AbstractConstructor.sol' is open with the following 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 'ContractDefinition C' section at the bottom shows a log entry: '[call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: Test.getResult() data: 0xde2...92789'. There is a 'Debug' button for this call.

```

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 Remix interface shows the deployment of the contract at address 0xD91...39138 (MEMORY). A transaction record is visible in the bottom right.

## Interfaces

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 Remix interface shows the deployment of the contract at address 0x583...eddC4. A transaction record is visible in the bottom right.

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with options for 'DEPLOY & RUN TRANSACTIONS' (Deploy, Publish to IPFS), 'CONTRACT' (Test - contracts/Interfaces.sol), and 'Low level interactions' (CALLDATA, Transect). The main area displays the Solidity code for 'Interfaces.sol':

```

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;
    }
}

```

Below the code, the 'Transactions recorded' section shows a single transaction: 'TEST AT 0xD8B...33FA8 (MEMORY)'. It has a button labeled 'getResult' which shows the value '0: uint256: 78'. The 'Low level interactions' section shows a 'CALL' to the contract with the same result.

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

#### Libraries

#### Code:

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORER' sidebar open, displaying various Solidity files like 2\_Owners.sol, 3\_Ballot.sol, etc. The current file being edited is 'Libraries.sol'.

The code for 'Libraries.sol' is as follows:

```

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;
    }
}

```

```

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;
    }
}

```

## Assembly

### Code:

```

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:

```

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)
        }
    }
}

```

The Remix interface shows the deployment status of the contract, with a Deploy button highlighted. The Transactions recorded section shows a single transaction for the constructor call. The Deployed Contracts section lists the deployed contract ASSEMBLY AT 0xD91...39138. The Low level interactions section shows a call to the add function with parameters a=17 and b=45.

## Events

### 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;
    }
}

ContractDefinition eventExample 1 reference(s)

```

The Remix interface shows the deployment status of the contract, with a Deploy button highlighted. The Transactions recorded section shows a transaction for the constructor call. The Deployed Contracts section lists the deployed contract Events. The Low level interactions section shows a call to the getValue function with parameters \_a=1 and \_b=2.

```

// 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;
    }
}

ContractDefinition eventExample 1 reference(s)

```

## Error Handling

Code:

```

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;
    }
}

```

Remix - Ethereum IDE

Not secure | remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13js&language=Solidity

Gmail YouTube Maps tf.keras.Model Te...

DEPLOY & RUN TRANSACTIONS

Deploy

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded

Deployed Contracts

ERRORHANDLING AT 0X90d7...B5B

checkinput

\_input: 243 call

0: string: Input is Uint8

Odd

\_input: 243 call

0: bool: true

Low level interactions

ErrorHandling.sol

```
pragma solidity ^0.5.0;

contract ErrorHandling {

    function checkInput(uint _input) public view returns(string memory){
        require(_input >= 0, "invalid uint8");
        require(_input <= 255, "invalid uint8");

        return "Input is Uint8";
    }

    function Odd(uint _input) public view returns(bool){
        require(_input % 2 != 0);
        return true;
    }
}
```

0 listen on all transactions Search with transaction hash or address

call [call] from: 0x5838Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.checkInput(uint256) data: 0x021...000F3 Debug

call to ErrorHandling.Odd

call [call] from: 0x5838Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.Odd(uint256) data: 0x922...000F3 Debug

**NATURAL  
LANGUAGE  
PROCESSING**

## INDEX

| Sr. No | Practicals   | Sign |
|--------|--|------|
| 1      | <ul style="list-style-type: none"> <li>a) Install NLTK.</li> <li>b) Convert the given text to speech.</li> <li>c) Convert audio file Speech to Text.</li> </ul>  |      |
| 2      | <ul style="list-style-type: none"> <li>a) Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories.</li> <li>b) Create and use your own corpora (plaintext, categorical)</li> <li>c) Study Conditional frequency distributions</li> <li>d) Study of tagged corpora with methods like tagged_sents, tagged_words.</li> <li>e) Write a program to find the most frequent noun tags.</li> <li>f) Map Words to Properties Using Python Dictionaries</li> <li>g) Study Default Tagger, Regular expression tagger, UnigramTagger</li> <li>h) Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find the score of words.</li> </ul> |      |
| 3      | <ul style="list-style-type: none"> <li>a) Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms</li> <li>b) Study lemmas, hyponyms, hypernyms.</li> <li>c) Write a program using python to find synonym and antonym of word "active" using Wordnet.</li> <li>d) Compare two nouns</li> <li>e) Handling stopword:</li> </ul>   |      |
| 4      | <ul style="list-style-type: none"> <li>a) Tokenization using Python's split() function code</li> <li>b) Tokenization using Regular Expressions (RegEx)</li> <li>c) Tokenization using NLTK</li> <li>d) Tokenization using the spaCy library</li> <li>e) Tokenization using Keras</li> <li>f) Tokenization using Gensim</li> </ul>  |      |
| 5      | <ul style="list-style-type: none"> <li>a) word tokenization in Hindi Source code</li> <li>b) Generate similar sentences from a given Hindi text input Source code</li> <li>c) Identify the Indian language of a text Source code</li> </ul>  |      |
| 6      | <ul style="list-style-type: none"> <li>a) Part of speech Tagging and chunking of user defined tex</li> <li>b) Named Entity recognition of user defined text.</li> </ul>  |      |

|    |   |  |
|----|---|--|
|    | c) Named Entity recognition with diagram using NLTK corpus – treebank   |  |
| 7  | <ul style="list-style-type: none"> <li>a) Define grammar using nltk. Analyze a sentence using the same</li> <li>b) Accept the input string with Regular expression of Finite Automaton</li> <li>c) Accept the input string with Regular expression of FA</li> <li>d) Implementation of Deductive Chart Parsing using context free grammar and a given sentence</li> </ul> |  |
| 8  | Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer Study WordNetLemmatize  |  |
| 9  | Implement Naive Bayes classifier.   |  |
| 10 | <ul style="list-style-type: none"> <li>a) Speech Tagging</li> <li>b) Statistical parsing</li> <li>c) Malt parsing</li> </ul>  |  |
| 11 | <ul style="list-style-type: none"> <li>a) Multiword Expressions in NLP</li> <li>b) Normalized Web Distance and Word Similarity Source</li> <li>c) Word Sense Disambiguation Source</li> </ul>   |  |

## Practical 1

### a) Install NLTK

#### Python 3.9.2 Installation on Windows

Step 1) Go to link <https://www.python.org/downloads/>, and select the latest version for windows.



Note: If you don't want to download the latest version, you can visit the download tab and see all releases.

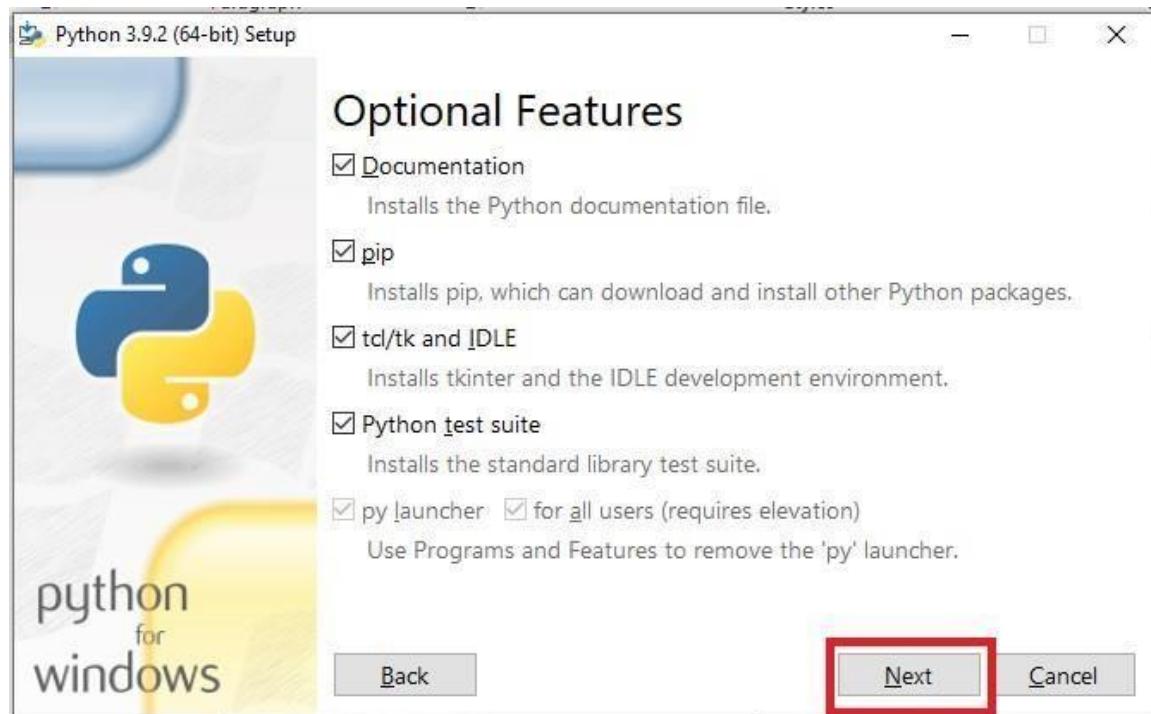
A screenshot of a web browser displaying the Python.org Downloads page for Python 3.9.2. The page has a light gray header with the Python logo and navigation links for About, Downloads, Documentation, Community, Success Stories, News, and Events. Below the header is a search bar with a magnifying glass icon and a 'GO' button. The main content area is titled 'Files' and contains a table of download options. A red arrow points to the 'Windows Installer (32-bit)' row, and another red arrow points to the 'Windows installer (64-bit)' row. The table columns are: Version, Operating System, Description, MD5 Sum, File Size, and GPG. The rows include: Gzipped source tarball (Source release), XZ compressed source tarball (Source release), macOS 64-bit intel installer (Mac OS X, for macOS 10.9 and later), macOS 64-bit universal2 installer (Mac OS X, for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)), Windows embeddable package (32-bit) (Windows), Windows embeddable package (64-bit) (Windows), Windows help file (Windows), Windows Installer (32-bit) (Windows), and Windows installer (64-bit) (Windows, Recommended). At the bottom of the page, there's a footer with links for About, Applications, Downloads, Documentation, Community, Success Stories, News, and Python News. A file icon followed by 'python-3.9.2-amd64.exe' is shown in the bottom left corner, along with a small up arrow icon.

Step 2) Click on the Windows installer (64 bit)

Step 3) Select Customize Installation

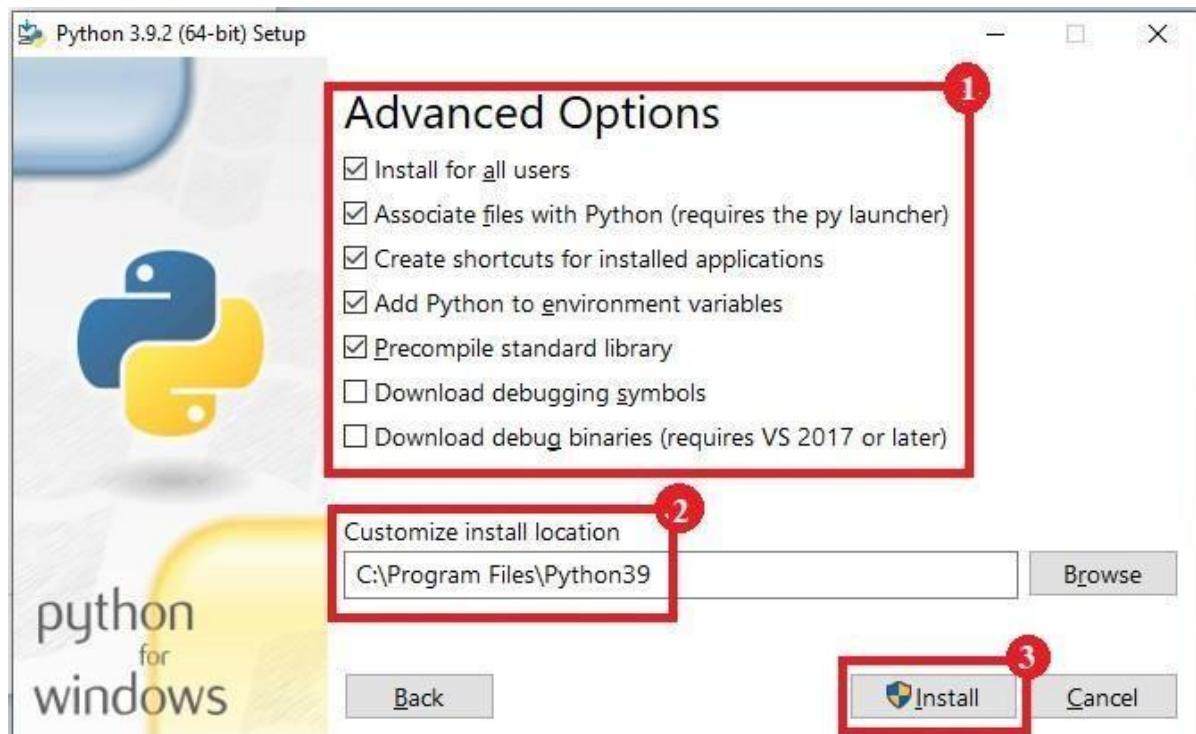


**Step 4)** Click NEXT



**Step 5)** In next screen

1. Select the advanced options
2. Give a Custom install location. Keep the default folder as c:\Program files\Python39
3. Click Install



**Step 6)** Click Close button once install is done.

**Step 7) open** command prompt window and run the following commands:

```
C:\Users\Beena Kapadia>pip install --upgrade pip
C:\Users\Beena Kapadia> pip install --user -U nltk
C:\Users\Beena Kapadia> >pip install --user -U numpy
C:\Users\Beena Kapadia>python
>>> import nltk
>>>
```

The screenshot shows a Command Prompt window titled 'Command Prompt - python'. It displays the output of several pip commands. The user installs nltk and numpy packages with the --user option. It also runs a python script that imports nltk. The output includes dependency resolution and warning messages about script locations.

```
C:\Users\Beena Kapadia>pip install --user -U nltk
Collecting nltk
  Using cached nltk-3.6.2-py3-none-any.whl (1.5 MB)
Requirement already satisfied: joblib in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk)
(1.0.1)
Requirement already satisfied: tqdm in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk)
(4.60.0)
Requirement already satisfied: regex in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk)
(2021.4.4)
Requirement already satisfied: click in c:\users\beena kapadia\appdata\roaming\python\python39\site-packages (from nltk)
(7.1.2)
Installing collected packages: nltk
  WARNING: The script nltk.exe is installed in 'C:\Users\Beena Kapadia\AppData\Roaming\Python\Python39\Scripts' which is
not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed nltk-3.6.2

C:\Users\Beena Kapadia>pip install --user -U numpy
Collecting numpy
  Using cached numpy-1.20.3-cp39-cp39-win_amd64.whl (13.7 MB)
Installing collected packages: numpy
  WARNING: The script f2py.exe is installed in 'C:\Users\Beena Kapadia\AppData\Roaming\Python\Python39\Scripts' which is
not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.20.3

C:\Users\Beena Kapadia>python
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>>
```

**b) Convert the given text to speech.**

**Source code:**

```
# text to speech
# pip install gtts
# pip install playsound

from playsound import playsound
# import required for text to speech conversion

from gtts import gTTS

mytext = "Welcome to Natural Language programming" language = "en"
myobj = gTTS(text=mytext, lang=language, slow=False) myobj.save("myfile.mp3")
playsound("myfile.mp3")
```

**Output:**

welcomeNLP.mp3 audio file is getting created and it plays the file with playsound() method, while running the program.

**c) Convert audio file Speech to Text. Source code:**

Note: required to store the input file "male.wav" in the current folder before running the program.

```
#pip3 install SpeechRecognition pydub

import speech_recognition as sr
filename = "male.wav"

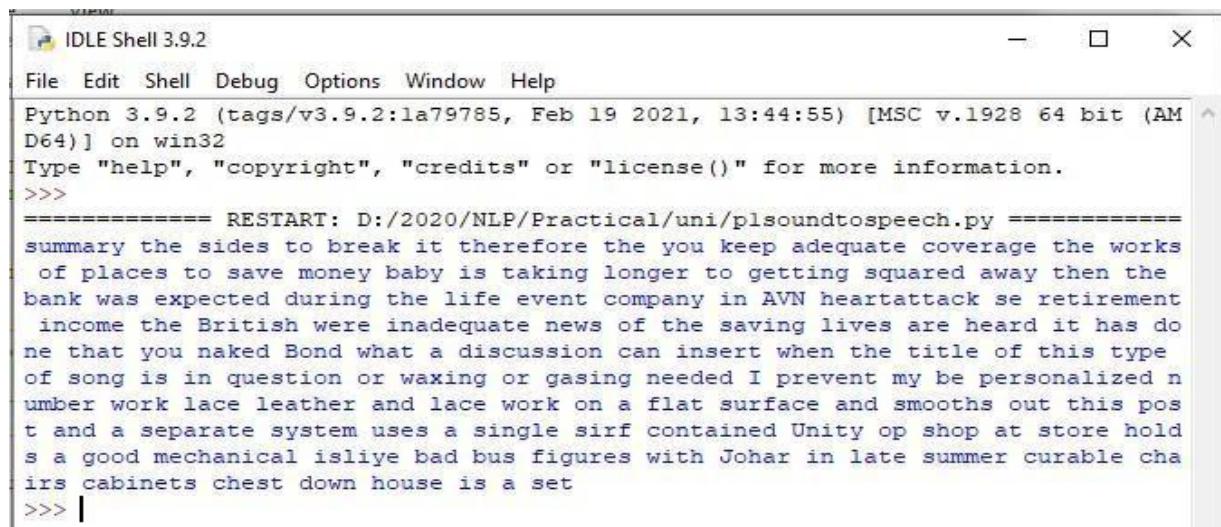
# initialize the recognizer
r = sr.Recognizer()

# open the file with sr.AudioFile(filename) as source:    # listen for the data (load audio to memory)
audio_data = r.record(source)
# recognize (convert from speech to text)    text = r.recognize_google(audio_data)
print(text)
```

Input:

male.wav (any wav file)

**Output:**



IDLE Shell 3.9.2

File Edit Shell Debug Options Window Help

```
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM  
D64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: D:/2020/NLP/Practical/uni/plsoundtospeech.py ======  
summary the sides to break it therefore the you keep adequate coverage the works  
of places to save money baby is taking longer to getting squared away then the  
bank was expected during the life event company in AVN heartattack se retirement  
income the British were inadequate news of the saving lives are heard it has do  
ne that you naked Bond what a discussion can insert when the title of this type  
of song is in question or waxing or gasing needed I prevent my be personalized n  
umber work lace leather and lace work on a flat surface and smooths out this pos  
t and a separate system uses a single sirf contained Unity op shop at store hold  
s a good mechanical isliye bad bus figures with Johar in late summer curable cha  
irs cabinets chest down house is a set  
>>> |
```

## Practical 2

a. **Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories, source code:**

"NLTK includes a small selection of texts from the Project brown electronic text archive, which contains some 25,000 free electronic books, hosted at <http://www.brown.org/>. We begin by getting the Python interpreter to load the NLTK package, then ask to see nltk.corpus.brown.fileids(), the file identifiers in this corpus:"

```
import nltk
from nltk.corpus import brown
print ('File ids of brown corpus\n',brown.fileids())
```

"Let's pick out the first of these texts — Emma by Jane Austen — and give it a short name, emma, then find out how many words it contains:" ca01 = brown.words('ca01')

```
# display first few words
print('\nca01 has following words:\n',ca01)
```

```
# total number of words in ca01
print('\nca01 has',len(ca01),'words')
```

```
#categories or files print ('\n\nCategories or file in brown corpus:\n') print
(brown.categories())
```

"display other information about each text, by looping over all the values of fileid corresponding to the brown file identifiers listed earlier and then computing statistics for each text."

```
print ('\n\nStatistics for each text:\n') print
('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFileName') for fileid in
brown.fileids():
    num_chars = len(brown.raw(fileid))    num_words = len(brown.words(fileid))    num_sents
= len(brown.sents(fileid))
    num_vocab = len(set([w.lower() for w in brown.words(fileid)]))
    print (int(num_chars/num_words),'\t\t\t', int(num_words/num_sents),'\t\t\t',
int(num_words/num_vocab),'\t\t\t', fileid)
```

**output:**

The screenshot shows the IDLE Shell 3.9.2 interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays Python code and its output. The code reads file IDs from a corpus and prints statistics for each text. A yellow box highlights the message "Squeezed text (50 lines.)".

