Keraleeya Samajam(Regd.) Dombivli's

MODEL COLLEGE

Re-Accredited Grade "A" by NAAC

Kanchan Goan Village, Khambalpada, Thakurli East – 421201
Contact No – 7045682157, 7045682158. www.model-college.edu.in

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr. /Miss _____

*Studying in Class*_____*Seat No.* _____

*Has completed the prescribed practicals in the subject*_____

*During the academic year*_____

Date : _____

External Examiner

Internal Examiner
M.Sc. Information Technology

Practical No	Title	Date	Signature
1	**Write the following programs for Blockchain in Python:** (I) A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it (II) A transaction class to send and receive money and test it.	1/4/2023	
2	**Write the following programs for Blockchain in Python:** (I). Create multiple transactions and display them. (II). Create a blockchain, a genesis block and execute it.	1/4/2023	
3	**Write the following programs for Blockchain in Python:** (I). Create a mining function and test it. (II). Add blocks to the miner and dump the blockchain.	1/4/2023	
4	**Implement and demonstrate the use of the following in Solidity:** 4.1.1: Variable 4.1.2: State Variable 4.1.3: Global Variable 4.2: Operators 4.3.1: Decision Making 4.3.2: Decision Making: if-else 4.4: String	29/4/2023	
5	**Implement and demonstrate the use of the following in Solidity:** 5.1: Arrays 5.2: Structs 5.3.1: Mapping 5.3.2: Mapping String	29/4/2023	
6	**Implement and demonstrate the use of the following in Solidity:** 6.1: Functions 6.2.1: View Function 6.2.2: function external 6.3: Function Overloading 6.4: Mathematical Function 6.5: Cryptographic Functions	29/4/2023	
7	**Implement and demonstrate the use of the following in Solidity:** 7.1: Contract 7.2: Inheritance 7.3: constructors	29/4/2023	

Practical No	Title	Date	Signature
8	**Implement and demonstrate the use of the following in Solidity:** 8.1: require statement 8.2.1: assert statement.1 8.2.2: assert statement.2 8.3: revert statement	29/4/2023	

Practical No: 1

Aim: Write the following programs for Blockchain in Python:

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

(II) A transaction class to send and receive money and test it.

Program 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

#pip install pycryptodome

import Crypto

import Crypto.Random

from Crypto.Hash import SHA

from Crypto.PublicKey import RSA

from Crypto.Signature import PKCS1_v1_5

class Client:

    def __init__(self):

        random = Crypto.Random.new().read

        self._private_key = RSA.generate(1024, random)

        self._public_key = self._private_key.public_key()

        self._signer = PKCS1_v1_5.new(self._private_key)
```

```
    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format = 'DER')).decode('ascii')
class Transaction:
    def __init__(self,sender,recipient,value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()
    def to_dict(self):
        if self.sender =="Gensis":
            identity = "Gensis"
        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')
Abhi = Client()
Raj = Client()
t = Transaction(Abhi, Raj.identity, 5.0)
signature = t.sign_transaction()
print(signature)
```

Program Output:

```
print(signature)
```

```
9c7e0de9f04a83aca758c71634c7e9940278f91a0ba58da46b98f43bbbbcc70c063d0132733acf0c8151536fb90702cb5b81aa800edb26d94e8c84d7923fdde
08b076bef54e22f30b777870e9e50628cfbcca86fcbb3b0eae8920be20f955ec35cdef81b0f39a17d4a03ab896dea6f835085eec6613fe07c4d479675ba0b50
bb
```

Practical No: 2

Aim: Write the following programs for Blockchain in Python:

(I). Create multiple transactions and display them.

(II). Create a blockchain, a genesis block and execute it.

Program 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 = []
Abhi = Client()
Raj = Client()
Ashwin = Client()
Sidd = Client()
t1 = Transaction(
   Abhi,
   Raj.identity,
   15.0
)
t1.sign_transaction()
transactions.append(t1)

t2 = Transaction(
```

```
    Abhi,
    Ashwin.identity,
    6.0
)
t2.sign_transaction()
transactions.append(t2)

t3 = Transaction(
    Raj,
    Sidd.identity,
    2.0
)
t3.sign_transaction()
transactions.append(t3)

t4 = Transaction(
    Ashwin,
    Raj.identity,
    4.0
)
t4.sign_transaction()
transactions.append(t4)

t5 = Transaction(
    Sidd,
    Ashwin.identity,
    7.0
)
t5.sign_transaction()
transactions.append(t5)
```

```
t6 = Transaction(
   Raj,
   Ashwin.identity,
   3.0
)
t6.sign_transaction()
transactions.append(t6)

t7 = Transaction(
   Ashwin,
   Abhi.identity,
   8.0
)
t7.sign_transaction()
transactions.append(t7)

t8 = Transaction(
   Ashwin,
   Raj.identity,
   1.0
)
t8.sign_transaction()
transactions.append(t8)

t9 = Transaction(
   Sidd,
   Abhi.identity,
   5.0
)
t9.sign_transaction()
```

```
transactions.append(t9)

t10 = Transaction(
  Sidd,
  Raj.identity,
  3.0
)
t10.sign_transaction()
transactions.append(t10)

for transaction in transactions:
   display_transaction(transaction)
   print ('--------------')
class Block:
   def __init__(self):
      self.verified_transactions = []
      self.previous_block_hash = ""
      self.Nonce = ""
last_block_hash = ""
Dinesh = Client()
t0 = Transaction (
   "Genesis",
   Dinesh.identity,
   500.0
)
block0 = Block()
block0.previous_block_hash = None
Nonce = None
block0.verified_transactions.append (t0)
digest = hash (block0)
```

```
last_block_hash = digest
TPCoins = []
def dump_blockchain (self):
    print ("Number of blocks in the chain: " + str(len (self)))
    for x in range (len(TPCoins)):
        block_temp = TPCoins[x]
        print ("block # " + str(x))
        for transaction in block_temp.verified_transactions:
            display_transaction(transaction)
            print ('------')
        print ('====================================')
TPCoins.append (block0)
dump_blockchain(TPCoins)
```