```

File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/2020/NLP/Practical/uni/p2acorpus.py =====
File ids of brown corpus
Squeezed text (50 lines).

ca01 has following words:
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

ca01 has 2242 words

Categories or file in brown corpus:
['adventure', 'belles lettres', 'editorial', 'fiction', 'government', 'hobbies',
'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance',
'science_fiction']

Statistics for each text:

```

| AvgWordLen | AvgSentenceLen | no.ofTimesEachWordAppearsOnAvg | FileName |
|------------|----------------|--------------------------------|----------|
| 9          | 22             | 2                              | ca01     |
| 8          | 23             | 2                              | ca02     |
| 8          | 20             | 2                              | ca03     |
| 9          | 25             | 2                              | ca04     |
| 8          | 26             | 3                              | ca05     |

### b. Create and use your own corpora (plaintext, categorical) source code:

"NLTK includes a small selection of texts from the Project filelist electronic text archive, which contains some 25,000 free electronic books, hosted at <http://www.filelist.org/>. We begin by getting the Python interpreter to load the NLTK package, then ask to see `nltk.corpus.filelist.fileids()`, the file identifiers in this corpus:""

```
import nltk from nltk.corpus import PlaintextCorpusReader
```

```
corpus_root = 'D:/2020/NLP/Practical/uni' filelist = PlaintextCorpusReader(corpus_root, '*')
print ('\n File list: \n')
```

```

print (filelist.fileids())

print (filelist.root)

"""display other information about each text, by looping over all the values of fileid
corresponding to the filelist file identifiers listed earlier and then computing statistics for each
text."""
print ('\n\nStatistics for each text:\n') print
('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\tFileName')
for fileid in filelist.fileids():
    num_chars = len(filelist.raw(fileid))    num_words = len(filelist.words(fileid))    num_sents
    = len(filelist.sents(fileid))
    num_vocab = len(set([w.lower() for w in filelist.words(fileid)]))    print
    (int(num_chars/num_words),'\t\t\t', int(num_words/num_sents),'\t\t\t',
    int(num_words/num_vocab),'\t\t\t', fileid)

```

**output:**

```

IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)]
] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/2020/NLP/Practical/uni/p2b_ownCorpus.py =====

File list:

['TTS.py', 'male.txt', 'plsoundtospeech.py', 'p2acorpus.py', 'p2b_ownCorpus.py']
D:\2020\NLP\Practical\uni

Statistics for each text:

      AvgWordLen      AvgSentenceLen  no.ofTimesEachWordAppearsOnAvg  FileName
4                  14                      2              TTS.py
5                 140                     1            male.txt
5                  20                      1  plsoundtospeech.py
4                  38                      2        p2acorpus.py
4                  33                      2  p2b_ownCorpus.py
>>> |

```

**c. Study Conditional frequency**

**distributions source code:**

```

#process a sequence of pairs
text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...] pairs = [('news', 'The'), ('news',
'Fulton'), ('news', 'County'), ...] import nltk
from nltk.corpus import brown fd = nltk.ConditionalFreqDist(
    (genre, word)
    for genre in brown.categories()
    for word in brown.words(categories=genre))

```

```

genre_word = [(genre, word)
    for genre in ['news', 'romance']
    for word in brown.words(categories=genre)]

print(len(genre_word))

print(genre_word[:4])

print(genre_word[-4:])

cfд = nltk.ConditionalFreqDist(genre_word)

print(cfд)

print(cfд.conditions())

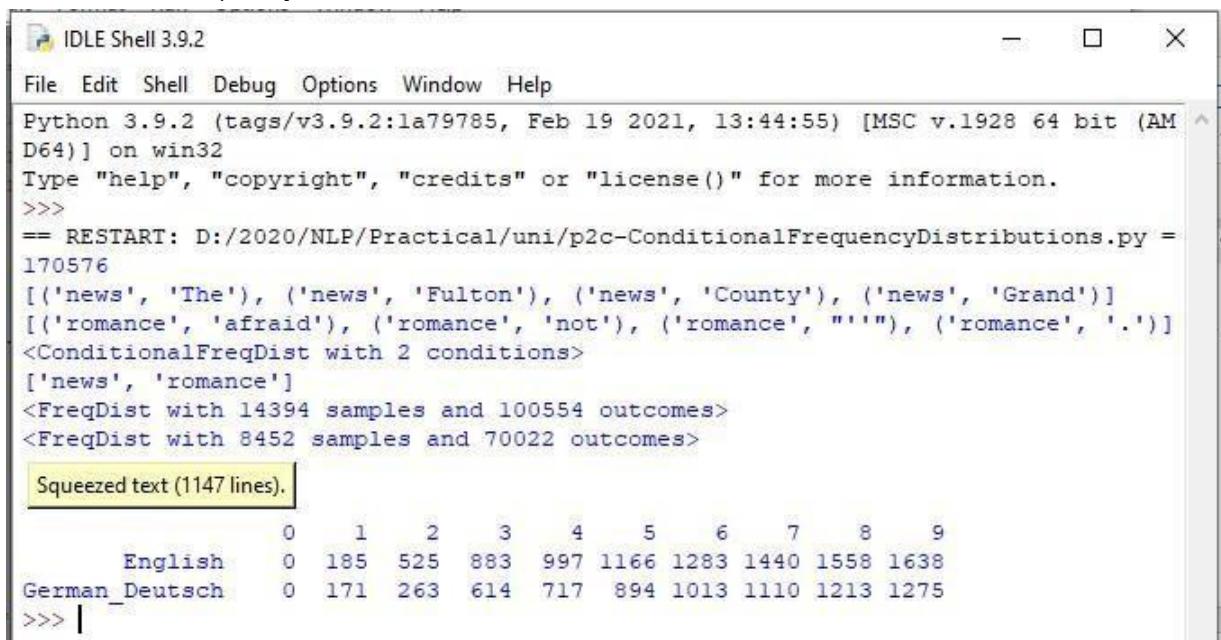
print(cfд['news']) print(cfд['romance'])
print(list(cfд['romance']))

from nltk.corpus import inaugural
cfд = nltk.ConditionalFreqDist(
    (target, fileid[:4])      for fileid in inaugural.fileids()      for w in
    inaugural.words(fileid)   for target in ['america', 'citizen']
    if w.lower().startswith(target))

from nltk.corpus import udhr languages = ['Chickasaw', 'English', 'German_Deutsch',
'Greenlandic_Inuktikut', 'Hungarian_Magyar', 'Ibibio_Efik'] cfд =
nltk.ConditionalFreqDist( (lang, len(word))      for lang in languages
    for word in udhr.words(lang + '-Latin1'))

cfд.tabulate(conditions=['English', 'German_Deutsch'],           samples=range(10),
cumulative=True) output:

```



The screenshot shows the Python 3.9.2 shell interface. The title bar reads "IDLE Shell 3.9.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the script's code and its execution output.

```

File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: D:/2020/NLP/Practical/uni/p2c-ConditionalFrequencyDistributions.py =
170576
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]
[('romance', 'afraid'), ('romance', 'not'), ('romance', "'"), ('romance', '.')]
<ConditionalFreqDist with 2 conditions>
['news', 'romance']
<FreqDist with 14394 samples and 100554 outcomes>
<FreqDist with 8452 samples and 70022 outcomes>
Squeezed text (1147 lines).
          0   1   2   3   4   5   6   7   8   9
English    0  185  525  883  997 1166 1283 1440 1558 1638
German_Deutsch  0  171  263  614  717  894 1013 1110 1213 1275
>>> |

```

**d. Study of tagged corpora with methods like tagged\_sents, tagged\_words.**

**Source code:** import nltk  
from nltk import tokenize  
nltk.download('punkt')  
nltk.download('words')

```
para = "Hello! My name is Beena Kapadia. Today you'll be learning NLTK." sents =  
tokenize.sent_tokenize(para)  
print("\nsentence tokenization\n=====\\n",sents)
```

```
# word tokenization print("\nword tokenization\\n=====\\n") for index in  
range(len(sents)): words = tokenize.word_tokenize(sents[index]) print(words)
```

**output:**

```
IDLE Shell 3.9.2  
File Edit Shell Debug Options Window Help  
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM  
D64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: D:/2020/NLP/Practical/uni/p2dTagging.py ======  
[nltk_data] Downloading package punkt to C:\Users\Beena  
[nltk_data]     Kapadia\AppData\Roaming\nltk_data...  
[nltk_data] Package punkt is already up-to-date!  
[nltk_data] Downloading package words to C:\Users\Beena  
[nltk_data]     Kapadia\AppData\Roaming\nltk_data...  
[nltk_data] Package words is already up-to-date!  
  
sentence tokenization  
======[{'Hello!', 'My name is Beena Kapadia.', "Today you'll be learning NLTK."}]  
  
word tokenization  
======[['Hello', '!'], ['My', 'name', 'is', 'Beena', 'Kapadia', '.'], ['Today', 'you', "'ll", 'be', 'learning', 'NLTK', '.']]>>> |
```

**e. Write a program to find the most frequent noun**

**tags. Code:**

```
import nltk  
from collections import defaultdict  
text = nltk.word_tokenize("Nick likes to play football. Nick does not like to play cricket.")  
tagged = nltk.pos_tag(text) print(tagged)  
  
# checking if it is a noun or not addNounWords = [] count=0 for words in tagged:  
    val = tagged[count][1]    if(val == 'NN' or val == 'NNS' or val == 'NNPS' or val == 'NNP'):  
        addNounWords.append(tagged[count][0])    count+=1  
  
print (addNounWords)
```

```

temp = defaultdict(int)
# memoizing count for sub in addNounWords:    for wrd in sub.split():      temp[wrd] +=

1 # getting max frequency

res = max(temp, key=temp.get)

# printing result
print("Word with maximum frequency : " + str(res))
output:
=====
[('Nick', 'NNP'), ('likes', 'VBZ'), ('to', 'TO'), ('play', 'VB'), ('football', 'NN'), ('.', '.'), ('Nick', 'NNP'), ('does', 'VBZ'), ('not', 'RB'), ('like', 'VB'), ('to', 'TO'), ('play', 'VB'), ('cricket', 'NN'), ('.', '.')]
['Nick', 'football', 'Nick', 'cricket']
Word with maximum frequency : Nick
>>>

```

#### f. Map Words to Properties Using Python Dictionaries code:

```

#creating and printing a dictionay by mapping word with its properties thisdict = { "brand": "Ford",
"model": "Mustang",
"year": 1964
}
print(thisdict) print(thisdict["brand"]) print(len(thisdict))
print(type(thisdict))

```

#### output:

```

=====
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Ford
3
<class 'dict'>

```

#### g. Study i) DefaultTagger, ii) Regular expression tagger, iii) UnigramTagger

##### i) DefaultTagger code:

```

import nltk from nltk.tag import DefaultTagger exptagger = DefaultTagger('NN') from
nltk.corpus import treebank testsentences = treebank.tagged_sents() [1000:]
print(exptagger.evaluate (testsentences))

```

```

#Tagging a list of sentences import nltk from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
print(exptagger.tag_sents([('Hi', ','), ('How', 'are', 'you', '?')]))

```

#### output

```

=====
0.13198749536374715
[[('Hi', 'NN'), (',', 'NN')], [('How', 'NN'), ('are', 'NN'), ('you', 'NN'), ('?', 'NN')]]
>>>

```

##### ii) Regular expression tagger, code: from nltk.corpus import brown from nltk.tag import RegexpTagger test\_sent = brown.sents(categories='news')[0]

```

regexp_tagger = RegexpTagger(
    [(r'^-[0-9]+([.-][0-9]+)?$', 'CD'), # cardinal numbers
     (r'(The|the|A|a|An|an)$', 'AT'), # articles
     (r'.*able$', 'JJ'), # adjectives
     (r'.*ness$', 'NN'), # nouns formed from adjectives
     (r'.*ly$', 'RB'), # adverbs
     (r'.*s$', 'NNS'), # plural nouns
     (r'.*ing$', 'VBG'), # gerunds
     (r'.*ed$', 'VBD'), # past tense verbs
     (r'.*', 'NN') # nouns (default)
])

```

```
print(regexp_tagger) print(regexp_tagger.tag(test_sent)) output:
```

```
=====
RESTART: D:/2020/NLP/Practical/uni/p2g2RegularExp.py =====
<Regexp Tagger: size=9>
[('The', 'AT'), ('Fulton', 'NN'), ('County', 'NN'), ('Grand', 'NN'), ('Jury', 'NN'), ('said', 'NN'), ('Friday', 'NN'), ('an', 'AT'), ('investigation', 'NN'), ('of', 'NN'), ('Atlanta's', 'NNS'), ('recent', 'NN'), ('primary', 'NN'), ('electio n', 'NN'), ('produced', 'VBD'), ('``', 'NN'), ('no', 'NN'), ('evidence', 'NN'), (''''', 'NN'), ('that', 'NN'), ('any', 'NN'), ('irregularities', 'NNS'), ('took', 'NN'), ('place', 'NN'), ('.', 'NN')]
```

### iii) UnigramTagger code:

```
# Loading Libraries from nltk.tag import UnigramTagger
from nltk.corpus import treebank
```

```
# Training using first 10 tagged sentences of the treebank corpus as data.
```

```
# Using data
```

```
train_sents = treebank.tagged_sents()[:10]
```

```
# Initializing
```

```
tagger = UnigramTagger(train_sents)
```

```
# Lets see the first sentence # (of the treebank corpus) as list
```

```
print(treebank.sents()[0]) print('\n',tagger.tag(treebank.sents()[0]))
```

```
#Finding the tagged results after training.
```

```
tagger.tag(treebank.sents()[0])
```

```
#Overriding the context model
```

```
tagger = UnigramTagger(model ={'Pierre': 'NN'})
```

```
print('\n',tagger.tag(treebank.sents()[0])) output:
```

```
=====
RESTART: D:/2020/NLP/Practical/uni/p2g3Unigram.py =====
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'boa rd', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']
```

```
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'), (',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT'), ('board', 'NN'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive', 'JJ'), ('director', 'NN'), ('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')]
```

```
[('Pierre', 'NN'), ('Vinken', None), (',', None), ('61', None), ('years', None), ('old', None), (',', None), ('will', None), ('join', None), ('the', None), ('b oard', None), ('as', None), ('a', None), ('nonexecutive', None), ('director', No ne), ('Nov.', None), ('29', None), ('.', None)]
```

**h. Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find the score of words.**

**Question:**

Initialize the hash tag test data or URL test data and convert to plain text without any space.. Read a text file of different words and compare the plain text data with the words exist in that text file and find out different words available in that plain text. Also find out how many words could be found. (for example, text = "#whatismyname" or text = [www.whatismyname.com](http://www.whatismyname.com)).

Convert that to plain text without space as:

whatismyname and read text file as words.txt. Now compare plain text with words given in a file and find the words form the plain text and the count of words which could be found)

**Source code:** from \_\_future\_\_ import with\_statement #with statement for reading file import re # Regular expression

```
words = [] # corpus file words testword = [] # test words      ans = [] # words matches
with corpus

print("MENU")
print("      ") print(" 1 . Hash tag segmentation ") print(" 2 . URL segmentation ")
print("enter the input choice for performing word segmentation") choice = int(input())

if choice == 1:   text = "#whatismyname"      # hash tag test data to segment
print("input with HashTag",text)                  pattern=re.compile("[^\w']")      a =
pattern.sub("", text) elif choice == 2:
    text = "www.whatismyname.com"      # url test data to segment   print("input with
URL",text)   a=re.split("\s|(?<!\d)[.,](?!\d)", text)
    splitwords = ["www","com","in"]      # remove the words which is containg in the list   a
=""".join([each for each in a if each not in splitwords]) else:   print("wrong choice try again")
print(a)

for each in a:   testword.append(each) #test word
test_lenth = len(testword)      # lenth of the test data

# Reading the corpus with open('words.txt', 'r') as f:   lines = f.readlines()
words =[ (e.strip()) for e in lines]

def Seg(a,lenth):
    ans =[]   for k in range(0,lenth+1): # this loop checks char by char in the corpus

        if a[0:k] in words:
            print(a[0:k],"-appears in the corpus")      ans.append(a[0:k])
            break   if ans != []:
                g = max(ans,key=len)
                return g

test_tot_itr = 0 #each iteration value
answer = [] # Store the each word contains the corpus
Score = 0 # initial value for score
N = 37 # total no of corpus
M = 0 C = 0 while test_tot_itr < test_lenth:      ans_words = Seg(a,test_lenth)   if
ans_words != 0:
    test_itr = len(ans_words)      answer.append(ans_words)      a = a[test_itr:test_lenth]
    test_tot_itr += test_itr
```

```
Aft_Seg = " ".join([each for each in answer]) # print segmented words in the list
print("output") print("      ")
print(Aft_Seg) # print After segmentation the input

# Calculating Score C = len(answer)
score = C * N / N      # Calculate the
score print("Score",score)
```

**Input:**

**Words.txt**

---

```
check domain big rocks name cheap being human back
current rates                               social
ought to                                     media
go                                         30
down apple domains honesty hour follow    seconds
                                          earth
                                          this is
                                          insane it
                                          time
                                          what is
                                          my
                                          name
                                          let us
                                          go
```

**Output:**

```
IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/2020/NLP/Practical/uni/p2hWord.py =====
MENU
-----
1 . Hash tag segmentation
2 . URL segmentation
enter the input choice for performing word segmentation
1
input with HashTag #whatismyname
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
-----
what is my name
Score 4.0
>>>
===== RESTART: D:/2020/NLP/Practical/uni/p2hWord.py =====
MENU
-----
1 . Hash tag segmentation
2 . URL segmentation
enter the input choice for performing word segmentation
2
input with URL www.whatismyname.com
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
-----
what is my name
Score 4.0
>>> |
```

## Practical 3

### a. Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms

#### Source code:

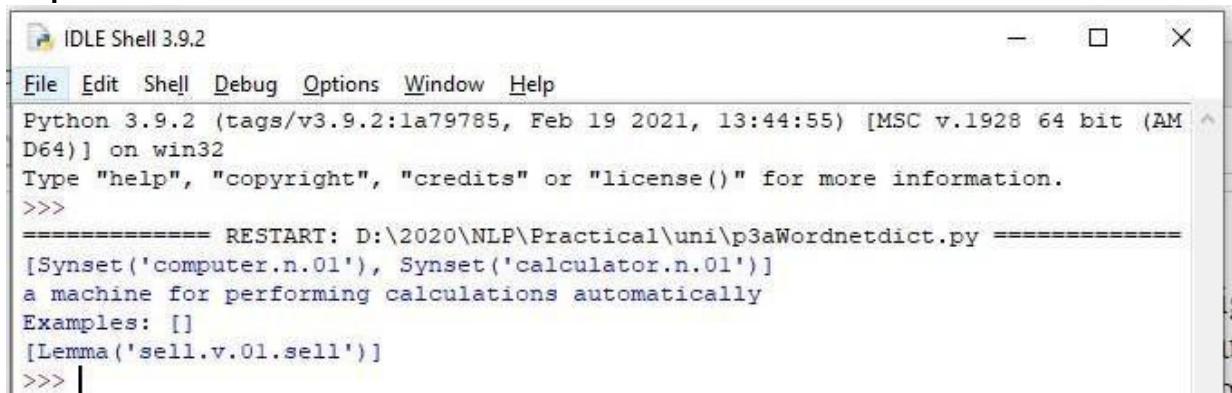
```
'''WordNet provides synsets which is the collection of synonym words also called  
"lemmas''' import nltk from nltk.corpus import wordnet  
print(wordnet.synsets("computer"))
```

```
# definition and example of the word 'computer'  
print(wordnet.synset("computer.n.01").definition())
```

```
#examples  
print("Examples:", wordnet.synset("computer.n.01").examples())
```

```
#get Antonyms  
print(wordnet.lemma('buy.v.01.buy').antonyms())
```

#### output:



```
IDLE Shell 3.9.2  
File Edit Shell Debug Options Window Help  
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM  
D64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: D:\2020\NLP\Practical\uni\p3aWordnetdict.py ======  
[Synset('computer.n.01'), Synset('calculator.n.01')]  
a machine for performing calculations automatically  
Examples: []  
[Lemma('sell.v.01.sell')]  
>>> |
```

### b. Study lemmas, hyponyms, hypernyms.

#### Source code:

```
import nltk from nltk.corpus import wordnet print(wordnet.synsets("computer"))  
print(wordnet.synset("computer.n.01").lemma_names()) #all lemmas for each synset.  
for e in wordnet.synsets("computer"): print(f'{e} --> {e.lemma_names()}')
```

```
#print all lemmas for a given synset  
print(wordnet.synset('computer.n.01').lemmas())
```

```
#get the synset corresponding to lemma  
print(wordnet.lemma('computer.n.01.computing_device').synset())
```

```
#Get the name of the lemma  
print(wordnet.lemma('computer.n.01.computing_device').name())
```

```
#Hyponyms give abstract concepts of the word that are much more specific #the list of  
hyponyms words of the computer
```

```

syn = wordnet.synset('computer.n.01') print(syn.hyponyms)

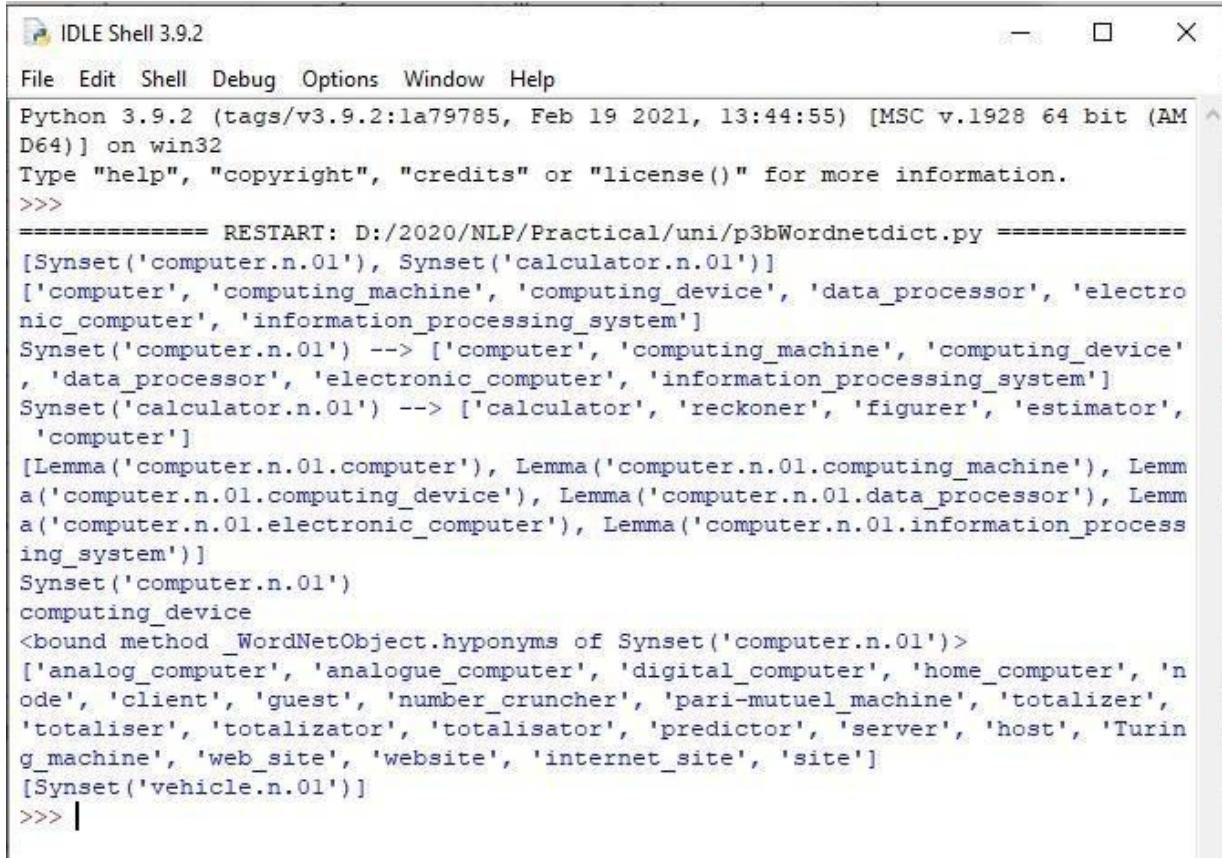
print([lemma.name() for synset in syn.hyponyms() for lemma in synset.lemmas()])

#the semantic similarity in WordNet vehicle = wordnet.synset('vehicle.n.01') car =
wordnet.synset('car.n.01')

print(car.lowest_common_hypernyms(vehicle))

```

**Output:**



The screenshot shows the Python 3.9.2 IDLE Shell interface. The title bar reads "IDLE Shell 3.9.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following Python code and its output:

```

File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/2020/NLP/Practical/uni/p3bWordnetdict.py =====
[Synset('computer.n.01'), Synset('calculator.n.01')]
['computer', 'computing_machine', 'computing_device', 'data_processor', 'electro
nic_computer', 'information_processing_system']
Synset('computer.n.01') --> ['computer', 'computing_machine', 'computing_device'
, 'data_processor', 'electronic_computer', 'information_processing_system']
Synset('calculator.n.01') --> ['calculator', 'reckoner', 'figurer', 'estimator',
'computer']
[Lemma('computer.n.01.computer'), Lemma('computer.n.01.computing_machine'), Lemm
a('computer.n.01.computing_device'), Lemma('computer.n.01.data_processor'), Lemm
a('computer.n.01.electronic_computer'), Lemma('computer.n.01.information_process
ing_system')]
Synset('computer.n.01')
computing_device
<bound method _WordNetObject.hyponyms of Synset('computer.n.01')>
['analog_computer', 'analogue_computer', 'digital_computer', 'home_computer', 'n
ode', 'client', 'guest', 'number_cruncher', 'pari-mutuel_machine', 'totalizer',
'totaliser', 'totalizator', 'totalisator', 'predictor', 'server', 'host', 'Turin
g_machine', 'web_site', 'website', 'internet_site', 'site']
[Synset('vehicle.n.01')]
>>> |

```

- c. Write a program using python to find synonym and antonym of word "active" using Wordnet.

**Source code:** from nltk.corpus import wordnet print( wordnet.synsets("active"))

```
print(wordnet.lemma('active.a.01.active').antonyms())
```

**Output:**

```

IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: D:/2020/NLP/Practical/uni/p3cWordnetdict.py =====
[Synset('active_agent.n.01'), Synset('active_voice.n.01'), Synset('active.n.03')
, Synset('active.a.01'), Synset('active.s.02'), Synset('active.a.03'), Synset('a
ctive.s.04'), Synset('active.a.05'), Synset('active.a.06'), Synset('active.a.07'
), Synset('active.s.08'), Synset('active.a.09'), Synset('active.a.10'), Synset('
active.a.11'), Synset('active.a.12'), Synset('active.a.13'), Synset('active.a.14
')]
[Lemma('inactive.a.02.inactive')]

```

**d. Compare two nouns source code:**

```

import nltk
from nltk.corpus import wordnet

syn1 = wordnet.synsets('football')
syn2 = wordnet.synsets('soccer')

# A word may have multiple synsets, so need to compare each synset of word1 with synset of
word2
for s1 in syn1:  for s2 in syn2:      print("Path similarity of: ")      print(s1, '(', s1.pos(), ')',
'[', s1.definition(), ']')      print(s2, '(', s2.pos(), ')', '[', s2.definition(), ']')
print(" is", s1.path_similarity(s2))      print()

```

**output:**

```

IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: D:/2020/NLP/Practical/uni/p3dcompareNouns.py =====
Path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with a ball (round o
r oval) in which two teams try to kick or carry or propel the ball into each oth
er's goal ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players t
ry to kick or head a ball into the opponents' goal ]
    is 0.5

Path similarity of:
Synset('football.n.02') ( n ) [ the inflated oblong ball used in playing America
n football ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players t
ry to kick or head a ball into the opponents' goal ]
    is 0.05

```

**e. Handling stopword:**

**i) Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List**

**code:**

```

import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize

```

```

text = "Yashesh likes to play football, however he is not too fond of tennis." text_tokens =
word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]

print(tokens_without_sw)

#add the word play to the NLTK stop word collection all_stopwords =
stopwords.words('english') all_stopwords.append('play')

text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]

print(tokens_without_sw)

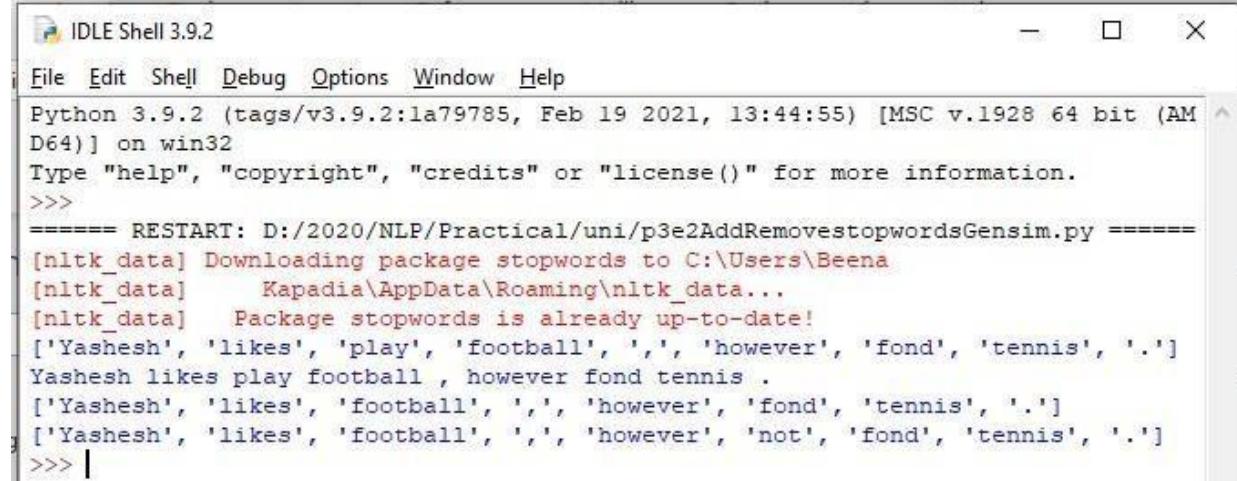
#remove 'not' from stop word collection all_stopwords.remove('not')

text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]

print(tokens_without_sw)

```

### **output**



```

IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/2020/NLP/Practical/uni/p3e2AddRemovestopwordsGensim.py =====
[nltk_data] Downloading package stopwords to C:\Users\Beena
[nltk_data]   Kapadia\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['Yashesh', 'likes', 'play', 'football', ',', 'however', 'fond', 'tennis', '.']
Yashesh likes play football , however fond tennis .
['Yashesh', 'likes', 'football', ',', 'however', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'however', 'not', 'fond', 'tennis', '.']
>>> |

```

### **ii) Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List code:**

```
#pip install gensim import gensim
from gensim.parsing.preprocessing import remove_stopwords
```

```

text = "Yashesh likes to play football, however he is not too fond of tennis." filtered_sentence
= remove_stopwords(text)

print(filtered_sentence)

all_stopwords = gensim.parsing.preprocessing.STOPWORDS print(all_stopwords)

```

```
'''The following script adds likes and play to the list of stop words in Gensim:'''

from gensim.parsing.preprocessing import STOPWORDS

all_stopwords_gensim = STOPWORDS.union(set(['likes', 'play']))

text = "Yashesh likes to play football, however he is not too fond of tennis." text_tokens =
word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords_gensim]

print(tokens_without_sw)
```

'''Output:

```
['Yashesh', 'football', ',', 'fond', 'tennis', '.']
```

The following script removes the word "not" from the set of stop words in Gensim:'''

```
from gensim.parsing.preprocessing import STOPWORDS
all_stopwords_gensim = STOPWORDS
sw_list = {"not"}
all_stopwords_gensim = STOPWORDS.difference(sw_list)

text = "Yashesh likes to play football, however he is not too fond of tennis." text_tokens =
word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords_gensim]

print(tokens_without_sw)
```

#### **output**

Microsoft Visual C++ 14.0 is required. Get it with "Build Tools for Visual Studio":

<https://visualstudio.microsoft.com/downloads/>

### **iii) Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List**

**code:**

```
#pip install spacy
#python -m spacy download en_core_web_sm
#python -m spacy download en

import spacy import nltk
from nltk.tokenize import word_tokenize

sp = spacy.load('en_core_web_sm')

#add the word play to the NLTK stop word collection
all_stopwords =
sp.Defaults.stop_words
all_stopwords.add("play")

text = "Yashesh likes to play football, however he is not too fond of tennis." text_tokens =
word_tokenize(text)
```

```
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
```

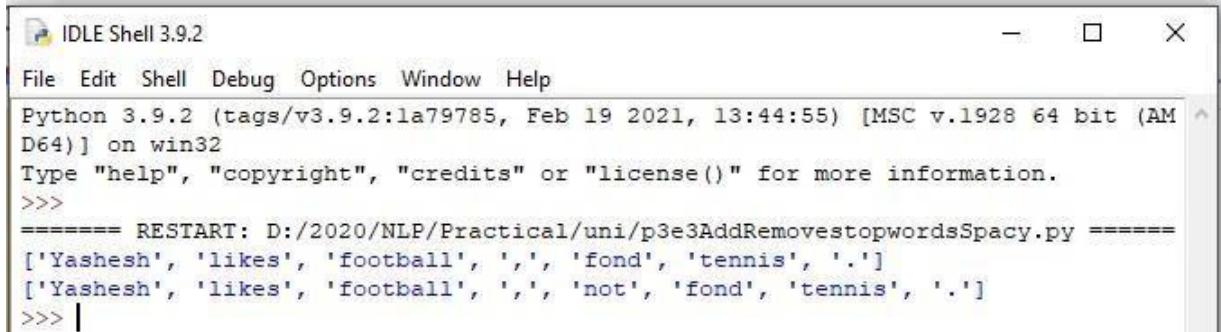
```
print(tokens_without_sw)
```

```
#remove 'not' from stop word collection all_stopwords.remove('not')
```

```
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
```

```
print(tokens_without_sw)
```

**output:**



The screenshot shows the Python IDLE Shell window. The title bar reads "IDLE Shell 3.9.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/2020/NLP/Practical/uni/p3e3AddRemovestopwordsSpacy.py =====
['Yashesh', 'likes', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'not', 'fond', 'tennis', '.']
>>> |
```

## Practical 4

### a. Tokenization using Python's split() function code:

text = """ This tool is an a beta stage. Alexa developers can use Get Metrics API to seamlessly analyse metric. It also supports custom skill model, prebuilt Flash Briefing model, and the Smart Home Skill API. You can use this tool for creation of monitors, alarms, and dashboards that spotlight changes. The release of these three tools will enable developers to create visual rich skills for Alexa devices with screens. Amazon describes these tools as the collection of tech and tools for creating visually rich and interactive voice experiences. """

```
data = text.split('.') for i in data:    print (i)
```

#### output:

```
>>>
=====
RESTART: D:/2020/NLP/Practical/uni/p4a.py =====
This tool is an a beta stage
Alexa developers can use Get Metrics API to seamlessly analyse metric
It also supports custom skill model, prebuilt Flash Briefing model, and the Smart Home Skill API
You can use this tool for creation of monitors, alarms, and dashboards that spotlight changes
The release of these three tools will enable developers to create visual rich skills for Alexa devices with screens
Amazon describes these tools as the collection of tech and tools for creating visually rich and interactive voice experiences
```

### b. Tokenization using Regular Expressions (RegEx) code:

```
import nltk
# import RegexpTokenizer() method from nltk
from nltk.tokenize import RegexpTokenizer

# Create a reference variable for Class RegexpTokenizer tk = RegexpTokenizer('\s+', gaps = True)

# Create a string input
str = "I love to study Natural Language Processing in Python"

# Use tokenize method
tokens = tk.tokenize(str)

print(tokens)
```

#### output:

```
>>>
=====
RESTART: D:/2020/NLP/Practical/uni/p4b.py =====
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
>>> |
```

### c. Tokenization using NLTK

**code:**

```
import nltk
from nltk.tokenize import word_tokenize

# Create a string input
str = "I love to study Natural Language Processing in Python"

# Use tokenize method
print(word_tokenize(str))
```

**output:**

```
=====
RESTART: D:/2020/NLP/Practical/uni/p4c.py =====
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
>>>
```

### d. Tokenization using the spaCy library

**code:** import spacy

```
nlp = spacy.blank("en")
```

```
# Create a string input
str = "I love to study Natural Language Processing in Python"
```

```
# Create an instance of document;
# doc object is a container for a sequence of Token objects. doc = nlp(str)
```

```
# Read the words; Print the words
# words = [word.text for word in doc] print(words)
```

**output:**

```
=====
RESTART: D:/2020/NLP/Practical/uni/p4d.py =====
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
>>>
```

### e. Tokenization using

**Keras code:**

```
#pip install keras #pip install tensorflow
from keras.preprocessing.text import text_to_word_sequence
```

```
# Create a string input
str = "I love to study Natural Language Processing in Python"
```

```
# tokenizing the text
tokens = text_to_word_sequence(str)
print(tokens)
```

**output:**

```
>>>
=====
RESTART: D:\2020\NLP\Practical\uni\p4e.py =====
['i', 'love', 'to', 'study', 'natural', 'language', 'processing', 'in', 'python']
]
```

## f. Tokenization using Gensim

### code:

```
#pip install gensim
from gensim.utils import tokenize

# Create a string input

str = "I love to study Natural Language Processing in Python"

# tokenizing the text
list(tokenize(str))
```

### output:

Microsoft Visual C++ 14.0 is required. Get it with "Build Tools for Visual Studio":  
<https://visualstudio.microsoft.com/downloads/>

## Practical 5

**Import NLP Libraries for Indian Languages and perform:** Note: Execute this practical in  
<https://colab.research.google.com/>

**a) word tokenization in Hindi Source code:**

```
!pip install torch==1.3.1+cpu -f https://download.pytorch.org/whl/torch\_stable.html
```

```
!pip install inltk
```

```
!pip install tornado==4.5.3
```

```
from inltk.inltk import setup  
setup('hi')
```

```
from inltk.inltk import tokenize
```

```
hindi_text = """प्राकृ तिक भाषा सीखना बहुत तिलचस्प है।"""
```

```
# tokenize(input text, language code) tokenize(hindi_text, "hi")
```

**output**

```
['प्राकृ तिक', 'भाषा', 'सीखना', 'बहुत', 'तिलचस्प', 'है', '!']
```

**b) Generate similar sentences from a given Hindi text input Source code:**

```
!pip install torch==1.3.1+cpu -f https://download.pytorch.org/whl/torch\_stable.html
```

```
!pip install inltk
```

```
!pip install tornado==4.5.3
```

```
from inltk.inltk import setup  
setup('hi')
```

```
from inltk.inltk import get_similar_sentences
```

```
# get similar sentences to the one given in hindi  
output = get_similar_sentences('मैं आज बहुत खुश हूं', 5, 'hi')
```

```
print(output)
```

**Output:**

```
['मैं आजकल बहुत खुश हूं', 'मैं आज अत्यतिक खुश हूं', 'मैं अभी बहुत खुश हूं', 'मैं फिर मैं बहुत खुश हूं', 'मैं फिर मैं बहुत खुश हूं']
```

**c) Identify the Indian language of a text Source code:**

```
!pip install torch==1.3.1+cpu -f https://download.pytorch.org/whl/torch\_stable.html
```

```
!pip install inltk
```

```
!pip install tornado==4.5.3

from inltk.inltk import setup
setup('gu')

from inltk.inltk import identify_language #Identify the language of given text
identify_language('બીજા કાપડીયા')
```

**Output:** gujarati

## Practical 6

- a. Part of speech Tagging and chunking of user defined text.
- b. Named Entity recognition of user defined text.
- c. Named Entity recognition with diagram using NLTK corpus – treebank

**POS Tagging, chunking and NER:**

a) sentence tokenization, word tokenization, Part of speech Tagging and chunking of user defined text. Source code: import nltk from nltk import tokenize  
nltk.download('punkt') from nltk import tag from nltk import chunk  
nltk.download('averaged\_perceptron\_tagger') nltk.download('maxent\_ne\_chunker')  
nltk.download('words')

```
para = "Hello! My name is Beena Kapadia. Today you'll be learning NLTK." sents =  
tokenize.sent_tokenize(para)  
print("\nsentence tokenization\n=====\\n",sents)
```

```
# word tokenization print("\nword tokenization\\n=====\\n") for index in  
range(len(sents)): words = tokenize.word_tokenize(sents[index]) print(words)
```

```
# POS Tagging
```

```
tagged_words = [] for index in range(len(sents)):  
tagged_words.append(tag.pos_tag(words))  
print("\nPOS Tagging\\n=====\\n",tagged_words)
```

```
# chunking
```

```
tree = [] for index in range(len(sents)):  
tree.append(chunk.ne_chunk(tagged_words[index])) print("\\nchunking\\n=====\\n")  
print(tree)
```