Program Output:

```
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59aeda1
1fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862641
6125f83f4c0b0f6c6869e1cb4734243b878b67bb277c4fd47ece61ec4cdbdf51f09f39530203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d756246536389525581504c7eb2c065cb304317fecd6da49361
285417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e
303bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
value: 15.0
-----
time: 2023-06-22 01:26:04.293351
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59aeda1
1fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862641
6125f83f4c0b0f6c6869e1cb4734243b878b67bb277c4fd47ece61ec4cdbdf51f09f39530203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b
4dd875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f
```

```
value: 6.0
-----
time: 2023-06-22 01:26:04.293351
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d7562465363895255815 04c7eb2c065cb304317fecd6da49361285
417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e303
bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c5843d8edfbb3077827e51daad6c84cb86a3e5ca80f7a43e0a4fb218
c766d732971599bb495410301fd04a3bbb15d8643c74c6635f966e6629e7be9c065cc5b2499a081e361bf642bbb23e046e35542cd9e0ca14233a7875acb1b
27ed08547e98ef189c1fee56715ee6033322d85443501d1d593d40c0bbf4cef1eb8052bbd6d0203010001
-----
value: 2.0
-----
time: 2023-06-22 01:26:04.293351
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b4dd
875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f130
```

```
285417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e
303bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
value: 4.0
-----
time: 2023-06-22 01:26:04.293351
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c5843d8edfbb3077827e51daad6c84cb86a3e5ca80f7a43e0a4fb218c76
6d732971599bb495410301fd04a3bbb15d8643c74c6635f966e6629e7be9c065cc5b2499a081e361bf642bbb23e046e35542cd9e0ca14233a7875acb1b27e
d08547e98ef189c1fee56715ee6033322d85443501d1d593d40c0bbf4cef1eb8052bbd6d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b
4dd875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f
1305d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
value: 7.0
-----
time: 2023-06-22 01:26:04.293351
-----
```

```
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d7562465363895255815 04c7eb2c065cb304317fecd6da49361285
417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e303
bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b
4dd875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f
1305d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
value: 3.0
-----
time: 2023-06-22 01:26:04.309004
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b4dd
875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f130
5d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59ae
da11fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862
```

```
value: 8.0
-----
time: 2023-06-22 01:26:04.309004
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b4dd
875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f130
5d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d7562465363895255815 04c7eb2c065cb304317fecd6da49361
285417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e
303bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
value: 1.0
-----
time: 2023-06-22 01:26:04.309004
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c5843d8edfbb3077827e51daad6c84cb86a3e5ca80f7a43e0a4fb218c76
6d732971599bb495410301fd04a3bbb15d8643c74c6635f966e6629e7be9c065cc5b2499a081e361bf642bbb23e046e35542cd9e0ca14233a7875acb1b27e
```

```
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59ae
da11fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862
6416125f83f4c0b0f6c6869e1cb4734243b878b67bb277c4fd47ece61ec4cdbdf51f09f39530203010001
-----
value: 5.0
-----
time: 2023-06-22 01:26:04.309004
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c5843d8edfbb3077827e51daad6c84cb86a3e5ca80f7a43e0a4fb218c76
6d732971599bb495410301fd04a3bbb15d8643c74c6635f966e6629e7be9c065cc5b2499a081e361bf642bbb23e046e35542cd9e0ca14233a7875acb1b27e
d08547e98ef189c1fee56715ee6033322d85443501d1d593d40c0bbf4cef1eb8052bbd6d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d7562465363895255815 04c7eb2c065cb304317fecd6da49361
285417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e
303bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
value: 3.0
-----
```

```
value: 3.0
-----
time: 2023-06-22 01:26:04.309004
-----
--------------
Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d03ff684e148983d98ae8c422917f19aa995489192549e72df0e479a
4481174965e80523f841a2cc107bbb18c09e3484db0794b96ebd8d9e3ef70ce798f96437a67002f74e2205655b250a29ace236bfbae9173f3b69d38fe9aee
df9d740928d329d1c07d7fea6490861d74553871829bad8b52533ec5c926de207d97683b89d0203010001
-----
value: 500.0
-----
time: 2023-06-22 01:26:04.620320
-----
------
====================================
```

Practical No: 3

Aim: Write the following programs for Blockchain in Python:

(I). Create a mining function and test it.

(II). Add blocks to the miner and dump the blockchain.

Program 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)
last_transaction_index = 0
block = Block()
for i in range(3):
    temp_transaction = transactions[last_transaction_index]
    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
block = Block()
```

```
for i in range(3):
    temp_transaction = transactions[last_transaction_index]
    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]
    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)
```

Program Output:

```
Number of blocks in the chain: 4
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d03ff684e148983d98ae8c422917f19aa995489192549e72df0e479a
4481174965e80523f841a2cc107bbb18c09e3484db0794b96ebd8d9e3ef70ce798f96437a67002f74e2205655b250a29ace236bfbae9173f3b69d38fe9aee
df9d740928d329d1c07d7fea6490861d74553871829bad8b52533ec5c926de207d97683b89d0203010001
-----
value: 500.0
-----
time: 2023-06-22 01:26:04.620320
-----
------
=====================================
block # 1
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59aeda1
1fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862641
6125f83f4c0b0f6c6869e1cb4734243b878b67bb277c4fd47ece61ec4cdbdf51f09f39530203010001
-----
```

```
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d7562465363895255815 04c7eb2c065cb304317fecd6da49361
285417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e
303bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
value: 15.0
-----
time: 2023-06-22 01:26:04.293351
-----
------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59aeda1
1fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862641
6125f83f4c0b0f6c6869e1cb4734243b878b67bb277c4fd47ece61ec4cdbdf51f09f39530203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b
4dd875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f
1305d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
value: 6.0
-----
```

```
-----
time: 2023-06-22 01:26:04.293351
-----
------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d7562465363895255815 04c7eb2c065cb304317fecd6da49361285
417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e303
bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c5843d8edfbb3077827e51daad6c84cb86a3e5ca80f7a43e0a4fb218
c766d732971599bb495410301fd04a3bbb15d8643c74c6635f966e6629e7be9c065cc5b2499a081e361bf642bbb23e046e35542cd9e0ca14233a7875acb1b
27ed08547e98ef189c1fee56715ee6033322d85443501d1d593d40c0bbf4cef1eb8052bbd6d0203010001
-----
value: 2.0
-----
time: 2023-06-22 01:26:04.293351
-----
------
=====================================
block # 2
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b4dd
```

```
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b4dd
875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f130
5d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d7562465363895255815 04c7eb2c065cb304317fecd6da49361
285417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e
303bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
value: 4.0
-----
time: 2023-06-22 01:26:04.293351
-----
------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c5843d8edfbb3077827e51daad6c84cb86a3e5ca80f7a43e0a4fb218c76
6d732971599bb495410301fd04a3bbb15d8643c74c6635f966e6629e7be9c065cc5b2499a081e361bf642bbb23e046e35542cd9e0ca14233a7875acb1b27e
d08547e98ef189c1fee56715ee6033322d85443501d1d593d40c0bbf4cef1eb8052bbd6d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b
4dd875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f
1305d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
```

```
value: 7.0
-----
time: 2023-06-22 01:26:04.293351
-----
------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d756246536389525581504c7eb2c065cb304317fecd6da49361285
417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e303
bb0084ddebec29b6e8affac8868fdab1bf180b7ff7437a1c40ef57774099276051a7323d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b
4dd875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f
1305d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
value: 3.0
-----
time: 2023-06-22 01:26:04.309004
-----
------
====================================
block # 3
```

```
block # 3
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b4dd
875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f130
5d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59ae
da11fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862
6416125f83f4c0b0f6c6869e1cb4734243b878b67bb277c4fd47ece61ec4cdbdf51f09f39530203010001
-----
value: 8.0
-----
time: 2023-06-22 01:26:04.309004
-----
------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b9341354db288f5178a2bd277fb83e9324571d097249a2ced437735b4dd
875c1209c2932c7924dca00b35dd943804098ce4dea300cc4a7233f428519ac9c6e65571408f283a6138b23f0a34c13ef222230b9aa99b3d44d75cc86f130
5d076c387dfdee5d45ed05db15ffc0e0fb9c0eb6ecb0face85ba4d56491f35b95447a3b90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100aab82d756246536389525581504c7eb2c065cb304317fecd6da49361
285417200d50b6e961ec6e44a6ae2d27b74d7e9889f152a6bb79c834c48b5abcde39aab0d696252dc891b6520b59dee6527f109f8b9b16c627bf462d8313e
```

```
value: 1.0
-----
time: 2023-06-22 01:26:04.309004
-----
------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c5843d8edfbb3077827e51daad6c84cb86a3e5ca80f7a43e0a4fb218c76
6d732971599bb495410301fd04a3bbb15d8643c74c6635f966e6629e7be9c065cc5b2499a081e361bf642bbb23e046e35542cd9e0ca14233a7875acb1b27e
d08547e98ef189c1fee56715ee6033322d85443501d1d593d40c0bbf4cef1eb8052bbd6d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100973702 40d5c15f370fe70490b8affaea92d954cdc7092afdbd9d59ae
da11fd9837767c634005ff477c559a14bd84918dd3bc1fa777f929697a3571cdd3d78499425a975c09510056acb22c25cf8737d777dc8e8b22b2b829c6862
6416125f83f4c0b0f6c6869e1cb4734243b878b67bb277c4fd47ece61ec4cdbdf51f09f39530203010001
-----
value: 5.0
-----
time: 2023-06-22 01:26:04.309004
-----
------
====================================
```

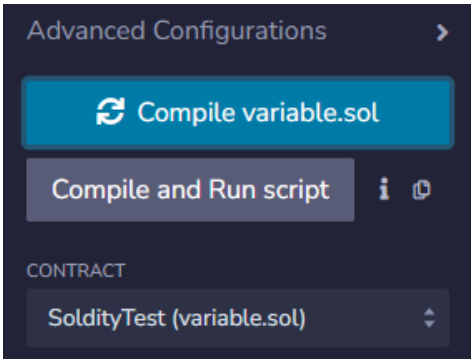
Practical No: 4.1.1

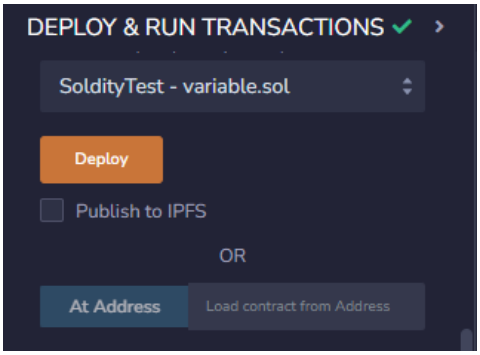
Aim: Implement and demonstrate the use of the following in Solidity: Variable

Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract SoldityTest{
    uint value;
    constructor() public {
        value = 10;
    }

    function getResult() public  view returns (uint){
        uint a = 8;
        uint b = 2;
        uint result = a +b;
        return result;
    }
}
```

Program Output:






Deployed Contracts
SOLDITYTEST AT 0XD7A...F771B
Balance: 0 ETH
getResult
0: uint256: 10

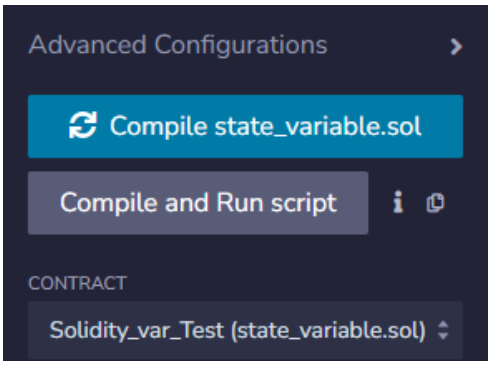
Practical No: 4.1.2

Aim: State Variable

Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Solidity_var_Test{
    uint8 public state_var;
    constructor() public {
        state_var =16;
    }
}
```

Program Output:



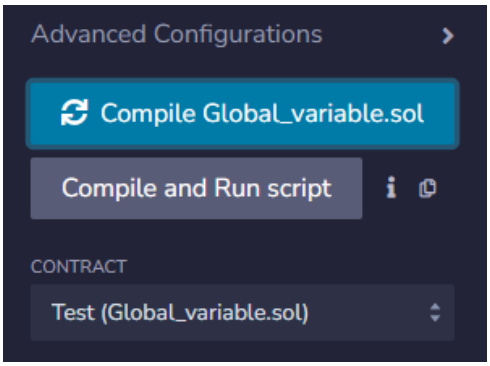
Practical No: 4.1.3

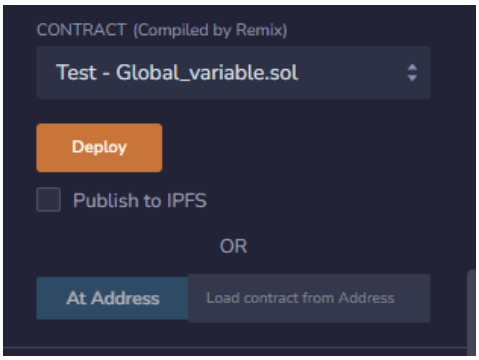
Aim: Global Variable

Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Test {
   address public admin;

   constructor()  public {
      admin = msg.sender;
   }
}
```

Program Output:






Deployed Contracts
TEST AT 0XD2A...FD005 (MEMOF
Balance: 0 ETH
admin
admin - call
0: address: 0x5B38Da6a701c568545dCfcB
03FcB875f56beddC4