**Output:**

sentence tokenization

=====

['Hello!', 'My name is Beena Kapadia.', "Today you'll be learning NLTK."]

word tokenization

=====

```
['Hello', '!']  
['My', 'name', 'is', 'Beena', 'Kapadia', '.']  
['Today', 'you', "'ll", 'be', 'learning', 'NLTK', '.']
```

POS Tagging

=====

```
[[('Today', 'NN'), ('you', 'PRP'), ("\"I\"", 'MD'), ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('you', 'PRP'), ("\"I\"", 'MD'), ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('you', 'PRP'), ("\"I\"", 'MD'), ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')]]
```

chunking

=====

```
[Tree('S', [('Today', 'NN'), ('you', 'PRP'), ("\"I\"", 'MD'), ('be', 'VB'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')]), Tree('S', [('Today', 'NN'), ('you', 'PRP'), ("\"I\"", 'MD'), ('be', 'VB'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')]), Tree('S', [('Today', 'NN'), ('you', 'PRP'), ("\"I\"", 'MD'), ('be', 'VB'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')])]
```

**b) Named Entity recognition using user defined**

**text. Source code:**

```
!pip install -U spacy
```

```
!python -m spacy download en_core_web_sm
```

```
import spacy
```

```
# Load English tokenizer, tagger, parser and NER nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving
cars at "      "Google in 2007, few people outside of the company took him "
"seriously. "I can tell you very senior CEOs of major American "
"car companies would shake my hand and turn away because I wasn't "
"worth talking to," said Thrun, in an interview with Recode earlier "
>this week.")
doc = nlp(text)
```

```
# Analyse syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])
```

**Output:**

```
Noun phrases: ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people', 'the company',
'him', 'I', 'you', 'very senior CEOs', 'major American car companies', 'my hand', 'I', 'Thrun', 'an
interview', 'Recode']
```

```
Verbs: ['start', 'work', 'drive', 'take', 'tell', 'shake', 'turn', 'be', 'talk', 'say']
```

**c) Named Entity recognition with diagram using NLTK corpus – treebank.**

**Source code:**

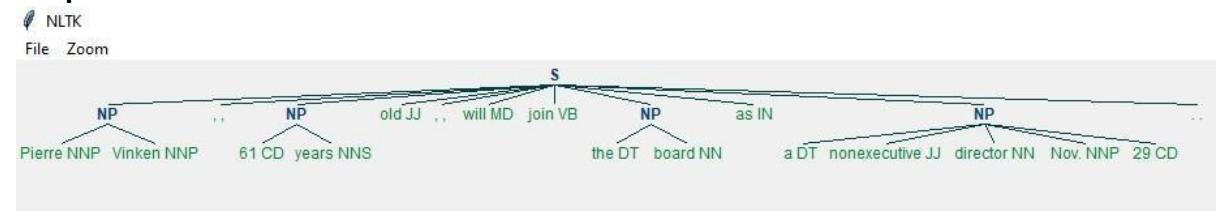
Note: It runs on Python IDLE

```
import nltk
```

```
nltk.download('treebank') from nltk.corpus import treebank_chunk
treebank_chunk.tagged_sents()[0]
```

```
treebank_chunk.chunked_sents()[0]
treebank_chunk.chunked_sents()[0].draw()
```

## Output:



## Practical 7

a) Define grammar using nltk. Analyze a sentence using the same. Code:

```
import nltk from nltk import tokenize grammar1 = nltk.CFG.fromstring(""""
```

```
    S -> VP
```

```
    VP -> VP NP
```

```
    NP -> Det NP
```

```
    Det -> 'that'
```

```
    NP -> singular Noun
```

```
    NP -> 'flight'
```

```
    VP -> 'Book'
```

```
    """")
```

```
sentence = "Book that flight"
```

```
for index in range(len(sentence)):
```

```
    all_tokens = tokenize.word_tokenize(sentence)
```

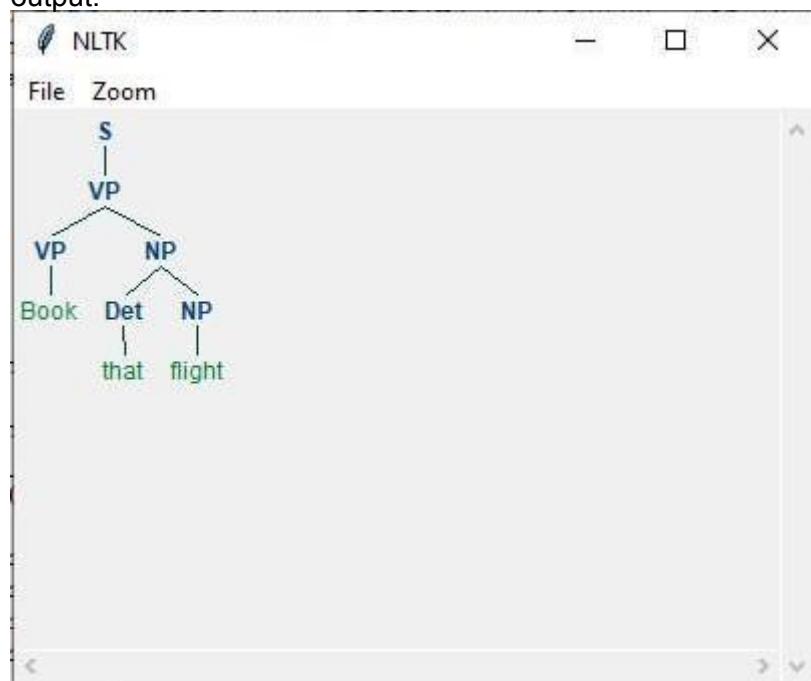
```
print(all_tokens)
```

```
parser = nltk.ChartParser(grammar1) for tree in parser.parse(all_tokens):
```

```
    print(tree)
```

```
    tree.draw()
```

output:



b) Accept the input string with Regular expression of Finite Automaton: 101+.

Source code: def FA(s):

```
#if the length is less than 3 then it can't be accepted, Therefore end the process. if
```

```
len(s)<3:
```

```
    return "Rejected"
```

```

#first three characters are fixed. Therefore, checking them using index      if s[0]=='1':      if
s[1]=='0':          if s[2]=='1':
    # After index 2 only "1" can appear. Therefore break the process if any other
character is detected          for i in range(3,len(s)):          if s[i]!='1':
        return "Rejected"
    return "Accepted" # if all 4 nested if true      return "Rejected" # else of 3rd
if return "Rejected" # else of 2nd if  return "Rejected" # else of 1st if
inputs=['1','10101','101','10111','01010','100','','10111101','1011111'] for i in inputs:
print(FA(i))

```

**Output:**

```

Rejected
Rejected
Accepted
Accepted
Rejected
Rejected
Rejected
Rejected
Accepted

```

**c) Accept the input string with Regular expression of FA: (a+b)\*bba.**

```

Code: def FA(s):  size=0
#scan complete string and make sure that it contains only 'a' & 'b'  for i in s:
    if i=='a' or i=='b':
        size+=1  else:
            return "Rejected"
#After checking that it contains only 'a' & 'b' #check it's length it should be 3 atleast  if
size>=3:
#check the last 3 elements  if s[size-3]=='b':
    if s[size-2]=='b':          if s[size-1]=='a':          return "Accepted" # if all
4 if true      return "Rejected" # else of 4th if      return "Rejected" # else of 3rd if
return "Rejected" # else of 2nd if
    return "Rejected" # else of 1st if

```

```

inputs=['bba', 'ababbba', 'abba','abb', 'baba','bbb',''] for i in inputs:  print(FA(i))

```

**output:** Rejected

```

Rejected
Accepted
Accepted
Rejected
Rejected
Rejected
Rejected
Accepted

```

**d) Implementation of Deductive Chart Parsing using context free grammar and a given sentence. Source code:** import nltk from nltk import tokenize grammar1 =
nltk.CFG.fromstring(""""

S -> NP VP

```

PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'a' | 'my'
N -> 'bird' | 'balcony'
V -> 'saw'
P -> 'in'      """
sentence = "I saw a bird in my balcony"

```

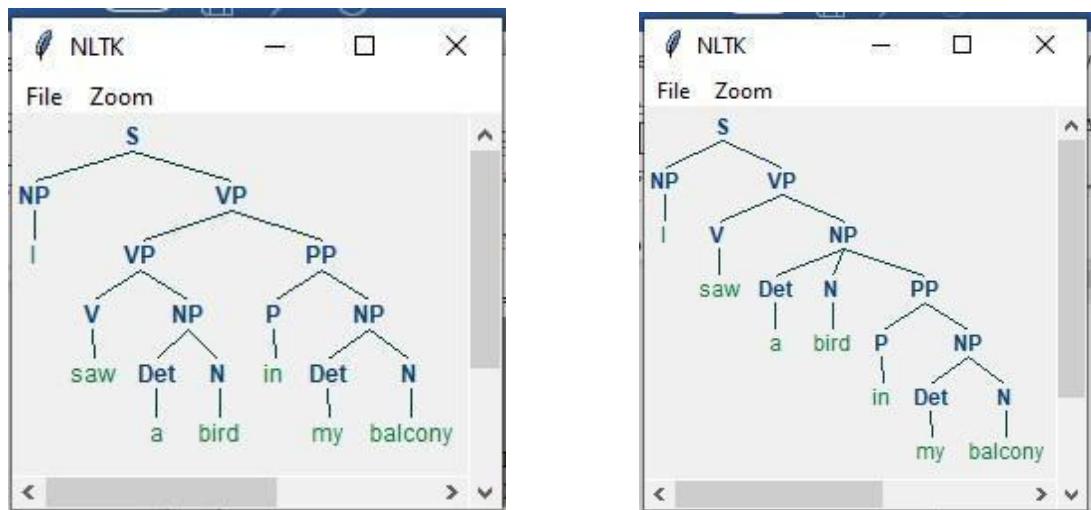
```

for index in range(len(sentence)):
    all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)

# all_tokens = ['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
parser = nltk.ChartParser(grammar1) for tree in parser.parse(all_tokens):
    print(tree)
    tree.draw()

```

**output:**



## Practical 8

### Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer Study WordNetLemmatizer

Code:

```
# PorterStemmer import nltk
from nltk.stem import PorterStemmer word_stemmer = PorterStemmer()
print(word_stemmer.stem('writing')) Output:
```

```
===== RESTART: D:/2020/NLP/Practical/uni/p8aPorterStemmer.py ======
write
>>> |
```

**#LancasterStemmer**

```
import nltk from nltk.stem import LancasterStemmer Lanc_stemmer = LancasterStemmer()
print(Lanc_stemmer.stem('writing')) Output:
```

```
===== RESTART: D:/2020/NLP/Practical/uni/p8bLancasterStemmer.py ======
writ
>>> |
```

**#RegexpStemmer**

```
import nltk
from nltk.stem import RegexpStemmer
Reg_stemmer = RegexpStemmer('ing\$|s\$|e\$|able$', min=4)
print(Reg_stemmer.stem('writing'))
```

**output**

```
===== RESTART: D:/2020/NLP/Practical/uni/p8cRegexprStemmer.py ======
writ
>>> |
```

**#SnowballStemmer** import nltk

```
from nltk.stem import SnowballStemmer english_stemmer = SnowballStemmer('english')
print(english_stemmer.stem ('writing'))
```

**output**

```
===== RESTART: D:/2020/NLP/Practical/uni/p8dSnowballStemmer.py ======
write
>>> |
```

**#WordNetLemmatizer**

```
from nltk.stem import WordNetLemmatizer
```

```
lemmatizer = WordNetLemmatizer()
```

```
print("word :\tlemma") print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
```

```
# a denotes adjective in "pos"
```

```
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

**Output:**

```
===== RESTART: D:/2020/NLP/Practical/uni/p8eWordNetLemmatizer.py ======
word : lemma
rocks : rock
corpora : corpus
better : good
>>> |
```

## Practical 9

### Implement Naive Bayes classifier

Code:

```
#pip install pandas
#pip install sklearn

import pandas as pd import numpy as np

sms_data = pd.read_csv("spam.csv", encoding='latin-1')

import re import nltk from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

stemming = PorterStemmer() corpus = [] for i in range
(0,len(sms_data)): s1 = re.sub('[^a-zA-Z]',repl = ' ',string =
sms_data['v2'][i]) s1.lower()
s1 = s1.split()
s1 = [stemming.stem(word) for word in s1 if word not in set(stopwords.words('english'))]
s1 = ''.join(s1)
corpus.append(s1)

from sklearn.feature_extraction.text import CountVectorizer countvectorizer
=CountVectorizer()

x = countvectorizer.fit_transform(corpus).toarray() print(x)

y = sms_data['v1'].values
print(y)

from sklearn.model_selection import train_test_split x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size = 0.3,
stratify=y,random_state=2)

#Multinomial Naïve Bayes.
from sklearn.naive_bayes import MultinomialNB multinomialnb = MultinomialNB()
multinomialnb.fit(x_train,y_train)

# Predicting on test data:
y_pred = multinomialnb.predict(x_test) print(y_pred)
#Results of our Models
from sklearn.metrics import classification_report, confusion_matrix from sklearn.metrics
import accuracy_score

print(classification_report(y_test,y_pred))
print("accuracy_score: ",accuracy_score(y_test,y_pred))
```

**input:** spam.csv file from github

**output:**

```
===== RESTART: D:\2020\NLP\Practical\uni\p9NaiveBayesClassifier.py =====
[[0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 0 0 0 0 0 0 0 1 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 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 0 0 0 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 2 0 2 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0
 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0]
[0 1 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 0 0 0 0 0 0 0 0 0
 0 0 0 1 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 2 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 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
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 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 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1]
[1 0 0 0 0 0 0 0 1 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 2 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 1 0 0 0 0 1 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 0 0 0 0 0 0 0 1 2 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 2 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 0 0 0 0 0 0 1 0 0 0 0 0 2 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0]
['ham' 'ham' 'spam' 'ham' 'ham' 'spam' 'ham' 'ham' 'spam']
['ham' 'ham' 'ham']

      precision    recall   f1-score   support
      ham       0.67      1.00      0.80       2
      spam       0.00      0.00      0.00       1

      accuracy                           0.67      3
     macro avg       0.33      0.50      0.40       3
weighted avg       0.44      0.67      0.53       3

accuracy_score:  0.6666666666666666
>>> |
```

## Practical 10

### i. Speech tagging using spacy

```
code import spacy
sp = spacy.load('en_core_web_sm')
sen = sp(u"I like to play football. I hated it in my childhood though") print(sen.text)
print(sen[7].pos_) print(sen[7].tag_) print(spacy.explain(sen[7].tag_)) for word in sen:
print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}}'
{spacy.explain(word.tag_)}')

sen = sp(u'Can you go      ogle it?') word = sen[2]
print(f'{word.text:{12}} {word.pos_:{10}}'
{word.tag_:{8}}
{spacy.explain(word.tag_)}) sen = sp(u'Can you search it on google?') word = sen[5]

print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}}'
{spacy.explain(word.tag_)}')

#Finding the Number of POS Tags
sen = sp(u"I like to play football. I hated it in my childhood though")

num_pos = sen.count_by(spacy.attrs.POS)
num_pos

for k,v in sorted(num_pos.items()):
    print(f'{k}. {sen.vocab[k].text:{8}}: {v}')

#Visualizing Parts of Speech Tags
from spacy import displacy

sen = sp(u"I like to play football. I hated it in my childhood though") displacy.serve(sen,
style='dep', options={'distance': 120})
```

#### output:

To view the dependency tree, type the following address in your browser:  
<http://127.0.0.1:5000/>. You will see the following dependency tree:

### ii. Speech tagging

#### using nltk code:

```
import nltk
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer

#create our training and testing data:
```

```
===== RESTART: D:\2020\NLP\Practical\uni\p10a1.py =====
I like to play football. I hated it in my childhood though
VERB
VBD
verb, past tense
I      PRON    PRP    pronoun, personal
like   VERB    VBP    verb, non-3rd person singular present
to     PART    TO     infinitival "to"
play   VERB    VB     verb, base form
football NOUN   NN    noun, singular or mass
.      PUNCT   .
hated   PRON   PRP    pronoun, personal
it     VERB    VBD    verb, past tense
in     ADP    IN     conjunction, subordinating or preposition
my     PRON   PRP$   pronoun, possessive
childhood NOUN   NN    noun, singular or mass
though  ADV    RB     adverb
google  VERB    VB    verb, base form
google  PROPN  NNP    noun, proper singular
85. ADP : 1
86. ADV : 1
92. NOUN : 2
94. PART : 1
95. PRON : 4
97. PUNCT : 1
100. VERB : 3

Using the 'dep' visualizer
Serving on http://0.0.0.0:5000 ...
```

```
train_text = state_union.raw("2005-GWBush.txt")
sample_text = state_union.raw("2006-GWBush.txt")
```

```
#train the Punkt tokenizer like:
custom_sent_tokenizer = PunktSentenceTokenizer(train_text)
```

```
# tokenize:
tokenized = custom_sent_tokenizer.tokenize(sample_text)
```

```
def process_content(): try: for i in tokenized[:2]: words = nltk.word_tokenize(i)
tagged = nltk.pos_tag(words) print(tagged)
```

```
except Exception as e:
    print(str(e))
```

```
process_content()
```

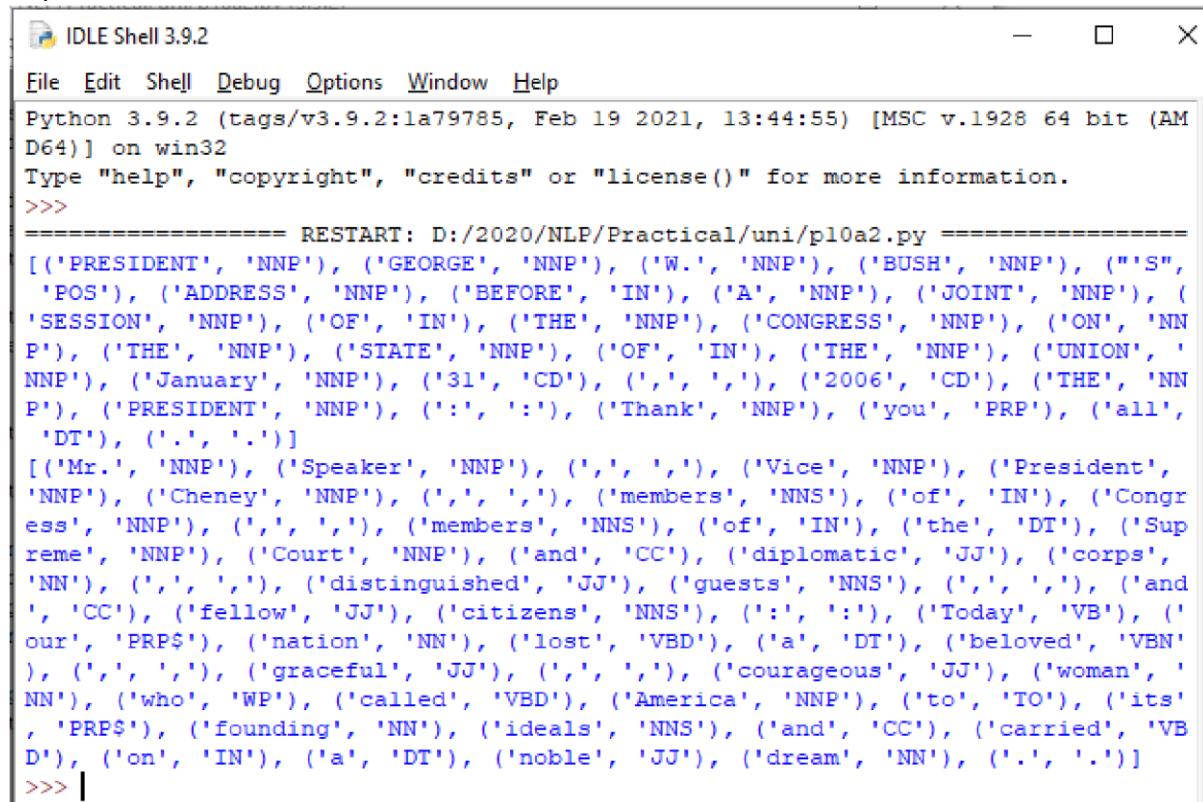
output:

**b. Statistical parsing:**

**i. Usage of Give and Gave in the Penn Treebank sample Source code:**