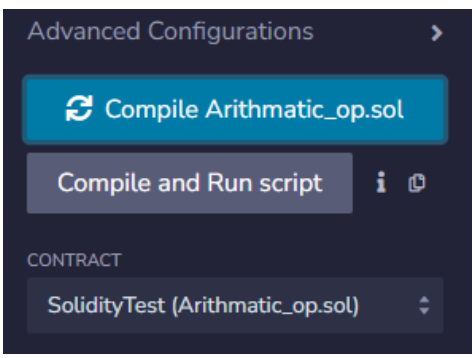
Practical No: 4.2

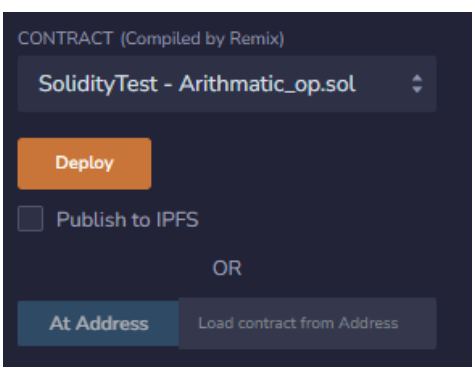
Aim: Implement and demonstrate the use of the following in Solidity: Operators

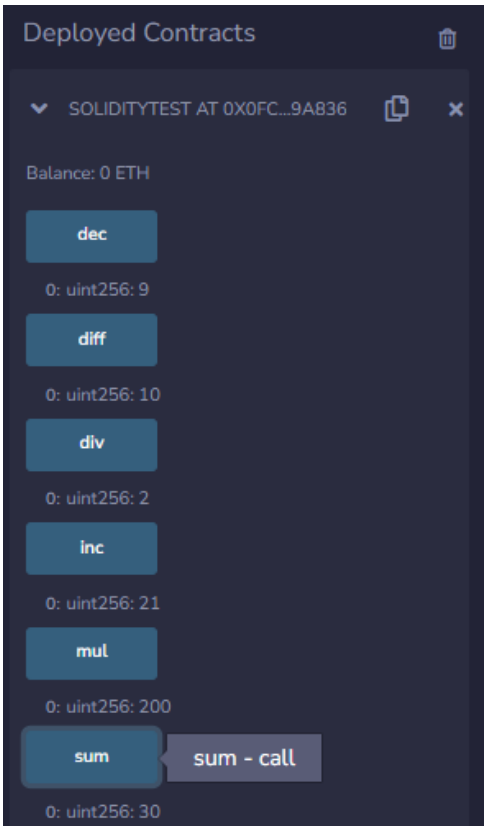
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;
contract SolidityTest{
   uint16 a = 20;
   uint16 b = 10;
   uint public  sum = a +b ;
   uint public diff = a-b;
   uint public mul = a*b;
   uint public div = a/b;
   uint public dec = --b;
   uint public inc = ++a;
}
```

Program Output:






Deployed Contracts
SOLIDITYTEST AT 0X0FC...9A836
Balance: 0 ETH
dec
0: uint256: 9
diff
0: uint256: 10
div
0: uint256: 2
inc
0: uint256: 21
mul
0: uint256: 200
sum
sum - call
0: uint256: 30

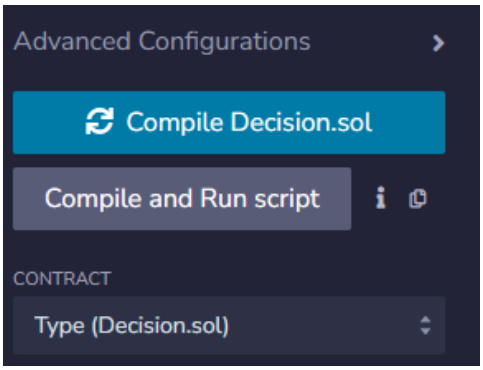
Practical No: 4.3.1

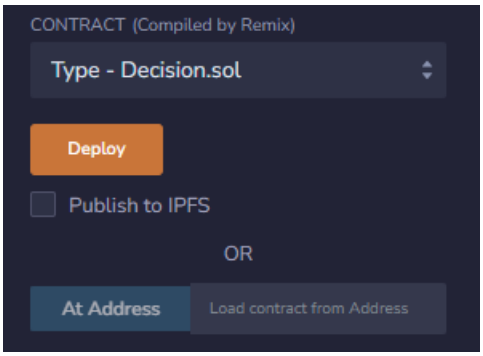
Aim: Implement and demonstrate the use of the following in Solidity: Decision Making

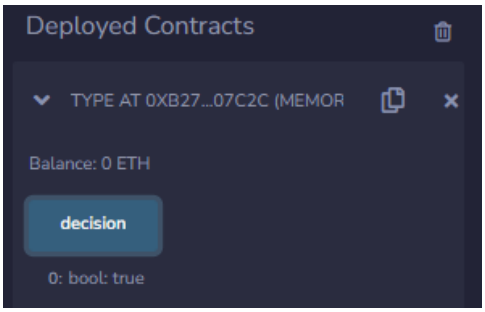
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Type{
    uint i = 10;
    function decision() public view returns(bool){
        if(i<=10){
            return true;
        }
    }
}
```

Program Output:






Deployed Contracts
TYPE AT 0XB27...07C2C (MEMOR
Balance: 0 ETH
decision
0: bool: true

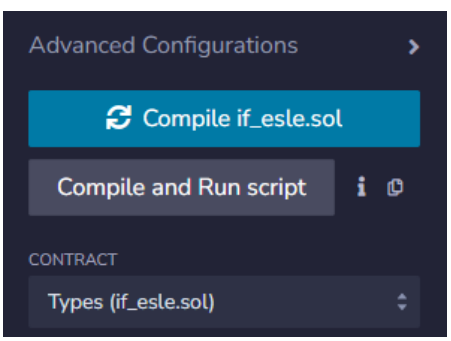
Practical No: 4.3.2

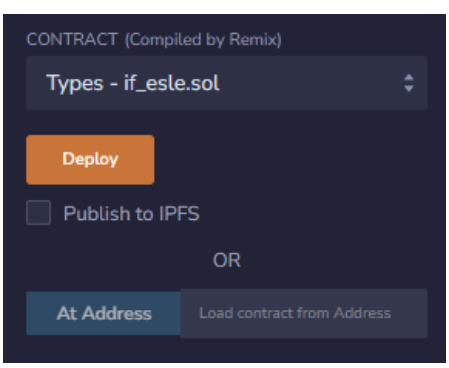
Aim: Implement and demonstrate the use of the following in Solidity: Decision Making using if else