```
#probabilistic parser
#Usage of Give and Gave in the Penn Treebank sample
```

```
import nltk import nltk.parse.viterbi
import nltk.parse.pchart
```



The screenshot shows the IDLE Shell 3.9.2 interface with the following content:

```
IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: D:/2020/NLP/Practical/uni/pl0a2.py =====
[('PRESIDENT', 'NNP'), ('GEORGE', 'NNP'), ('W.', 'NNP'), ('BUSH', 'NNP'), ("'S",
'POS'), ('ADDRESS', 'NNP'), ('BEFORE', 'IN'), ('A', 'NNP'), ('JOINT', 'NNP'), (
'SESSION', 'NNP'), ('OF', 'IN'), ('THE', 'NNP'), ('CONGRESS', 'NNP'), ('ON', 'NN
P'), ('THE', 'NNP'), ('STATE', 'NNP'), ('OF', 'IN'), ('THE', 'NNP'), ('UNION', 'NN
P'), ('January', 'NNP'), ('31', 'CD'), ('.', '.'), ('2006', 'CD'), ('THE', 'NN
P'), ('PRESIDENT', 'NNP'), (':', ':'), ('Thank', 'NNP'), ('you', 'PRP'), ('all',
'DT'), ('.', '.')]
[('Mr.', 'NNP'), ('Speaker', 'NNP'), ('.', '.'), ('Vice', 'NNP'), ('President',
'NNP'), ('Cheney', 'NNP'), ('.', '.'), ('members', 'NNS'), ('of', 'IN'), ('Congr
ess', 'NNP'), ('.', '.'), ('members', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('Sup
reme', 'NNP'), ('Court', 'NNP'), ('and', 'CC'), ('diplomatic', 'JJ'), ('corps',
'NN'), ('.', '.'), ('distinguished', 'JJ'), ('guests', 'NNS'), ('.', '.'), ('and
', 'CC'), ('fellow', 'JJ'), ('citizens', 'NNS'), (':', ':'), ('Today', 'VB'),
('our', 'PRP$'), ('nation', 'NN'), ('lost', 'VBD'), ('a', 'DT'), ('beloved', 'VBN
'), ('.', '.'), ('graceful', 'JJ'), ('.', '.'), ('courageous', 'JJ'), ('woman',
'NN'), ('who', 'WP'), ('called', 'VBD'), ('America', 'NNP'), ('to', 'TO'), ('its
', 'PRP$'), ('founding', 'NN'), ('ideals', 'NNS'), ('and', 'CC'), ('carried',
'VB
D'), ('on', 'IN'), ('a', 'DT'), ('noble', 'JJ'), ('dream', 'NN'), ('.', '.')]
```

```
def give(t):
    return t.label() == 'VP' and len(t) > 2 and t[1].label() == 'NP' and (t[2].label() == 'PP-
    DTV' or t[2].label() == 'NP') and ('give' in t[0].leaves() or 'gave' in t[0].leaves())
```

```
def sent(t): return ''.join(token for token in t.leaves() if token[0] not in '*-0')
def print_node(t, width):
    output = "%s %s: %s / %s: %s" %
        (sent(t[0]), t[1].label(), sent(t[1]), t[2].label(), sent(t[2])) if len(output) > width:
    output = output[:width] + "..." print (output)
```

```
for tree in nltk.corpus.treebank.parsed_sents(): for t in tree.subtrees(give):
    print_node(t, 72)
```

**Output:**

ii. probabilistic parser Source code: import nltk from nltk import PCFG

```
grammar = PCFG.fromstring("")
NP -> NNS [0.5] | JJ NNS [0.3] | NP CC NP [0.2]
NNS -> "men" [0.1] | "women" [0.2] | "children" [0.3] | NNS CC NNS [0.4]
JJ -> "old" [0.4] | "young" [0.6]
CC -> "and" [0.9] | "or" [0.1]
"")

print(grammar)

viterbi_parser = nltk.ViterbiParser(grammar)
=====
RESTART: D:/2020/NLP/Practical/uni/p10b1.py =====
gave NP: the chefs / NP: a standing ovation
give NP: advertisers / NP: discounts for maintaining or increasing ad sp...
give NP: it / PP-DTV: to the politicians
gave NP: them / NP: similar help
give NP: them / NP:
give NP: only French history questions / PP-DTV: to students in a Europe...
give NP: federal judges / NP: a raise
give NP: consumers / NP: the straight scoop on the U.S. waste crisis
gave NP: Mitsui / NP: access to a high-tech medical product
give NP: Mitsubishi / NP: a window on the U.S. glass industry
give NP: much thought / PP-DTV: to the rates she was receiving , nor to ...
give NP: your Foster Savings Institution / NP: the gift of hope and free...
give NP: market operators / NP: the authority to suspend trading in futu...
gave NP: quick approval / PP-DTV: to $ 3.18 billion in supplemental appr...
give NP: the Transportation Department / NP: up to 50 days to review any...
give NP: the president / NP: such power
give NP: me / NP: the heebie-jeebies
give NP: holders / NP: the right , but not the obligation , to buy a cal...
gave NP: Mr. Thomas / NP: only a `` qualified '' rating , rather than `...
give NP: the president / NP: line-item veto power
>>> |
```

token = "old men and women".split()

obj = viterbi\_parser.parse(token)

print("Output: ") for x in obj: print(x)

**Output:****c. Malt parsing:**

**Parse a sentence and draw a tree using malt parsing.**

Note: 1) Java should be installed.

- 2) maltparser-1.7.2 zip file should be copied in C:\Users\Beena Kapadia\AppData\Local\Programs\Python\Python39 folder and should be extracted in the same folder.
- 3) engmalt.linear-1.7.mco file should be copied to C:\Users\Beena Kapadia\AppData\Local\Programs\Python\Python39 folder **Source code:**

```
# copy maltparser-1.7.2(unzipped version) and engmalt.linear-1.7.mco files to
```

```
===== RESTART: D:/2020/NLP/Practical/uni/p10b2.py =====
Grammar with 11 productions (start state = NP)
NP -> NNS [0.5]
NP -> JJ NNS [0.3]
NP -> NP CC NP [0.2]
NNS -> 'men' [0.1]
NNS -> 'women' [0.2]
NNS -> 'children' [0.3]
NNS -> NNS CC NNS [0.4]
JJ -> 'old' [0.4]
JJ -> 'young' [0.6]
CC -> 'and' [0.9]
CC -> 'or' [0.1]
Output:
(NP (JJ old) (NNS (NNS men) (CC and) (NNS women))) (p=0.000864)
>>> |
```

C:\Users\Beena Kapadia\AppData\Local\Programs\Python\Python39 folder

# java should be installed

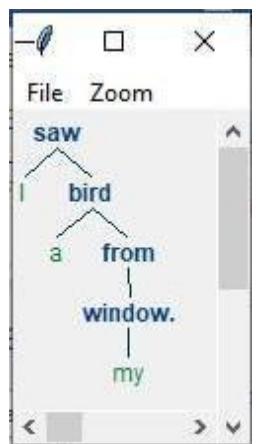
# environment variables should be set - MALT\_PARSER - C:\Users\Beena Kapadia\AppData\Local\Programs\Python\Python39\maltparser-1.7.2 and MALT\_MODEL - C:\Users\Beena

Kapadia\AppData\Local\Programs\Python\Python39\engmalt.linear-1.7.mco

```
from nltk.parse import malt
mp = malt.MaltParser('maltparser-1.7.2', 'engmalt.linear-1.7.mco')#file      t =
mp.parse_one('I saw a bird from my window.'.split()).tree() print(t)
t.draw()
```

#### Output:

(saw I (bird a (from (window. my))))



## Practical 11

### a) Multiword Expressions in NLP Source code:

```
# Multiword Expressions in NLP
```

```
from nltk.tokenize import MWETokenizer from nltk import sent_tokenize, word_tokenize
s = "Good cake cost Rs.1500\kg in Mumbai. Please buy me one of them.\n\nThanks." mwe
= MWETokenizer([('New', 'York'), ('Hong', 'Kong')], separator='_') for sent in
sent_tokenize(s): print(mwe.tokenize(word_tokenize(sent))) Output:
```

```
===== RESTART: D:/2020/NLP/Practical/uni/plla.py =====
['Good', 'cake', 'cost', 'Rs.1500\\kg', 'in', 'Mumbai', '.']
['Please', 'buy', 'me', 'one', 'of', 'them', '.']
['Thanks', '.']
>>> |
```

### b) Normalized Web Distance and Word Similarity Source code:

```
# Normalized Web Distance and Word Similarity
```

```
#convert
```

```
#Reliance
supermarket
#Reliance
hypermarket
#Reliance
#Reliance
#Reliance
#Reliance downtown
#Relianc market
#Mumbai
#Mumbai Hyper
#Mumbai dxb
#mumbai airport
#k.m trading
#KM Trading
#KM trade
#K.M. Trading
#KM.Trading
```

```
#into
```

```
#Reliance
#Reliance
#Reliance
#Reliance
#Reliance
#Reliance
#Reliance
#Mumbai
#Mumbai
#Mumbai
#Mumbai
#KM Trading
#KM Trading
```

```

#KM Trading
#KM Trading
#KM Trading

import numpy as np
import re
import textdistance # pip install textdistance
# we will need scikit-learn>=0.21 import sklearn #pip install sklearn
from sklearn.cluster import AgglomerativeClustering

texts = [
    'Reliance supermarket', 'Reliance hypermarket', 'Reliance', 'Reliance', 'Reliance downtown',
    'Reliance market',
    'Mumbai', 'Mumbai Hyper', 'Mumbai dxb', 'mumbai airport',
    'k.m trading', 'KM Trading', 'KM trade', 'K.M. Trading', 'KM.Trading'
]

def normalize(text):
    """ Keep only lower-cased text and numbers"""
    return re.sub('[^a-z0-9]+', ' ', text.lower())

def group_texts(texts, threshold=0.4):
    """ Replace each text with the representative of its cluster"""
    normalized_texts = np.array([normalize(text) for text in texts])
    distances = 1 - np.array([
        [textdistance.jaro_winkler(one, another) for one in normalized_texts] for another in
        normalized_texts
    ])
    clustering = AgglomerativeClustering(
        distance_threshold=threshold, # this parameter needs to be tuned carefully
        affinity="precomputed", linkage="complete", n_clusters=None
    ).fit(distances)
    centers = dict()
    for cluster_id in set(clustering.labels_):
        index = np.where(clustering.labels_ == cluster_id)[0]
        centrality = distances[:, index][index].sum(axis=1)
        centers[cluster_id] = normalized_texts[index][centrality.argmin()]
    return [centers[i] for i in clustering.labels_]

print(group_texts(texts))

```

#### Output:

```

=====
RESTART: D:/2020/NLP/Practical/uni/pl1b.py =====
['reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'mumbai',
 'mumbai', 'mumbai', 'mumbai', 'km trading', 'km trading', 'km trading', 'km t
 rading', 'km trading']
>>> |

```

#### c) Word Sense Disambiguation Source code:

```
#Word Sense Disambiguation from nltk.corpus import wordnet as wn
```

```

def get_first_sense(word, pos=None):
    if pos:
        synsets = wn.synsets(word, pos)
    else:
        synsets = wn.synsets(word)
    return synsets[0]

best_synset = get_first_sense('bank')
print ('%s: %s' % (best_synset.name, best_synset.definition))
best_synset =
get_first_sense('set','n')

```

```
print ('%s: %s' % (best_synset.name, best_synset.definition)) best_synset =  
get_first_sense('set','v') print ('%s: %s' % (best_synset.name, best_synset.definition))
```

**Output:**

# **DEEP**

# **LEARNING**

## INDEX

| Sr. No | Practicals   | Sign |
|--------|--|------|
| 1      | Get the result of XOR gate as the output by using perceptron model               |      |
| 2      | Find the local minima by using gradient descent                                  |      |
| 3      | Do image classification by using Convolutional neural network                    |      |
| 4      | Do Stock market prediction of Reliance company by using Recurrent neural network |      |
| 5      | Do Stock market prediction of Reliance company by using Recurrent neural network |      |
| 6      | Do feature extraction using Linear factor model                                  |      |
| 7      | Do auto encoding of the images   |      |
| 8      | Convert speech (audio format) into text format by neural network                 |      |
| 9      | Predict the perfect emoji for the given text                                     |      |
| 10     | To remove the noise from the image using autoencoders                            |      |
| 11     | To detect the object from image accurately.                                      |      |
| 12     | To classify the disease type by using CNN mode                                   |      |

## Practical 1

**Aim :** To get the result of XOR gate as the output by using perceptron model

**Theory :** First we build the function for classification of 0 and 1 that is. a unitStep function, and after that we've made a simple perceptron model which is doing dot product of the inputs, then we have added NOT,AND,OR,XOR logic functions and we have assigned weights and biases and passed the values to perceptron model for each function respectively finally we pass truth table values and we get correct result.

**Input:**

```
In [1]: import numpy as np

In [2]: #Defin unit step function

In [3]: def unitStep(v):
         if v>=0:
             return 1
         else:
             return 0

In [4]: #design perceptron model

In [5]: def perceptronModel(x,w,b):
         v=np.dot(x,w)+b
         y=unitStep(v)
         return y

In [6]: #Not logic
def NOT_logicFunction(x):
    wNOT = -1
    bNOT = 0.5
    return perceptronModel(x, wNOT, bNOT)

In [7]: #And logic
def AND_logicFunction(x):
    w = np.array([1, 1])
    bAND = -1.5
    return perceptronModel(x, w, bAND)
```

```
In [8]: #OR logic
def OR_logicFunction(x):
    w = np.array([1, 1])
    bOR = -0.5
    return perceptronModel(x, w, bOR)
```

```
In [9]: #XOR Logic Function
def XOR_logicFunction(x):
    y1 = AND_logicFunction(x)
    y2 = OR_logicFunction(x)
    y3 = NOT_logicFunction(y1)
    final_x = np.array([y2, y3])
    finalOutput = AND_logicFunction(final_x)
    return finalOutput
```

```
In [10]: test1 = np.array([0, 0])
test2 = np.array([0, 1])
test3 = np.array([1, 0])
test4 = np.array([1, 1])
```

#### Output:

```
In [11]: print("XOR({}, {}) = {}".format(0, 1, XOR_logicFunction(test1)))
print("XOR({}, {}) = {}".format(0, 1, XOR_logicFunction(test2)))
print("XOR({}, {}) = {}".format(1, 0, XOR_logicFunction(test3)))
print("XOR({}, {}) = {}".format(1, 1, XOR_logicFunction(test4)))
```

```
XOR(0, 1) = 0
XOR(0, 1) = 1
XOR(1, 0) = 1
XOR(1, 1) = 0
```

**Conclusion:** We can easily build the XOR gate by help of simple basic perceptron model

## Practical 2

**Aim :** To find the local minima by using gradient descent

**Theory :** We pass the current value as 3, step size, maximum iterations and precision at the start and we loop through each iteration and we obtain the derivation function after looping on values then we are ready to plot our derivative function after finding derivative function plot we will plot our final graph and we zoom in to the graph to see more details finally we get the local minima as 2.66 after running through multiple epochs

**Input:**

```
In [1]: #experiment 2 gradient descent (finding local minima)
cur_x = 3 # The algorithm starts at x=3
rate = 0.01 # Learning rate
precision = 0.000001 #This tells us when to stop the algorithm
previous_step_size = 1 #
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
df = lambda x: 2*(x+5) #Gradient of our function

while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x #Store current x value in prev_x
    cur_x = cur_x - rate * df(prev_x) #Grad descent
    previous_step_size = abs(cur_x - prev_x) #Change in x
    iters = iters+1 #iteration count
    print("Iteration",iters,"X value is",cur_x) #Print iterations

print("The local minimum occurs at", cur_x)
```

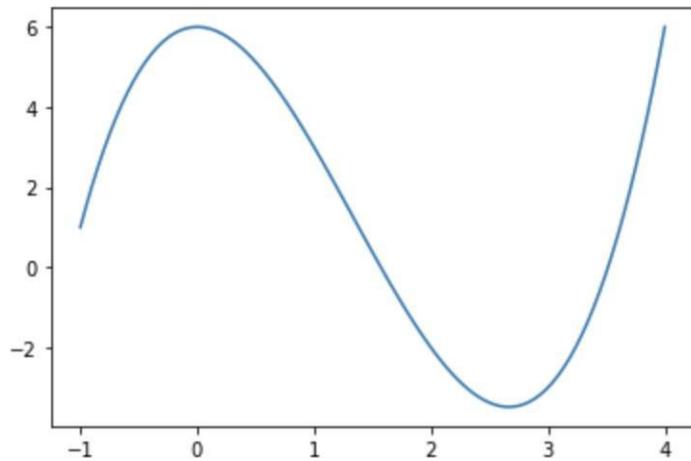
Iteration 1  
X value is 2.84  
Iteration 2  
X value is 2.6832  
Iteration 3  
X value is 2.529536  
Iteration 4  
X value is 2.37894528  
Iteration 5  
X value is 2.2313663744  
Iteration 6  
X value is 2.0867390469119997  
Iteration 7  
X value is 1.9450042659737599  
Iteration 8  
X value is 1.8061041806542846  
Iteration 9  
X value is 1.669982097041199  
Iteration 10

```
In [2]: import numpy as np
import matplotlib.pyplot as plt

f_x = lambda x: (x**3)-4*(x**2)+6
x = np.linspace(-1,4,100)

#Plot the curve
plt.plot(x, f_x(x))
plt.show()
#plotting of first order derivative

f_x_derivative = lambda x: 3*(x**2)-8*x
```



```
In [3]: def plot_gradient(x, y, x_vis, y_vis):
    plt.subplot(1,2,2)
    plt.scatter(x_vis, y_vis, c = "b")
    plt.plot(x, f_x(x), c = "r")
    plt.title("Gradient Descent")
    plt.show()

    plt.subplot(1,2,1)
    plt.scatter(x_vis, y_vis, c = "b")
    plt.plot(x,f_x(x), c = "r")
    plt.xlim([2.0,3.0])
    plt.title("Zoomed in Figure")
    plt.show()
```

```

def gradient_iteration(x_start, iterations, learning_rate):

    x_grad = [x_start]
    y_grad = [f_x(x_start)]

    for i in range(iterations):

        x_start_derivative = - f_x_derivative(x_start)

        x_start += (learning_rate * x_start_derivative)

        x_grad.append(x_start)
        y_grad.append(f_x(x_start))

    print ("Local minimum occurs at: {:.2f}".format(x_start))
    print ("Number of steps: ",len(x_grad)-1)
    plot_gradient(x, f_x(x) ,x_grad, y_grad)

gradient_iteration(0.5, 1000, 0.05)

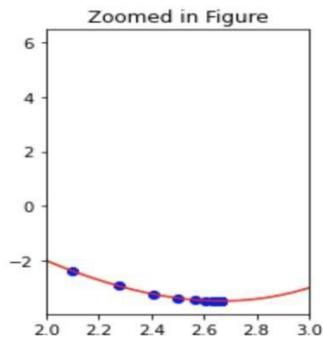
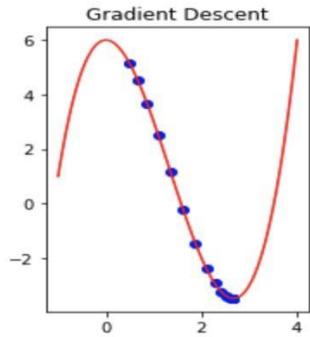
```

### Output:

```

Local minimum occurs at: 2.67
Number of steps: 1000

```



**Conclusion :** We are able to create the gradient descent function perfectly and we are getting accurate local minima results

## Practical 3

**Aim :** To do image classification by using Convolutional neural network

**Theory :** We are going to use inbuilt cifar dataset for image classification task in this experiment, we have provided classes and tags for each image after that we have trained our CNN model with training dataset and finally we are getting perfect output our model is doing fantastic job with 79% accuracy on training data.

**Input:**