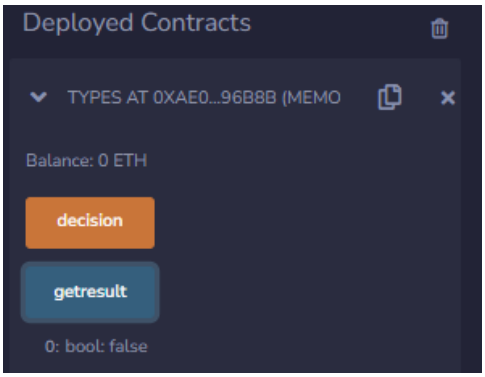
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Types{
    uint i = 7;
    bool even ;
    function decision() public {
        if (i%2 ==0){
            even = true;
        }
        else{
            even = false;
        }
    }
    function getresult() public view returns(bool)
    {
        return even;
    }
}
```

Program Output:






Deployed Contracts
TYPES AT 0XAE0...96B8B (MEMO
Balance: 0 ETH
decision
getresult
0: bool: false

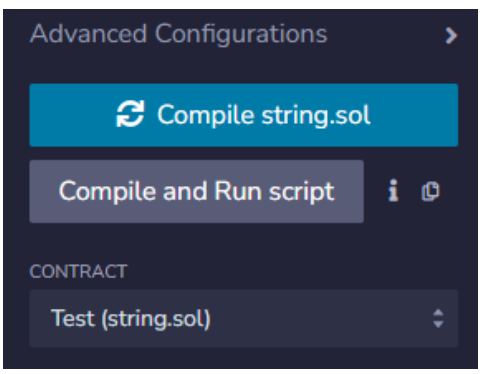
Practical No: 4.4

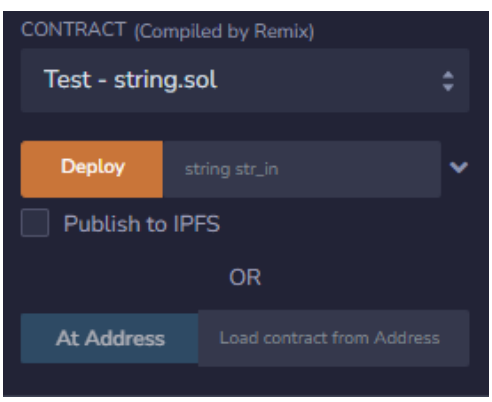
Aim: Implement and demonstrate the use of the following in Solidity: String

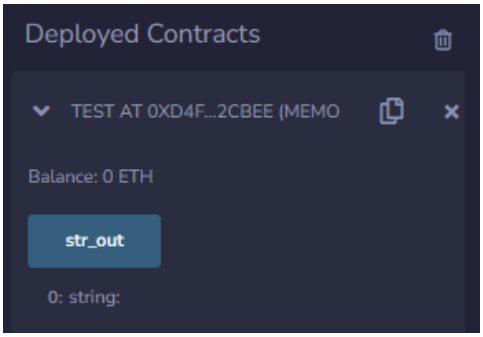
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Test{
    string str;
    constructor (string memory str_in)public {
        str = str_in;
    }
    function str_out() public view returns(string memory)
    {
        return str;
    }

}
```

Program Output:






Deployed Contracts
TEST AT 0XD4F...2CBEE (MEMO
Balance: 0 ETH
str_out
0: string:

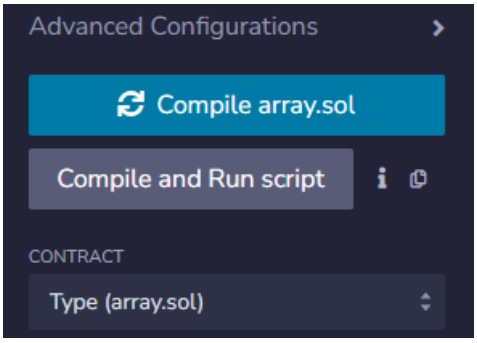
Practical No: 5.1

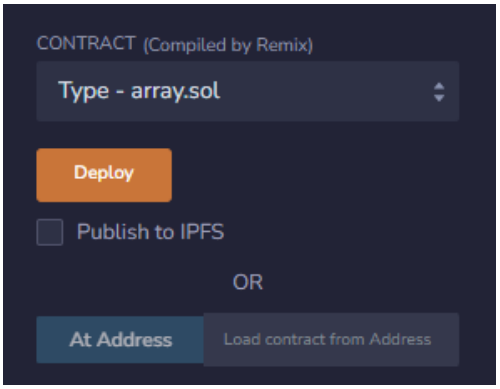
Aim: Implement and demonstrate the use of the following in Solidity: Arrays

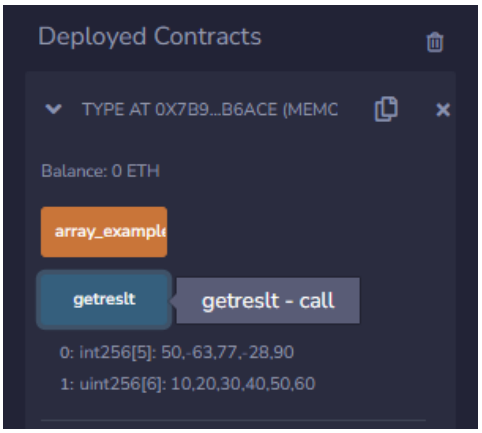
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Type{
   uint[6]data1;
   int[5]data;

   function array_example() public returns(int[5] memory, uint[6]memory){
      data = [int(50), -63, 77,-28, 90];
      data1 =[uint(10), 20,30,40,50,60];
   }
   function getreslt() public view returns(int[5]memory, uint[6]memory){
   return (data, data1);
   }
}
```

Program Output:






Deployed Contracts
TYPE AT 0X7B9...B6ACE (MEMC
Balance: 0 ETH
array_exampl
getreslt
getreslt - call
0: int256[5]: 50,-63,77,-28,90
1: uint256[6]: 10,20,30,40,50,60

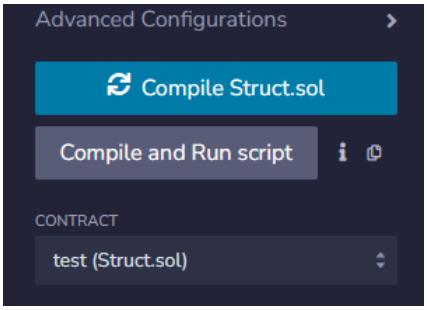
Practical No: 5.2

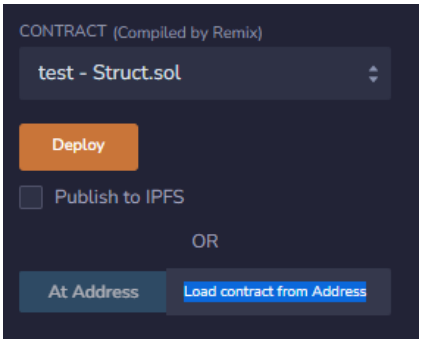
Aim: Implement and demonstrate the use of the following in Solidity: Structs

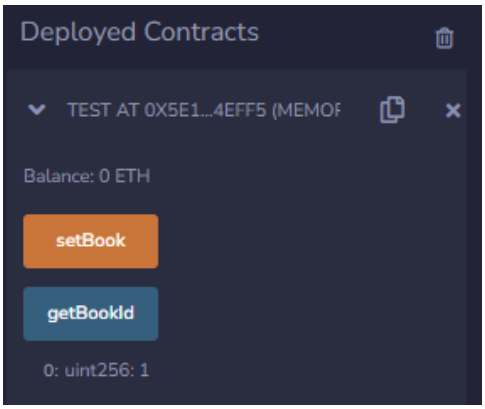
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract test{
   struct Book{
      string title;
      string author;
      uint book_id;
   }

   Book book;
   function setBook() public {
      book = Book('Lern Java','Tp', 1);
   }

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

Program Output:






Deployed Contracts
TEST AT 0X5E1...4EFF5 (MEMOF
Balance: 0 ETH
setBook
getBookId
0: uint256: 1

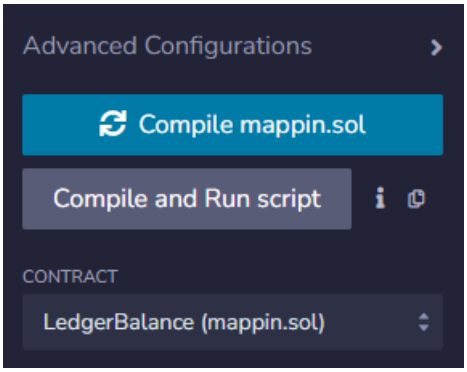
Practical No: 5.3.1

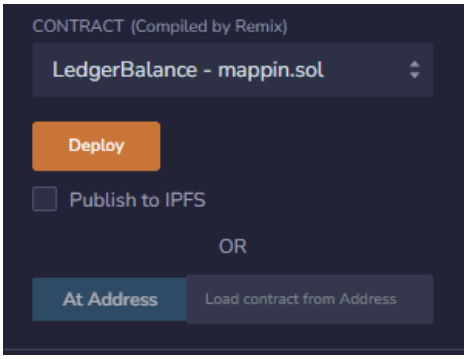
Aim: Implement and demonstrate the use of the following in Solidity: Mappings

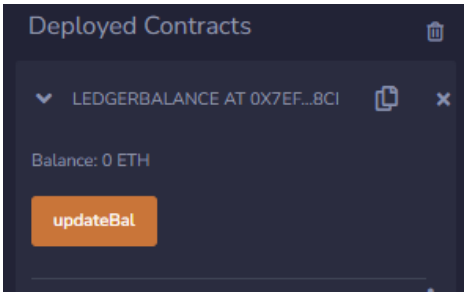
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract LedgerBalance {
   mapping (address => uint) bal;
   function updateBal() public returns (uint){
      bal[msg.sender] = 20 ;
      return bal[msg.sender];
   }
}
```

Program Output:







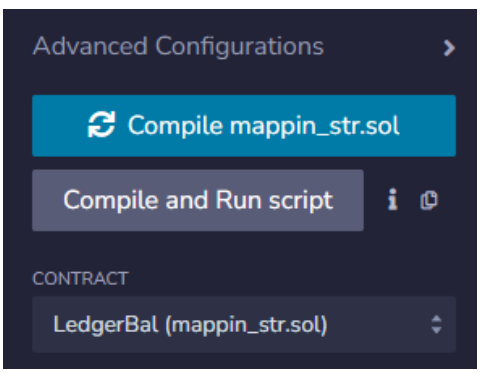
Practical No: 5.3.2

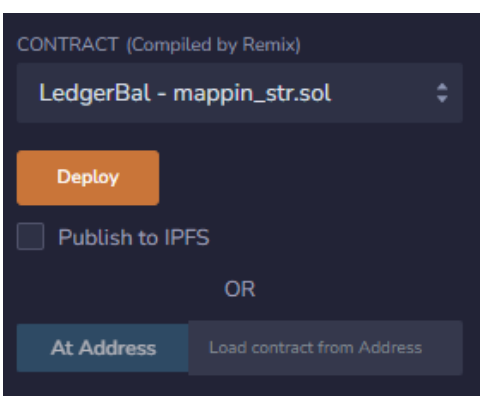
Aim: Implement and demonstrate the use of the following in Solidity: Mapping String

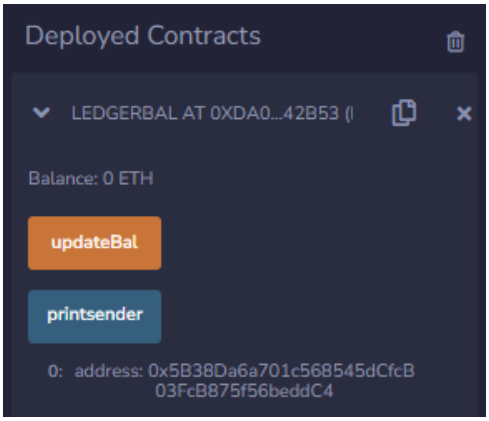
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract LedgerBal{
   mapping (address => string) name;

   function updateBal() public returns(string memory)
   {
      name[msg.sender] = 'Dip';
      return name[msg.sender];
   }

   function printsender() public view returns (address)
   {
      return msg.sender;
   }
}
```

Program Output:






Deployed Contracts
LEDGERBAL AT 0XDA0...42B53 (
Balance: 0 ETH
updateBal
printsender
0: address: 0x5B38Da6a701c568545dCfcB
03FcB875f56beddC4

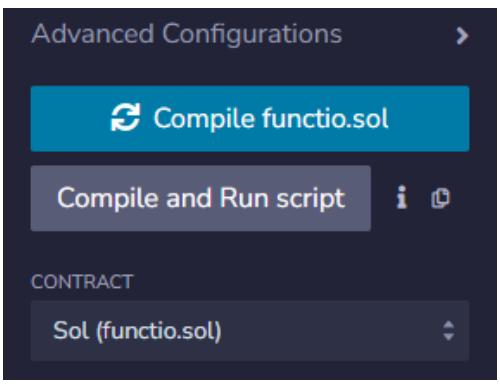
Practical No: 6.1

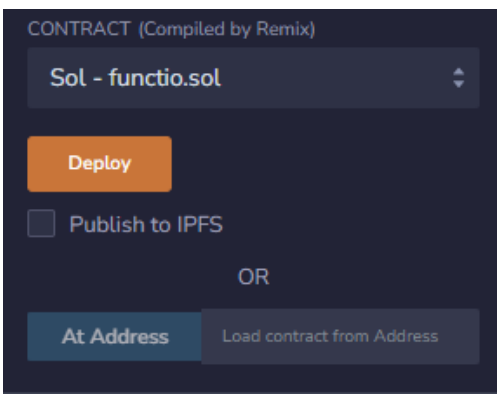
Aim: Implement and demonstrate the use of the following in Solidity: Functions

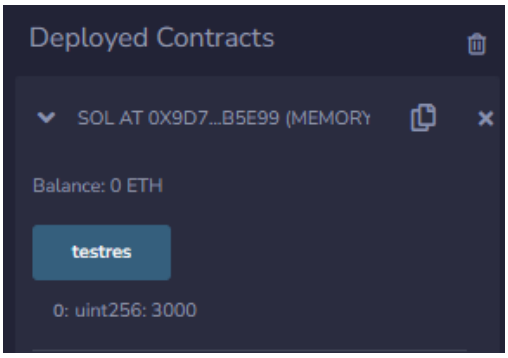
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Sol {
   function testres() public view returns (uint)
   {
      uint a = 2000;
      uint b = 1000;
      uint res = a + b;
      return res;
   }
}
```