```
In [1]: import tensorflow as tf
        from tensorflow.keras import datasets, layers, models
        import matplotlib.pyplot as plt
        import numpy as np
        from keras.datasets import cifar10

In [2]: (X_train, y_train), (X_test,y_test) = datasets.cifar10.load_data()
        X_train.shape

Out[2]: (50000, 32, 32, 3)

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [4]: y_train.shape

Out[4]: (50000, 1)

In [5]: y_train[:5]

Out[5]: array([[6,
       9,
       9,
       4,
       1]], dtype=uint8)

In [6]: y_train = y_train.reshape(-1,)
        y_train[:5]

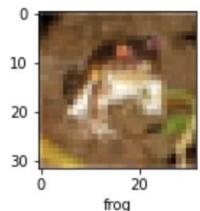
Out[6]: array([6, 9, 9, 4, 1], dtype=uint8)

In [7]: y_test = y_test.reshape(-1,)
```

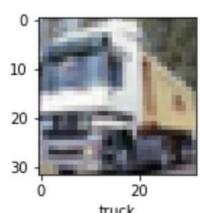
```
In [8]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
```

```
In [9]: def plot_sample(x, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

```
In [10]: plot_sample(X_train, y_train, 0)
```



```
In [11]: plot_sample(X_train, y_train, 1)
```



```
In [12]: X_train = X_train / 255.0
X_test = X_test / 255.0
```

```
In [13]: cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
In [14]: cnn.compile(optimizer='adam',
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])
```

```
In [15]: cnn.fit(X_train, y_train, epochs=10)
```

```
In [15]: cnn.fit(x_train, y_train, epochs=10)

Epoch 1/10
1563/1563 [=====] - 21s 13ms/step - loss: 1.7069 - accuracy: 0.3754
Epoch 2/10
1563/1563 [=====] - 21s 14ms/step - loss: 1.1595 - accuracy: 0.5890
Epoch 3/10
1563/1563 [=====] - 20s 13ms/step - loss: 1.0082 - accuracy: 0.6475
Epoch 4/10
1563/1563 [=====] - 19s 12ms/step - loss: 0.9102 - accuracy: 0.6857
Epoch 5/10
1563/1563 [=====] - 18s 12ms/step - loss: 0.8445 - accuracy: 0.7043
Epoch 6/10
1563/1563 [=====] - 18s 11ms/step - loss: 0.7804 - accuracy: 0.7293
Epoch 7/10
1563/1563 [=====] - 18s 11ms/step - loss: 0.7259 - accuracy: 0.7471
Epoch 8/10
1563/1563 [=====] - 204s 131ms/step - loss: 0.6719 - accuracy: 0.7662
Epoch 9/10
1563/1563 [=====] - 20s 13ms/step - loss: 0.6319 - accuracy: 0.7798
Epoch 10/10
1563/1563 [=====] - 20s 13ms/step - loss: 0.5854 - accuracy: 0.7989

Out[15]: <tensorflow.python.keras.callbacks.History at 0x7fda4084b400>

In [16]: cnn.evaluate(x_test,y_test)

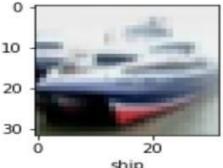
313/313 [=====] - 1s 2ms/step - loss: 0.9267 - accuracy: 0.6983

Out[16]: [0.9267432689666748, 0.6983000040054321]
```

## Output:

```
In [17]: print(y_test[:4])
plot_sample(x_test,y_test,1)

[3 8 8 0]

In [18]: y_pred = cnn.predict(x_test)
y_pred[:4]

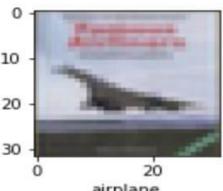
In [19]: y_classes = [np.argmax(element) for element in y_pred]
y_classes[:4]

Out[19]: [3, 8, 1, 0]

In [20]: y_test[:4]

Out[20]: array([3, 8, 8, 0], dtype=uint8)

In [21]: plot_sample(x_test, y_test,3)

In [22]: classes[y_classes[3]]

Out[22]: 'airplane'
```

**Conclusion :** We got 79% accuracy and model is classifying airplane image perfectly in output.

## Practical 4

**Aim :** To do Stock market prediction of Reliance company by using Recurrent neural network

**Theory :** First we have downloaded data of stock history of reliance from 1st jan 1996 to 17th april 2021 and we have tried to do prediction by using RNN model and we have used tensorflow library version 1.12, we have sorted the dataset by cleaning the data then we have prepared the model using tensorflow library which is giving 90% accuracy.

**Input:**

```
In [1]: import numpy as np
import pandas as pd
import math
import sklearn
import sklearn.preprocessing
import datetime
import os
import matplotlib.pyplot as plt
import tensorflow as tf

In [2]: dataset = pd.read_csv('RELIANCE.NS.csv', index_col = 0)
df_stock = dataset.copy()
df_stock = df_stock.dropna()
df_stock = df_stock[['Open', 'High', 'Low', 'Close']]

In [3]: def normalize_data(df):
    min_max_scaler = sklearn.preprocessing.MinMaxScaler()
    df['Open'] = min_max_scaler.fit_transform(df.Open.values.reshape(-1,1))
    df['High'] = min_max_scaler.fit_transform(df.High.values.reshape(-1,1))
    df['Low'] = min_max_scaler.fit_transform(df.Low.values.reshape(-1,1))
    df['Close'] = min_max_scaler.fit_transform(df['Close'].values.reshape(-1,1))
    return df
df_stock_norm = df_stock.copy()
df_stock_norm = normalize_data(df_stock_norm)

In [4]: valid_set_size_percentage = 10
test_set_size_percentage = 10
seq_len = 20

def load_data(stock, seq_len):
    data_raw = stock.to_numpy()
    data = []
    for index in range(len(data_raw) - seq_len):
        data.append(data_raw[index: index + seq_len])
    data = np.array(data);
    valid_set_size = int(np.round(valid_set_size_percentage/100*data.shape[0]));
    test_set_size = int(np.round(test_set_size_percentage/100*data.shape[0]));
    train_set_size = data.shape[0] - (valid_set_size + test_set_size);
    x_train = data[:train_set_size,:-1,:]
    y_train = data[:train_set_size,-1,:]
    x_valid = data[train_set_size:train_set_size+valid_set_size,:,:-1,:]
    y_valid = data[train_set_size:train_set_size+valid_set_size,-1,:]
    x_test = data[train_set_size+valid_set_size:,:-1,:]
    y_test = data[train_set_size+valid_set_size:,-1,:]
    return [x_train,y_train,x_valid,y_valid,x_test,y_test]

x_train,y_train,x_valid,y_valid,x_test,y_test = load_data(df_stock_norm,seq_len)
print('x_train.shape=',x_train.shape)
print('y_train.shape=',y_train.shape)
print('x_valid.shape=',x_valid.shape)
print('y_valid.shape=',y_valid.shape)
print('x_test.shape=',x_test.shape)
print('y_test.shape=',y_test.shape)
```

```
x_train.shape= (5070, 19, 4)
y_train.shape= (5070, 4)
x_valid.shape= (634, 19, 4)
y_valid.shape= (634, 4)
x_test.shape= (634, 19, 4)
y_test.shape= (634, 4)
```

```
In [5]: from tensorflow.python.framework import ops

n_steps = seq_len - 1
n_inputs = 4
n_neurons = 200
n_outputs = 4
n_layers = 2
learning_rate = 0.001
batch_size = 50
n_epochs = 100
train_set_size = x_train.shape[0]
test_set_size = x_test.shape[0]
ops.reset_default_graph()
X = tf.placeholder(tf.float32,[None, n_steps, n_inputs])
Y = tf.placeholder(tf.float32,[None, n_outputs])
```

```
In [6]: index_in_epoch = 0;
perm_array = np.arange(x_train.shape[0])
np.random.shuffle(perm_array)

def get_next_batch(batch_size):
    global index_in_epoch, x_train, perm_array
    start = index_in_epoch
    index_in_epoch += batch_size
    if index_in_epoch > x_train.shape[0]:
        np.random.shuffle(perm_array)
        start = 0
        index_in_epoch = batch_size
    end = index_in_epoch
    return x_train[perm_array[start:end]], y_train[perm_array[start:end]]
```

```
In [7]: layers = [tf.contrib.rnn.BasicRNNCell(num_units = n_neurons, activation = tf.nn.elu) for layer in range(n_layers)]
multi_layer_cell = tf.contrib.rnn.MultiRNNCell(layers)
rnn_outputs, states = tf.nn.dynamic_rnn(multi_layer_cell,X, dtype = tf.float32)
stacked_rnn_outputs = tf.reshape(rnn_outputs,[-1, n_neurons])
stacked_outputs = tf.layers.dense(stacked_rnn_outputs,n_outputs)
outputs = tf.reshape(stacked_outputs,[-1,n_steps,n_outputs])
outputs = outputs[:,n_steps-1,:,:]
```

```
In [9]: loss = tf.reduce_mean(tf.square(outputs - Y))
```

```
In [10]: optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
training_op = optimizer.minimize(loss)
```

```
In [12]: with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for iteration in range(int(n_epochs*train_set_size/batch_size)):
        x_batch, y_batch = get_next_batch(batch_size)
        sess.run(training_op, feed_dict={X:x_batch,Y:y_batch})
        if iteration % int(5*train_set_size/batch_size) == 0:
            mse_train = loss.eval(feed_dict={X:X_train,Y:y_train})
            mse_valid = loss.eval(feed_dict={X:X_valid,Y:y_valid})
            print('%.2f epochs : MSE train/valid = %.6f/%.6f'%(iteration*batch_size/train_set_size,mse_train,mse_valid))
    y_test_pred = sess.run(outputs,feed_dict={X:X_test})

0.00 epochs : MSE train/valid = 0.024536/0.076188
5.00 epochs : MSE train/valid = 0.000024/0.000085
10.00 epochs : MSE train/valid = 0.000029/0.000254
15.00 epochs : MSE train/valid = 0.000013/0.000045
20.00 epochs : MSE train/valid = 0.000015/0.000034
25.00 epochs : MSE train/valid = 0.000009/0.000042
30.00 epochs : MSE train/valid = 0.000009/0.000044
35.00 epochs : MSE train/valid = 0.000009/0.000047
40.00 epochs : MSE train/valid = 0.000008/0.000029
45.00 epochs : MSE train/valid = 0.000013/0.000072
50.00 epochs : MSE train/valid = 0.000010/0.000045
55.00 epochs : MSE train/valid = 0.000009/0.000022
60.00 epochs : MSE train/valid = 0.000013/0.000076
65.00 epochs : MSE train/valid = 0.000010/0.000030
70.00 epochs : MSE train/valid = 0.000007/0.000032
75.00 epochs : MSE train/valid = 0.000008/0.000022
80.00 epochs : MSE train/valid = 0.000008/0.000021
85.00 epochs : MSE train/valid = 0.000009/0.000027
90.00 epochs : MSE train/valid = 0.000008/0.000041
95.00 epochs : MSE train/valid = 0.000007/0.000026
```

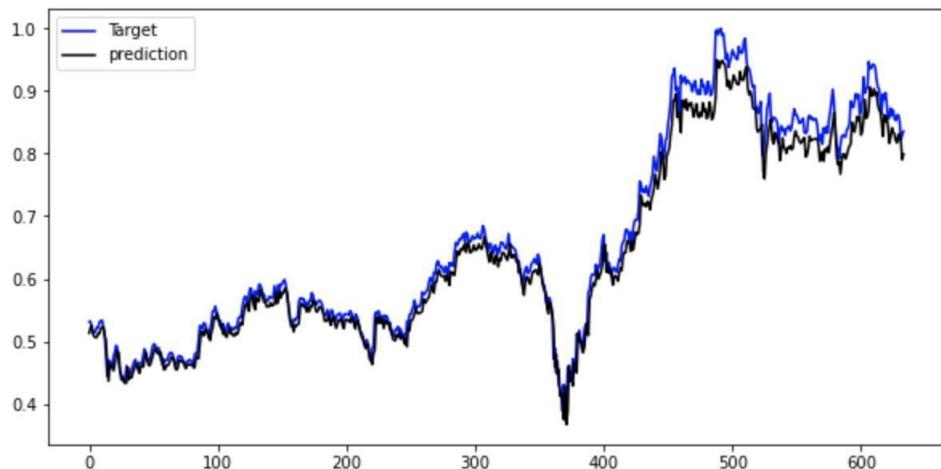
```
In [13]: y_test_pred.shape
```

```
Out[13]: (634, 4)
```

## Output:

```
In [14]: comp = pd.DataFrame({'Column1':y_test[:,3],'Column2':y_test_pred[:,3]})

plt.figure(figsize=(10,5))
plt.plot(comp['Column1'],color='blue',label='Target')
plt.plot(comp['Column2'],color='black',label='prediction')
plt.legend()
plt.show()
```



**Conclusion :** As we can see in the output the black line is predicted results all prediction was done by our model it is very close to the actual data we can conclude that RNN works very well for stock market prediction with very negligible loss rate.

## Practical 5

**Aim :** To do Stock market prediction of TATA company by using Long short term memory

**Theory :** First we have downloaded data of stock history of TATA for training and test dataset and we have tried to do prediction by using LSTM model and we have used tensorflow keras library We trained our LSTM model by adding a sequential model and obtaining good results by help of LSTM .

**Input:**

```
In [10]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout

In [11]: dataset_train = pd.read_csv('NSE-TATAGLOBAL.csv')
training_set = dataset_train.iloc[:, 1:2].values

In [12]: dataset_train.head()

Out[12]:
   Date  Open  High  Low  Last  Close  Total Trade Quantity  Turnover (Lacs)
0  2018-09-28  234.05  235.95  230.20  233.50  233.75          3069914      7162.35
1  2018-09-27  234.55  236.80  231.10  233.80  233.25          5082859      11859.95
2  2018-09-26  240.00  240.00  232.50  235.00  234.25          2240909      5248.60
3  2018-09-25  233.30  236.75  232.00  236.25  236.10          2349368      5503.90
4  2018-09-24  233.55  239.20  230.75  234.00  233.30          3423509      7999.55

In [13]: sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)

In [14]: X_train = []
y_train = []
for i in range(60, 2035):
    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)

X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

```
In [15]: 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)
Epoch 92/100
62/62 [=====] - 3s 51ms/step - loss: 6.9065e-04
Epoch 93/100
62/62 [=====] - 3s 51ms/step - loss: 6.7258e-04
Epoch 94/100
62/62 [=====] - 3s 51ms/step - loss: 6.1093e-04
Epoch 95/100
62/62 [=====] - 3s 51ms/step - loss: 6.4324e-04
Epoch 96/100
62/62 [=====] - 3s 51ms/step - loss: 7.5046e-04
Epoch 97/100
62/62 [=====] - 3s 51ms/step - loss: 5.5045e-04
Epoch 98/100
62/62 [=====] - 3s 50ms/step - loss: 6.2113e-04
Epoch 99/100
62/62 [=====] - 3s 50ms/step - loss: 8.3116e-04
Epoch 100/100
62/62 [=====] - 3s 50ms/step - loss: 5.8380e-04

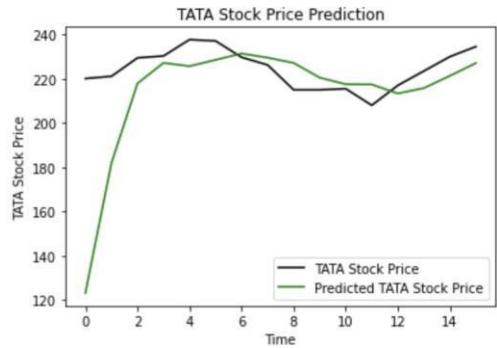
Out[15]: <tensorflow.python.keras.callbacks.History at 0x7f951d3308b0>
```

```
In [16]: dataset_test = pd.read_csv('tataatest.csv')
real_stock_price = dataset_test.iloc[:, 1:2].values

In [17]: 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, 76):
    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)
```

**Output:**

```
In [18]: plt.plot(real_stock_price, color = 'black', label = 'TATA Stock Price')
plt.plot(predicted_stock_price, color = 'green', label = 'Predicted TATA Stock Price')
plt.title('TATA Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('TATA Stock Price')
plt.legend()
plt.show()
```



**Conclusion:** So as we can see the green line is predicted Stock price and black one is actual stock price, we can conclude that our model is perfectly predicting the market for TATA stocks.

## Practical 6

**Aim :** To do feature extraction using Linear factor model

**Theory :** We are going to create sinusoidal signals and then by help of ica and pca we are going to extract features from the signal that we created.

**Input:**

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

from sklearn.decomposition import FastICA, PCA

np.random.seed(0) # set seed for reproducible results
n_samples = 2000
time = np.linspace(0, 8, n_samples)

s1 = np.sin(2 * time) # Signal 1 : sinusoidal signal
s2 = np.sign(np.sin(3 * time)) # Signal 2 : square signal
s3 = signal.sawtooth(2 * np.pi * time) # Signal 3: sawtooth signal

S = np.c_[s1, s2, s3]
S += 0.2 * np.random.normal(size=S.shape) # Add noise

S /= S.std(axis=0) # Standardize data
# Mix data
A = np.array([[1, 1, 1], [0.5, 2, 1.0], [1.5, 1.0, 2.0]]) # Mixing matrix
X = np.dot(S, A.T) # Generate observations
```

```
In [2]: # compute ICA
ica = FastICA(n_components=3)
S_ = ica.fit_transform(X) # Get the estimated sources
A_ = ica.mixing_ # Get estimated mixing matrix

# compute PCA
pca = PCA(n_components=3)
H = pca.fit_transform(X) # estimate PCA sources

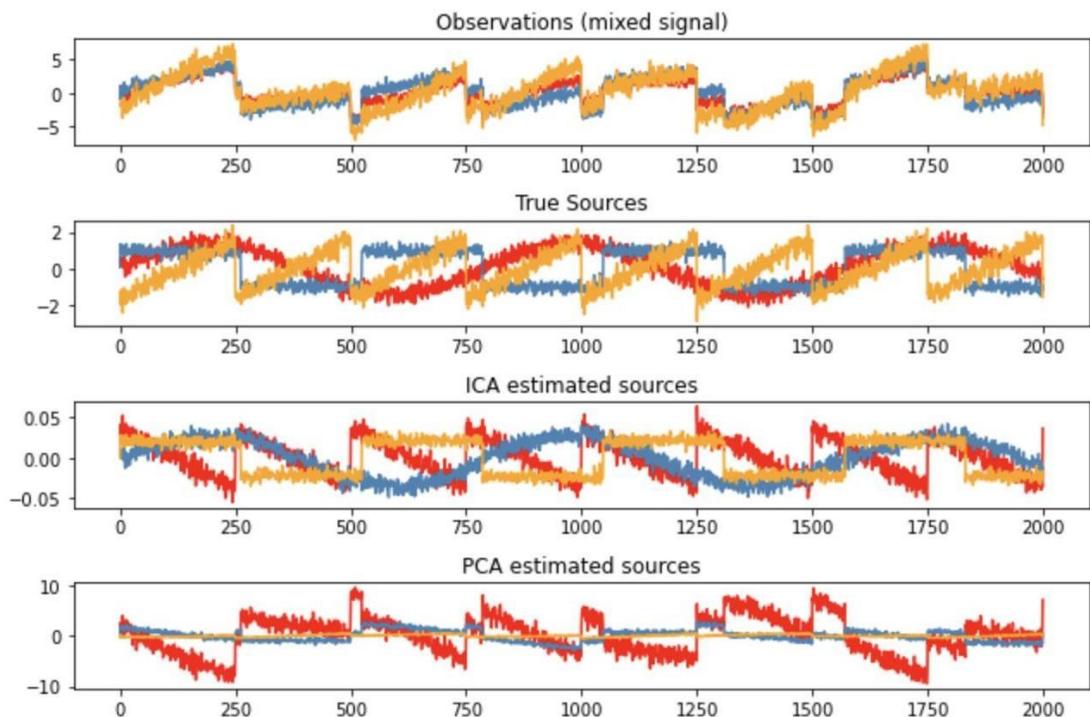
plt.figure(figsize=(9, 6))

models = [X, S, S_, H]
names = ['Observations (mixed signal)', 
         'True Sources',
         'ICA estimated sources',
         'PCA estimated sources']
colors = ['red', 'steelblue', 'orange']

for ii, (model, name) in enumerate(zip(models, names), 1):
    plt.subplot(4, 1, ii)
    plt.title(name)
    for sig, color in zip(model.T, colors):
        plt.plot(sig, color=color)

plt.tight_layout()
```

**Output:**



**Conclusion :** As we can see in above output the first diagram is a mixed signal, in the second diagram we can see that the first signal is a mixture of three signals yellow, red and blue. And by using ICA and PCA we can exactly extract the signal from mixed signal and can identify them clearly

## Practical 7

**Aim :** To do auto encoding of the images

**Theory :** We are going to take famous mnist digit dataset to perform this task,

**Input:**

```
▶ import numpy as np
    from keras.layers import Input, Dense
    from keras.models import Model
    from keras.datasets import mnist
    import matplotlib.pyplot as plt

▶ # this is the size of our encoded representations
    encoding_dim = 32 # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

    # this is our input placeholder
    input_img = Input(shape=(784,))
    # "encoded" is the encoded representation of the input
    encoded = Dense(encoding_dim, activation='relu')(input_img)
    # "decoded" is the lossy reconstruction of the input
    decoded = Dense(784, activation='sigmoid')(encoded)
    # this model maps an input to its reconstruction
    autoencoder = Model(input_img, decoded)
    # this model maps an input to its encoded representation
    encoder = Model(input_img, encoded)
    # create a placeholder for an encoded (32-dimensional) input
    encoded_input = Input(shape=(encoding_dim,))
    # retrieve the last layer of the autoencoder model
    decoder_layer = autoencoder.layers[-1]
    # create the decoder model
    decoder = Model(encoded_input, decoder_layer(encoded_input))
    # configure our model to use a per-pixel binary crossentropy loss, and the Adadelta optimizer:
    autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')

[ ] [(x_train, _), (x_test, _)] = mnist.load_data()
    # normalize all values between 0 and 1 and we will flatten the 28x28 images into vectors of size 784.
    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)

    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [=====] - 0s 0us/step
(60000, 784)
(10000, 784)
```

```

autoencoder.fit(x_train, x_train,
epochs=50,
batch_size=256,
shuffle=True,
validation_data=(x_test, x_test))
# encode and decode some digits
# note that we take them from the *test* set
encoded_imgs = encoder.predict(x_test)
decoded_imgs = decoder.predict(encoded_imgs)

235/235 [=====] - 3s 11ms/step - loss: 0.6885 - val_loss: 0.6882
Epoch 22/50
235/235 [=====] - 2s 10ms/step - loss: 0.6882 - val_loss: 0.6880
Epoch 23/50
235/235 [=====] - 2s 10ms/step - loss: 0.6879 - val_loss: 0.6877
Epoch 24/50

```

### Output:

```

n = 20 # how many digits we will display
plt.figure(figsize=(20, 4))
for i in range(n):
# display original
    ax = plt.subplot(2, n, 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 reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()

```



**Conclusion :** As we can see the encoding of the image to the bottleneck and then decoding of the digits is done perfectly and both the autoencoders are working pretty well.

## Practical 8

**Aim :** To convert speech (audio format) into text format by neural network

**Theory :** First we have done audio preprocessing on the input audio data we have done various operations on the audio data and then we passed the audio file to the CNN model, and our CNN model is taking care of all the attributes and learning the patterns from the audio perfectly

**Input:**

```
In [1]: import os
import librosa    #for audio processing
import IPython.display as ipd
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile #for audio processing
import warnings

train_audio_path = librosa.load('input.wav', sr = 16000)
samples, sample_rate = train_audio_path
```

```
In [2]: ipd.Audio(samples, rate=sample_rate)
print(sample_rate)
```

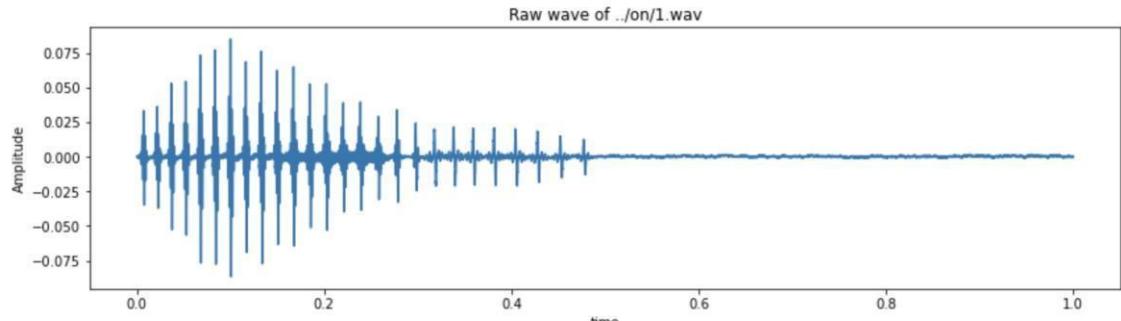
16000

```
In [3]: os.listdir('input')
```

```
Out[3]: ['bed',
'bird',
'cat',
'dog',
'down',
'eight',
'five',
'four',
'go',
'happy',
']'
```

```
In [4]: train_audio_path = 'input'
samples, sample_rate = librosa.load(train_audio_path + '/on/1.wav', sr = 16000)
fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(211)
ax1.set_title('Raw wave of ' + '.../on/1.wav')
ax1.set_xlabel('time')
ax1.set_ylabel('Amplitude')
ax1.plot(np.linspace(0, sample_rate/len(samples), sample_rate), samples)
```

```
Out[4]: <matplotlib.lines.Line2D at 0x2670a8a81c0>
```



```
In [5]: ipd.Audio(samples, rate=sample_rate)
```

```
Out[5]:
```

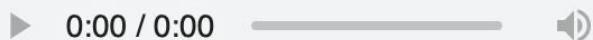


```
In [6]: print(sample_rate)
```

```
16000
```

```
In [7]: samples = librosa.resample(samples, sample_rate, 8000)
ipd.Audio(samples, rate=8000)
```

```
Out[7]:
```



```
In [8]: labels=os.listdir(train_audio_path)
```

```
In [13]: labels=["bed", "bird", "cat", "dog", "down", "eight", "five", "four", "go", "happy", "house", "left", "marvel", "nine", "no"]
```

```
In [15]: train_audio_path = 'input'
all_wave = []
all_label = []
for label in labels:
    print(label)
    waves = [f for f in os.listdir(train_audio_path + '/' + label) if f.endswith('.wav')]
    for wav in waves:
        samples, sample_rate = librosa.load(train_audio_path + '/' + label + '/' + wav)
        samples = librosa.resample(samples, sample_rate, 8000)
        if(len(samples) == 8000):
            all_wave.append(samples)
            all_label.append(label)

bed
bird
cat
dog
down
eight
five
four
go
```

```
In [32]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y=le.fit_transform(all_label)
classes= list(le.classes_)
```

```
In [33]: from keras.utils import np_utils
y=np_utils.to_categorical(y, num_classes=len(labels))
```

```
In [34]: all_wave = np.array(all_wave).reshape(-1,8000,1)
```

```
In [35]: from sklearn.model_selection import train_test_split
x_tr, x_val, y_tr, y_val = train_test_split(np.array(all_wave),np.array(y))
```

```
In [42]: from keras.layers import Dense, Dropout, Flatten, Conv1D, Input, MaxPooling1D
from keras.models import Model
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras import backend as K
K.clear_session()

inputs = Input(shape=(8000,1))
```

```
#First Conv1D layer
conv = Conv1D(8, 13, padding='valid', activation='relu', strides=1)(inputs)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Second Conv1D layer
conv = Conv1D(16, 11, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Third Conv1D layer
conv = Conv1D(32, 9, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Fourth Conv1D layer
conv = Conv1D(64, 7, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Flatten layer
conv = Flatten()(conv)

#Dense Layer 1
conv = Dense(256, activation='relu')(conv)
conv = Dropout(0.3)(conv)

#Dense Layer 2
conv = Dense(128, activation='relu')(conv)
conv = Dropout(0.3)(conv)

outputs = Dense(len(labels), activation='softmax')(conv)

model = Model(inputs, outputs)
model.summary()
```

```
In [43]: model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=[ 'ac'])

In [44]: es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10,
mc = ModelCheckpoint('best_model.hdf5', monitor='val_acc', verbose=1, save_)

In [45]: history=model.fit(x_tr, y_tr ,epochs=30, callbacks=[es,mc], batch_size=32,
```

Epoch 1/30  
1047/1047 [=====] - 190s 179ms/step - loss: 2.92  
18 - accuracy: 0.1606 - val\_loss: 2.1934 - val\_accuracy: 0.3624  
WARNING:tensorflow:Can save best model only with val\_acc available, skipping.  
Epoch 2/30  
1047/1047 [=====] - 187s 179ms/step - loss: 1.77  
27 - accuracy: 0.4425 - val\_loss: 1.1613 - val\_accuracy: 0.6352  
WARNING:tensorflow:Can save best model only with val\_acc available, skipping.  
Epoch 3/30  
1047/1047 [=====] - 189s 180ms/step - loss: 1.25  
61 - accuracy: 0.5911 - val\_loss: 0.8985 - val\_accuracy: 0.7091  
WARNING:tensorflow:Can save best model only with val\_acc available, skipping.  
Epoch 4/30  
1047/1047 [=====] - 190s 182ms/step - loss: 0.99  
49 - accuracy: 0.6684 - val\_loss: 0.6496 - val\_accuracy: 0.7849  
WARNING:tensorflow:Can save best model only with val\_acc available, skipping

```
In [48]: def predict(audio):
    prob=model.predict(audio.reshape(1,8000,1))
    index=np.argmax(prob[0])
    return classes[index]
```

```
In [49]: import random
index=random.randint(0,len(x_val)-1)
samples=x_val[index].ravel()
print("Audio:",classes[np.argmax(y_val[index])])
ipd.Audio(samples, rate=8000)
```

Audio: six

Out[49]:



Output:

```
In [54]: print("Text:",predict(samples))
```

```
Text: six
```

```
In [59]: import sounddevice as sd
import soundfile as sf

samplerate = 16000
duration = 1 # seconds
filename = 'yes.wav'
print("start")
mydata = sd.rec(int(samplerate * duration), samplerate=samplerate,
    channels=1, blocking=True)
print("end")
sd.wait()
sf.write(filename, mydata, samplerate)
```

```
start
end
```

```
In [76]: filepath='input'
```

```
In [75]: samples, sample_rate = librosa.load(filepath + '/happy/2.wav', sr = 16000)
samples = librosa.resample(samples, sample_rate, 8000)
ipd.Audio(samples,rate=8000)
```

```
Out[75]:
```



```
In [77]: predict(samples)
```

```
Out[77]: 'happy'
```

**Conclusion :** As we can see in above output we have passed the audio file in which a person is saying 'happy' word and our model is predicting the output as 'happy' accurately.

## Practical 9

**Aim :** To predict the perfect emoji for the given text

**Theory :** We are first converting a training emoji dataset and testing emoji dataset to dataframe then we are passing all the training data to RNN model recurrent neural network will help to sustain the information and it performs very well in classification of emoji on emoji dataset.

**Input:**

```
▶ import numpy as np
import pandas as pd
import emoji

from keras.models import Sequential
from keras.layers import Dense, Input, Dropout, SimpleRNN, LSTM, Activation
from keras.utils import np_utils

import matplotlib.pyplot as plt
```

```
# train_emoji data and test emoji data is Attached in the repo
train = pd.read_csv('train_emoji.csv', header=None)
test = pd.read_csv('test_emoji.csv', header=None)
```

```
# checking the data by printing first 5 entries
train.head()
```

|   | 0                               | 1 | 2   | 3   |
|---|---------------------------------|---|-----|-----|
| 0 | never talk to me again          | 3 | NaN | NaN |
| 1 | I am proud of your achievements | 2 | NaN | NaN |
| 2 | It is the worst day in my life  | 3 | NaN | NaN |
| 3 | Miss you so much                | 0 | NaN | [0] |
| 4 | food is life                    | 4 | NaN | NaN |

```
# checking the data by printing first 5 entries
test.head()
```

|   | 0                           | 1 |
|---|-----------------------------|---|
| 0 | I want to eat\rt            | 4 |
| 1 | he did not answer\rt        | 3 |
| 2 | he got a raise\rt           | 2 |
| 3 | she got me a present\rt     | 0 |
| 4 | ha ha ha it was so funny\rt | 2 |

```
# Creating the Dictionary of some emoji's with key a number and value is emoji
emoji_dict = { 0 : ":heart:", 1 : ":baseball:", 2:"smile:", 3 : ":disappointed:", 4 : ":fork_and_knife:"}
```

```
# printing the emoji icon by emojiifying each emoji
for ix in emoji_dict.keys():
    print(ix,end=" ")
    print (emoji.emojize(emoji_dict[ix], use_aliases=True))
```

```
0 ❤
1 ⚾
2 😊
3 😔
4 ||
```

```
# Creating the training and testing data

X_train = train[0]
Y_train = train[1]

X_test = test[0]
Y_test = test[1]

print(X_train.shape, Y_train.shape, X_test.shape, Y_test.shape)
print("-----")
print(X_train[0],Y_train[0])

(132,) (132,) (56,) (56,)
-----
never talk to me again 3
```

```
# Splitting the train data from sentences to list of words
for ix in range(X_train.shape[0]):
    X_train[ix] = X_train[ix].split()

# Splitting the test data from sentences to list of words
for ix in range(X_test.shape[0]):
    X_test[ix] = X_test[ix].split()

# Converting the labels into categorical Form
Y_train = np_utils.to_categorical(Y_train)
```

```
print(X_train[0],Y_train[0])
type(X_train)

['never', 'talk', 'to', 'me', 'again'] [0. 0. 0. 1. 0.]
pandas.core.series.Series
```

```
# To check what's the maximum length exist in the training data
np.unique(np.array([len(ix) for ix in X_train]), return_counts=True)

(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10]),
 array([ 4,  5, 26, 35, 20, 21, 11,  5,  1,  4], dtype=int64))
```

```
# Creating the Embedding dictionary with key = word and value = list of words
embeddings_index = {}
file = open('glove.6B.50d.txt', encoding="utf16")
file = open('glove.6B.50d.txt', errors="ignore")

try:
    for line in file:
        values = line.split()
    #    print(values)
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs
    file.close()
except:
    file.__next__()
```

```
# Checking the length of each vector
embeddings_index["i"].shape
```

```
(50,)
```

```
# Using Scipy library to import the embedding matrix
from scipy import spatial
```

```
# Checking the cosine similarity of happy and sad
spatial.distance.cosine(embeddings_index["happy"], embeddings_index["sad"])
```

```
0.31093674898147583
```

```
# Checking the cosine similarity of india and delhi
spatial.distance.cosine(embeddings_index["india"], embeddings_index["delhi"])
```

```
0.18572336435317993
```

```
# Checking the cosine similarity of france and paris
spatial.distance.cosine(embeddings_index["france"], embeddings_index["paris"])
```

```
0.19746702909469604
```

```
# Filling the Embedding Matrix
```

```
embedding_matrix_train = np.zeros((X_train.shape[0], 10, 50))
embedding_matrix_test = np.zeros((X_test.shape[0], 10, 50))
try:
    for ix in range(X_train.shape[0]):
        for ij in range(len(X_train[ix])):
            embedding_matrix_train[ix][ij] = embeddings_index[X_train[ix][ij].lower()]

    for ix in range(X_test.shape[0]):
        for ij in range(len(X_test[ix])):
            embedding_matrix_test[ix][ij] = embeddings_index[X_test[ix][ij].lower()]
except:
    X_train.shape[0]
```

```

print(embedding_matrix_train.shape, embedding_matrix_test.shape)

(132, 10, 50) (56, 10, 50)

model = Sequential()
model.add(SimpleRNN(64, input_shape=(10,50), return_sequences=True))
model.add(Dropout(0.5))
model.add(SimpleRNN(64, return_sequences=False))
model.add(Dropout(0.5))
model.add(Dense(5))
model.add(Activation('softmax'))

```

```

# Setting Loss, Optimizer of the Model

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

```

```

hist = model.fit(embedding_matrix_train,Y_train,
                  epochs = 50, batch_size=32,shuffle=True
                 )

5/5 [=====] - 0s 8ms/step - loss: 1.4612
Epoch 22/50
5/5 [=====] - 0s 9ms/step - loss: 1.5177
Epoch 23/50
5/5 [=====] - 0s 7ms/step - loss: 1.4679
Epoch 24/50
5/5 [=====] - 0s 8ms/step - loss: 1.5209
Epoch 25/50
5/5 [=====] - 0s 8ms/step - loss: 1.4869
Epoch 26/50
5/5 [=====] - 0s 7ms/step - loss: 1.4846
Epoch 27/50
5/5 [=====] - 0s 7ms/step - loss: 1.4558
Epoch 28/50
5/5 [=====] - 0s 8ms/step - loss: 1.5145
Epoch 29/50
5/5 [=====] - 0s 8ms/step - loss: 1.4911
Epoch 30/50
5/5 [=====] - 0s 7ms/step - loss: 1.4886

# printing the sentences with the predicted emoji and the labelled emoji

for ix in range(embedding_matrix_test.shape[0]):

    if pred[ix] != Y_test[ix]:
        print(ix)
        print(test[0][ix],end=" ")
        print(emoji.emojize(emoji_dict[pred[ix]], use_aliases=True),end=" ")
        print(emoji.emojize(emoji_dict[Y_test[ix]], use_aliases=True))

```

**Output :**

```

['I', 'want', 'to', 'eat'] 😊 ||
1
['he', 'did', 'not', 'answer'] 😊 😞
3
['she', 'got', 'me', 'a', 'present'] 😊 ❤
5
['he', 'is', 'a', 'good', 'friend'] 😊 ❤
6
['I', 'am', 'upset'] 😊 ❤
7
['We', 'had', 'such', 'a', 'lovely', 'dinner', 'tonight'] 😊 ❤
8
['where', 'is', 'the', 'food'] 😊 ||
10
['where', 'is', 'the', 'ball'] 😊 ⚾
11
['work', 'is', 'hard'] 😊 😞
12
['This', 'girl', 'is', 'messing', 'with', 'me'] 😊 😞
14
['Let', 'us', 'go', 'play', 'baseball'] 😊 ⚾
15
['This', 'stupid', 'grader', 'is', 'not', 'working'] 😊 😞
16
['work', 'is', 'horrible'] 😊 😞
18
['stop', 'messing', 'around'] 😊 😞
19
['any', 'suggestions', 'for', 'dinner'] 😊 ||
20
['I', 'love', 'taking', 'breaks'] 😊 ❤
22
['I', 'boiled', 'rice'] 😊 ||
23
['she', 'is', 'a', 'bully'] 😊 😞
24
['Why', 'are', 'you', 'feeling', 'bad'] 😊 😞
25
['I', 'am', 'upset'] 😊 😞
26
['I', 'worked', 'during', 'my', 'birthday'] 😊 😞
27
['My', 'grandmother', 'is', 'the', 'love', 'of', 'my', 'life'] 😊 ❤
29
['valentine', 'day', 'is', 'near'] 😊 ❤
30
['I', 'miss', 'you', 'so', 'much'] 😊 ❤

```

**Conclusion :** As we can see the model is perfectly classifying the perfect emoji for the respective text in the first example of 'i want to eat' the model is accurately showing the emoji of spoon and fork as it is classifying the eat word accurately.

# Practical 10

**Aim :** To remove the noise from the image using autoencoders.

**Theory :** We are first taking the perfect image and adding the noise to that image using CV2 library and after that we have created autoencoders by using the CNN model and we will remove the noise from the data which is having noise and create a new fresh image which not contain noise.