Program Output:






Deployed Contracts
SOL AT 0X9D7...B5E99 (MEMORY
Balance: 0 ETH
testres
0: uint256: 3000

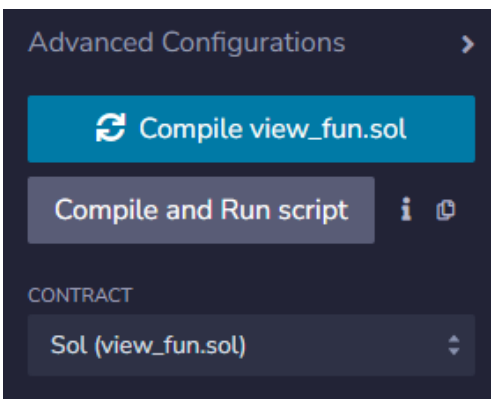
Practical No: 6.2.1

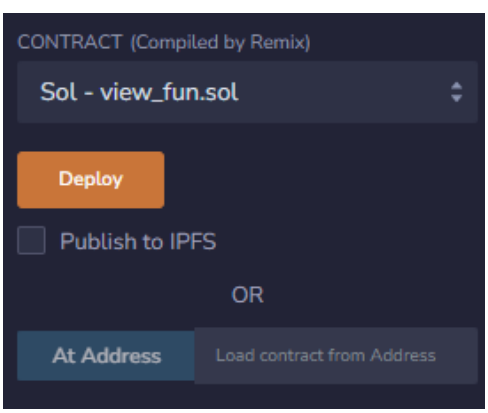
Aim: Implement and demonstrate the use of the following in Solidity: View Function

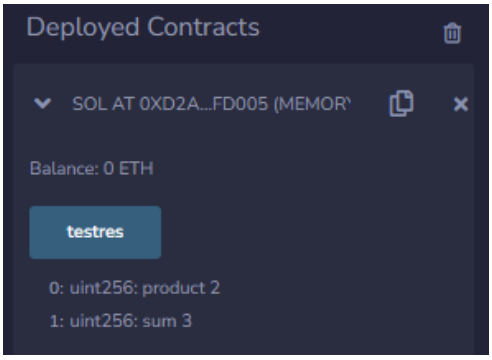
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Sol {
   function testres() public view returns (uint product, uint sum)
   {
      uint a = 2;
      uint b = 1;
      sum = a + b;
      product = a*b;
   }
}
```

Program Output:






Deployed Contracts
SOL AT 0XD2A...FD005 (MEMOR
Balance: 0 ETH
testres
0: uint256: product 2
1: uint256: sum 3

Practical No: 6.2.2

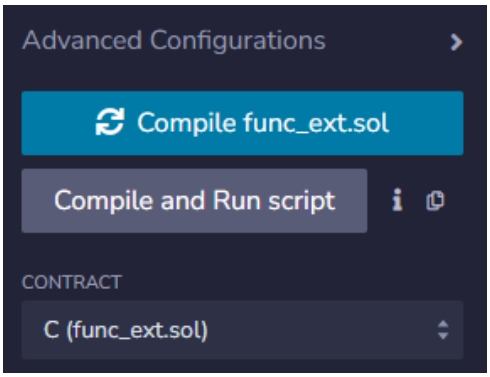
Aim: Implement and demonstrate the use of the following in Solidity: function external

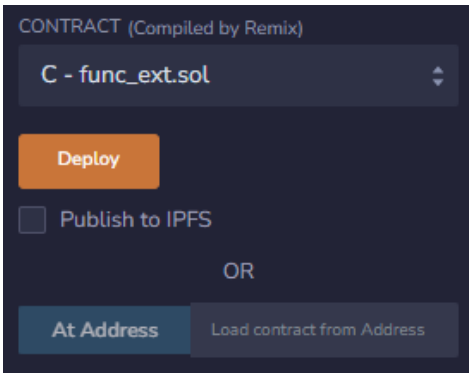
Program Code:

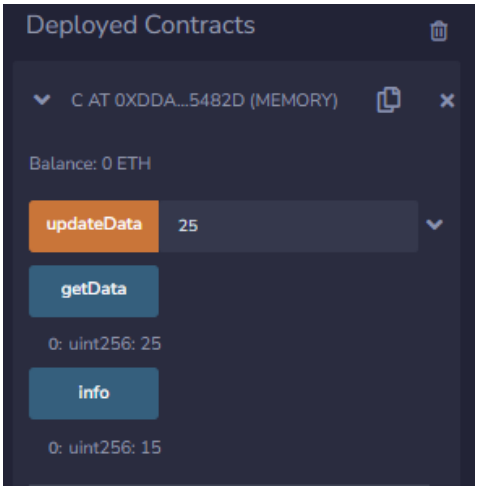
```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract C {
    uint private data;
    uint public info;
    constructor() public {
        info =15;
    }
    function inc(uint a) private pure returns(uint){
        return a +1;
    }
    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;
    }
}

contract E is C {
    uint private res;
    C private c;
    constructor() public {
        c = new C();
    }