## **Input :**

```
[28] img , img_noise = get_images()
    ↗
    9% | [ 339/3642 [00:12<02:06, 26.02it/s]
    9% | [ 342/3642 [00:13<02:05, 26.34it/s]
    9% | [ 345/3642 [00:13<02:04, 26.49it/s]
10% | [ 348/3642 [00:13<02:04, 26.40it/s]
10% | [ 351/3642 [00:13<02:02, 26.82it/s]
10% | [ 354/3642 [00:13<02:05, 26.26it/s]
10% | [ 357/3642 [00:13<02:04, 26.38it/s]
10% | [ 360/3642 [00:13<02:04, 26.34it/s]
10% | [ 363/3642 [00:13<02:04, 26.33it/s]
10% | [ 366/3642 [00:13<02:03, 26.50it/s]
10% | [ 369/3642 [00:14<02:03, 26.40it/s]
10% | [ 372/3642 [00:14<02:07, 25.71it/s]
10% | [ 375/3642 [00:14<02:04, 26.16it/s]
```

```
import numpy as np
from sklearn.model_selection import train_test_split
#x_train,x_test = train_test_split( img, test_size=0.33, random_state=42)

    ↗
x_train_noise,y_test_noise , x_train_img,y_test_img = train_test_split(img_noise, img, test_size=0.33, random_state=42)
```

```
import numpy as np
x_train_noise = np.array(x_train_noise)
x_train_img = np.array(x_train_img)

y_test_noise = np.array(y_test_noise)
y_test_img = np.array(y_test_img)

print(x_train_noise.shape)
print(y_test_img.shape)

(335, 80, 80, 3)
(165, 80, 80, 3)
```

```
import keras
from keras.layers import *
from keras.layers import *
from keras.models import *

Input_img = Input(shape=(80, 80, 3))

#encoding architecture
x1 = Conv2D(64, (3, 3), activation='relu', padding='same')(Input_img)
x1 = MaxPool2D( (2, 2), padding='same') Loading...
x2 = Conv2D(32, (3, 3), activation='relu', padding='same')(x1)
x2 = MaxPool2D( (2, 2), padding='same')(x2)
x3 = Conv2D(16, (3, 3), activation='relu', padding='same')(x2)
x3= MaxPool2D( (2, 2), padding='same')(x3)
x4= Conv2D(16, (3, 3), activation='relu', padding='same')(x3)
encoded = MaxPool2D( (2, 2), padding='same')(x3)

#decoder
x = Conv2DTranspose(16, (3, 3), activation='relu',strides=2,padding='same')(encoded)
x = Conv2DTranspose(16, (3, 3), activation='relu',strides=2,padding='same')(x)
x = Conv2DTranspose(32, (3, 3), activation='relu',strides=2,padding='same')(x)
x = Conv2DTranspose(64, (3, 3), activation='relu',strides=2,padding='same')(x)
decoded = Conv2DTranspose(3, (3, 3),padding='same', activation='sigmoid')(x)

autoencoder = Model(Input_img, decoded)
adam = keras.optimizers.Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999, amsgrad=False)
autoencoder.compile(optimizer=adam, loss='mse')
```

```
autoencoder.summary()
```

Model: "model\_2"

| Layer (type)                         | Output Shape       | Param # |
|--------------------------------------|--------------------|---------|
| input_2 (InputLayer)                 | (None, 80, 80, 3)  | 0       |
| conv2d_5 (Conv2D)                    | (None, 80, 80, 64) | 1792    |
| max_pooling2d_5 (MaxPooling2D)       | (None, 40, 40, 64) | 0       |
| conv2d_6 (Conv2D)                    | (None, 40, 40, 32) | 18464   |
| max_pooling2d_6 (MaxPooling2D)       | (None, 20, 20, 32) | 0       |
| conv2d_7 (Conv2D)                    | (None, 20, 20, 16) | 4624    |
| max_pooling2d_7 (MaxPooling2D)       | (None, 10, 10, 16) | 0       |
| max_pooling2d_8 (MaxPooling2D)       | (None, 5, 5, 16)   | 0       |
| conv2d_transpose_6 (Conv2DTranspose) | (None, 10, 10, 16) | 2320    |
| conv2d_transpose_7 (Conv2DTranspose) | (None, 20, 20, 16) | 2320    |

```
callback = keras.callbacks.callbacks.EarlyStopping(monitor='loss', min_delta=0, patience=4, verbose=0, mode='auto')
autoencoder.fit(x_train_noisy, x_train_img,
                 epochs=1000,
                 batch_size=10,
                 shuffle=True, callbacks=[Loading...],
                 validation_data=(y_test_noisy,y_test_img))

Epoch 21/1000
335/335 [=====] - 0s 1ms/step - loss: 0.0175 - val_loss: 0.0182
Epoch 22/1000
335/335 [=====] - 0s 1ms/step - loss: 0.0175 - val_loss: 0.0183
Epoch 23/1000
335/335 [=====] - 0s 1ms/step - loss: 0.0175 - val_loss: 0.0183
Epoch 24/1000
335/335 [=====] - 0s 1ms/step - loss: 0.0175 - val_loss: 0.0182
```

```

def compare_outputs(y_train_noise, decoded_imgs=None, n=10):
    plt.figure(figsize=(22, 5))
    for i in range(n):
        ax = plt.subplot(2, n, i+1)
        plt.imshow(y_train_noise[i].reshape(80,80,3))

        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

        if decoded_imgs is not None:
            ax = plt.subplot(2, n, i+1 +n)
            plt.imshow(decoded_imgs[i].reshape(80,80,3))

            ax.get_xaxis().set_visible(False)
            ax.get_yaxis().set_visible(False)
    plt.show()

decoded_imgs = autoencoder.predict(y_train_noise)
print('Upper row: Input image provided \nBottom row: Decoded output generated')

compare_outputs(y_train_noise, decoded_imgs)

```

**Output :**



**Conclusion :** As we can see the model is accurately getting rid of all the noise giving exact output.

## Practical 11

**Aim :** To detect the object from image accurately.

**Theory :** We are first going to provide some utility packages in our code and use them for the object detection task, then we are going to create some classes of the objection aeroplane, car, etc and we will pass it all to our model and see for the output accuracy.

**Input :**

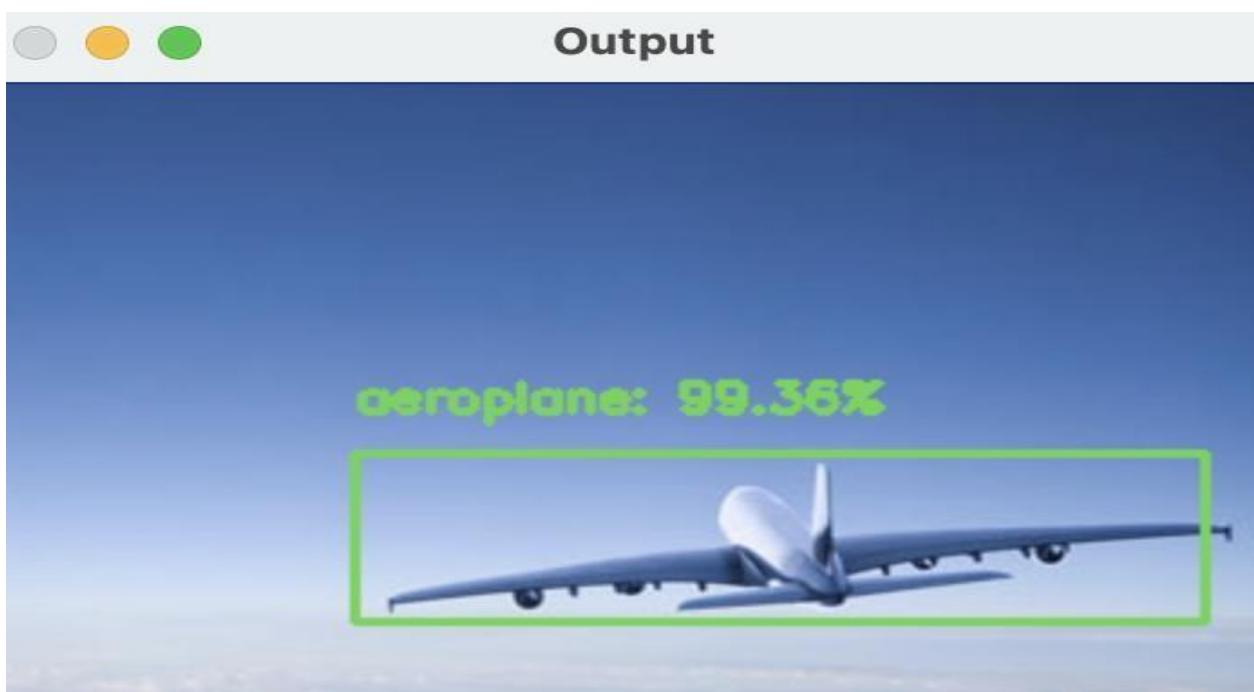
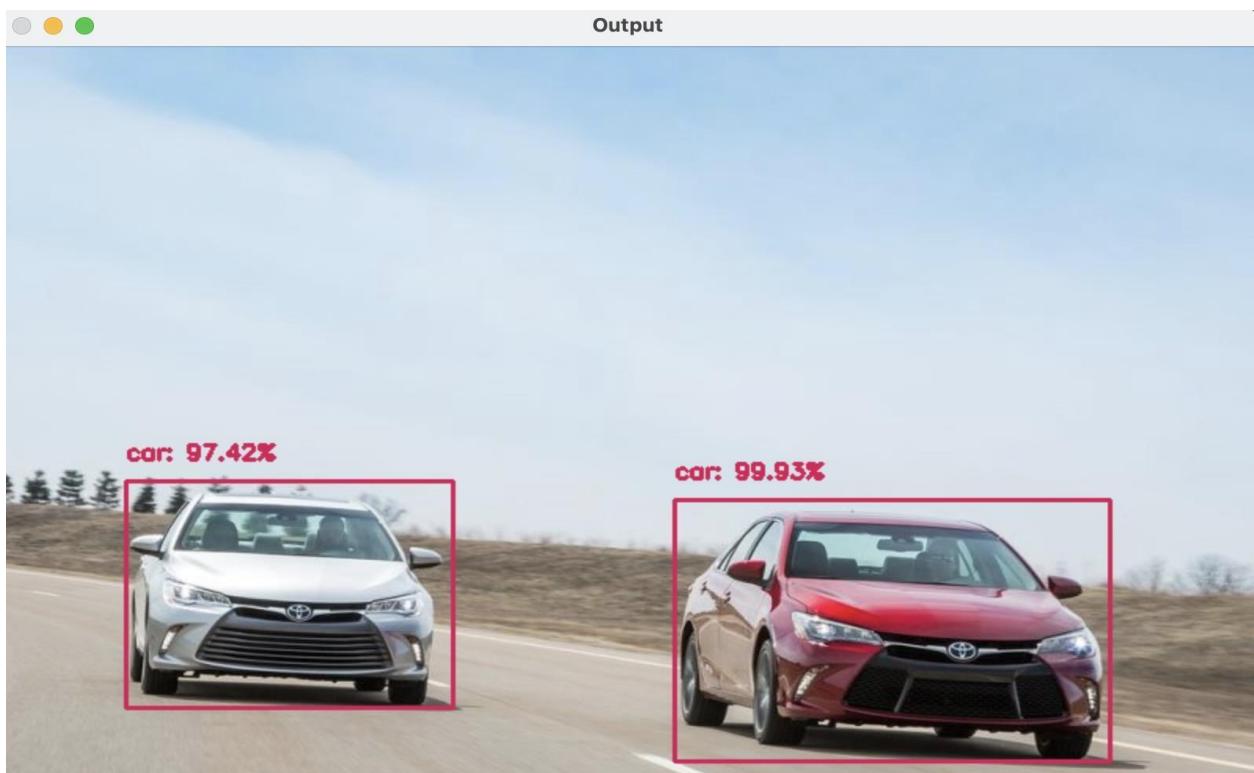
```
1 # import the necessary packages
2 import numpy as np
3 import argparse
4 import cv2
5
6 # construct the argument parse and parse the arguments
7 ap = argparse.ArgumentParser()
8 ap.add_argument("-i", "--image", required=True,
9                 help="path to input image")
10 ap.add_argument("-p", "--prototxt", required=True,
11                  help="path to Caffe 'deploy' prototxt file")
12 ap.add_argument("-m", "--model", required=True,
13                  help="path to Caffe pre-trained model")
14 ap.add_argument("-c", "--confidence", type=float, default=0.2,
15                  help="minimum probability to filter weak detections")
16 args = vars(ap.parse_args())
17
18 # initialize the list of class labels MobileNet SSD was trained to
19 # detect, then generate a set of bounding box colors for each class
20 CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
21             "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
22             "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
23             "sofa", "train", "tvmonitor"]
24 COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

```

26 # load our serialized model from disk
27 print("[INFO] loading model...")
28 net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
29
30 # load the input image and construct an input blob for the image
31 # by resizing to a fixed 300x300 pixels and then normalizing it
32 # (note: normalization is done via the authors of the MobileNet SSD
33 # implementation)
34 image = cv2.imread(args["image"])
35 (h, w) = image.shape[:2]
36 blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843, (300, 300), 127.5)
37
38 # pass the blob through the network and obtain the detections and
39 # predictions
40 print("[INFO] computing object detections...")
41 net.setInput(blob)
42 detections = net.forward()
43
44 # loop over the detections
45 for i in np.arange(0, detections.shape[2]):
46     # extract the confidence (i.e., probability) associated with the
47     # prediction
48     confidence = detections[0, 0, i, 2]
49
50     # filter out weak detections by ensuring the `confidence` is
51     # greater than the minimum confidence
52     if confidence > args["confidence"]:
53         # extract the index of the class label from the `detections`,
54         # then compute the (x, y)-coordinates of the bounding box for
55         # the object
56         idx = int(detections[0, 0, i, 1])
57         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
58         (startX, startY, endX, endY) = box.astype("int")
59
60         # display the prediction
61         label = "{}: {:.2f}%".format(CLASSES[idx], confidence * 100)
62         print("[INFO] {}".format(label))
63         cv2.rectangle(image, (startX, startY), (endX, endY),
64                       COLORS[idx], 2)
65         y = startY - 15 if startY - 15 > 15 else startY + 15
66         cv2.putText(image, label, (startX, y),
67                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
68
69 # show the output image
70 cv2.imshow("Output", image)
71 cv2.waitKey(0)

```

**Output :**



**Conclusion :** As we can see the accuracy of the model while detecting the object is very nice and it is working fantastic.

## Practical 12

**Aim :** To classify the disease type by using CNN model

**Theory :** We first took the dataset of Brain tumor and we applied encoding on it and printed various parts of the brain on which brain tumor can happen. Then we are training our CNN model of sequential layers and will pass all the data in it, it seems that it will perform perfectly and give accurate results.

**Input:**

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, Conv3D, BatchNormalization, Activation
from keras import backend as K
import os
from PIL import Image
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import pandas as pd

classes = os.listdir('\Brain-Tumor-Classification-Data\Training')

enc = OneHotEncoder()
enc.fit([[0], [1], [2], [3]])
def names(number):
    if(number == 0):
        return classes[0]
    elif(number == 1):
        return classes[1]
    elif(number == 2):
        return classes[2]
    elif(number == 3):
        return classes[3]
```

```
trainData = []
trainLabel = []
dim = (150, 150)
trainPath = "\Brain-Tumor-Classification-DataSet\Training"
index = 0
for dir in os.listdir(trainPath):
    filePaths = []
    subDir = os.path.join(trainPath, dir)
    for file in os.listdir(subDir):
        imgFullPath = os.path.join(subDir, file)
        filePaths.append(imgFullPath)
        img = Image.open(imgFullPath)
        x = img.resize(dim)
        x = np.array(x)
        trainData.append(np.array(x))
        trainLabel.append(enc.transform([[index]]).toarray())
    print(names(index))
    print(str(dir))
    index += 1

trainData = np.array(trainData)
trainLabel = np.array(trainLabel).reshape(2870, 4)
print(trainData.shape)
print(trainLabel.shape)

glioma_tumor
glioma_tumor
meningioma_tumor
meningioma_tumor
no_tumor
no_tumor
pituitary_tumor
pituitary_tumor
(2870, 150, 150, 3)
(2870, 4)
```

```
testData = []
testLabel = []
dim = (150, 150)
testPath = "\Brain-Tumor-Classification-Data\Testing"
index = 0
for dir in os.listdir(testPath):
    filePaths = []
    subDir = os.path.join(testPath, dir)
    for file in os.listdir(subDir):
        imgFullPath = os.path.join(subDir, file)
        filePaths.append(imgFullPath)
        img = Image.open(imgFullPath)
        x = img.resize(dim)
        x = np.array(x)
        testData.append(np.array(x))
        testLabel.append(enc.transform([[index]]).toarray())
    print(names(index))
    print(str(dir))
    index += 1
testData = np.array(testData)
testLabel = np.array(testLabel).reshape(394, 4)
print(testData.shape)
print(testLabel.shape)
```

```
glioma_tumor
glioma_tumor
meningioma_tumor
meningioma_tumor
no_tumor
no_tumor
pituitary_tumor
pituitary_tumor
(394, 150, 150, 3)
(394, 4)
```

```

model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape=(150,150,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(4))
model.add(Activation('softmax'))

model.compile(loss = "categorical_crossentropy", optimizer='adam')
print(model.summary())

```

Model: "sequential"

| Layer (type)                 | Output Shape         | Param # |
|------------------------------|----------------------|---------|
| <hr/>                        |                      |         |
| conv2d (Conv2D)              | (None, 148, 148, 32) | 896     |
| activation (Activation)      | (None, 148, 148, 32) | 0       |
| max_pooling2d (MaxPooling2D) | (None, 74, 74, 32)   | 0       |
| conv2d_1 (Conv2D)            | (None, 72, 72, 32)   | 9248    |
| activation_1 (Activation)    | (None, 72, 72, 32)   | 0       |
| max_pooling2d_1 (MaxPooling2 | (None, 36, 36, 32)   | 0       |
| flatten (Flatten)            | (None, 41472)        | 0       |
| dense (Dense)                | (None, 32)           | 1327136 |
| activation_2 (Activation)    | (None, 32)           | 0       |
| dropout (Dropout)            | (None, 32)           | 0       |
| dense_1 (Dense)              | (None, 4)            | 132     |
| activation_3 (Activation)    | (None, 4)            | 0       |
| <hr/>                        |                      |         |
| Total params:                | 1,337,412            |         |
| Trainable params:            | 1,337,412            |         |
| Non-trainable params:        | 0                    |         |

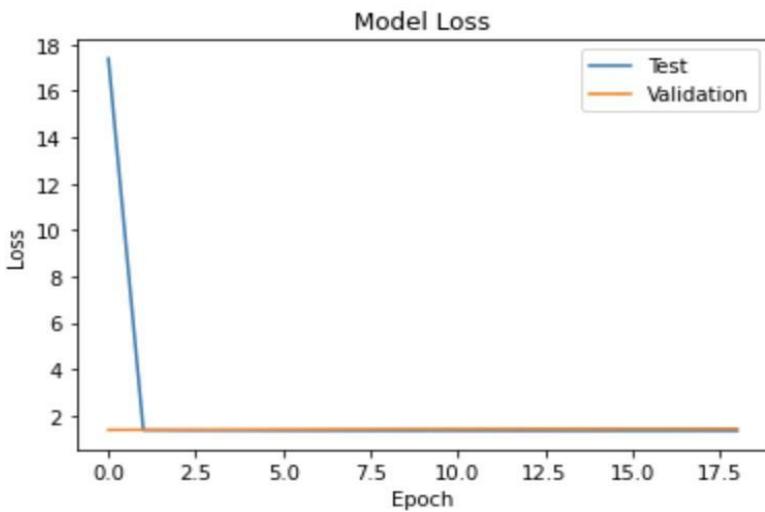
```

history = model.fit(trainData, trainLabel,batch_size = 32, epochs = 19, verbose=1,validation_data=(testData, testLabel))

Epoch 1/19
90/90 [=====] - 39s 424ms/step - loss: 60.9387 - val_loss: 1.3870
Epoch 2/19
90/90 [=====] - 38s 427ms/step - loss: 1.3772 - val_loss: 1.3899
Epoch 3/19
90/90 [=====] - 37s 414ms/step - loss: 1.3677 - val_loss: 1.3940

```

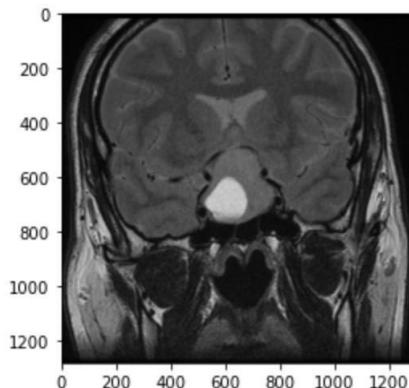
```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Test', 'Validation'], loc='upper right')
plt.show()
```



#### Output:

```
img = Image.open('pituitary_tumor\\image.jpg')
x = np.array(img.resize(dim))
x = x.reshape(1,150,150,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is ' + names(classification))
```

28.728872537612915% Confidence This Is pituitary\_tumor



**Conclusion :** As we can see the CNN model is perfectly classifying the area perfectly in which tumor is detected in this case it is a pituitary type of tumor.