    function getComuteRes() public {
        res = compute(3, 6);
    }

    function getRes() public view returns (uint){
        return res;
    }
```

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

Program Output:







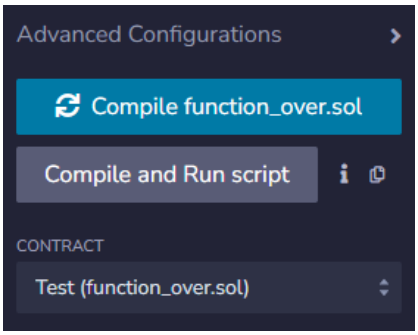
Practical No: 6.3

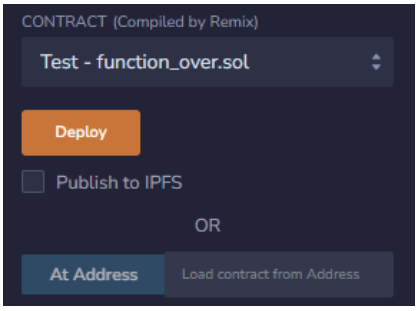
Aim: Implement and demonstrate the use of the following in Solidity: Function Overloading

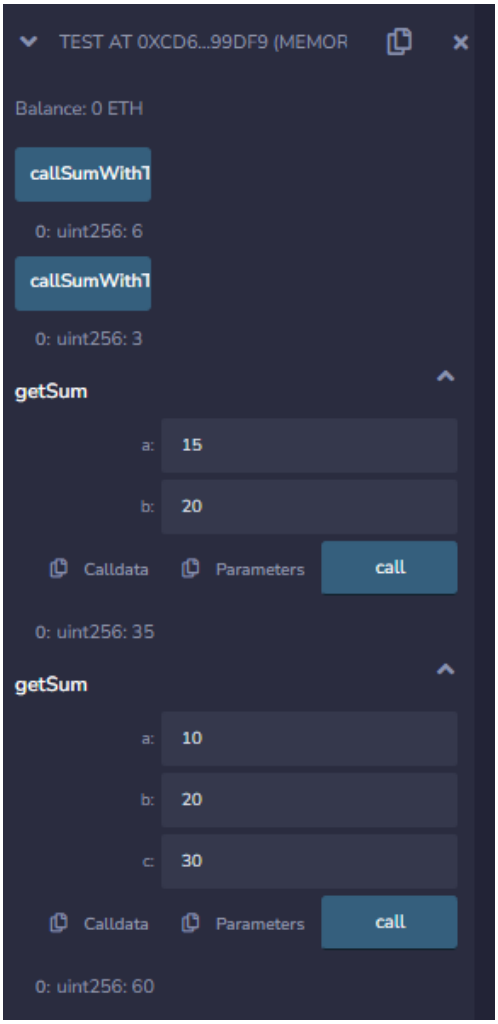
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Test {
   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 callSumWithTwoArgument() public pure returns(uint){
      return getSum(1, 2);
   }

   function callSumWithThreeArgument() public pure returns (uint){
      return getSum(1, 2, 3);
   }
}
```

Program Output:






TEST AT 0XCD6...99DF9 (MEMOR
Balance: 0 ETH
callSumWithT
0: uint256: 6
callSumWithT
0: uint256: 3
getSum
a: 15
b: 20
Calldata
Parameters
call
0: uint256: 35
getSum
a: 10
b: 20
c: 30
Calldata
Parameters
call
0: uint256: 60

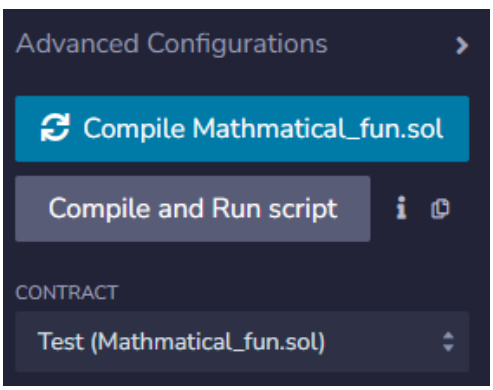
Practical No: 6.4

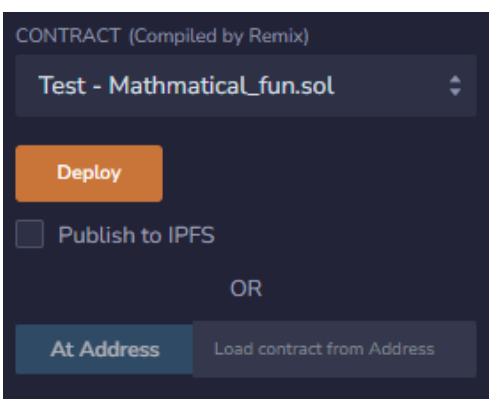
Aim: Implement and demonstrate the use of the following in Solidity: Mathematical Function

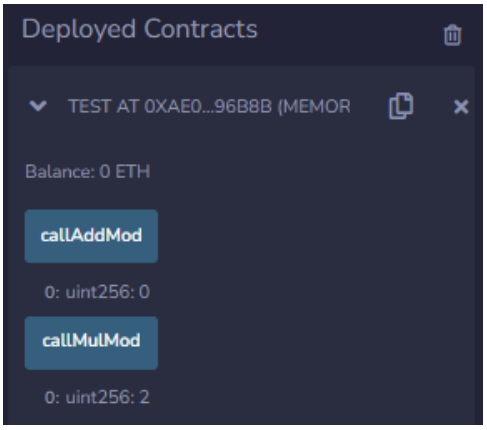
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Test{
    function callAddMod() public pure returns (uint){
        return addmod(4,5,3);
    }

    function callMulMod() public pure returns (uint){
        return mulmod(4, 5, 3);
    }
}
```

Program Output:






Deployed Contracts
TEST AT 0XAE0...96B8B (MEMOR
Balance: 0 ETH
callAddMod
0: uint256: 0
callMulMod
0: uint256: 2

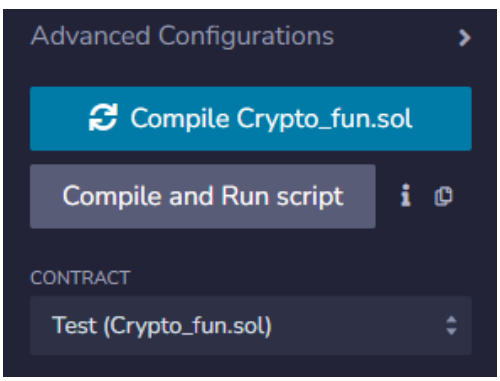
Practical No: 6.5

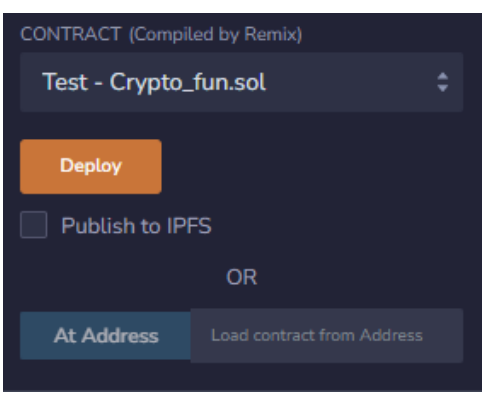
Aim: Implement and demonstrate the use of the following in Solidity: Cryptographic Functions

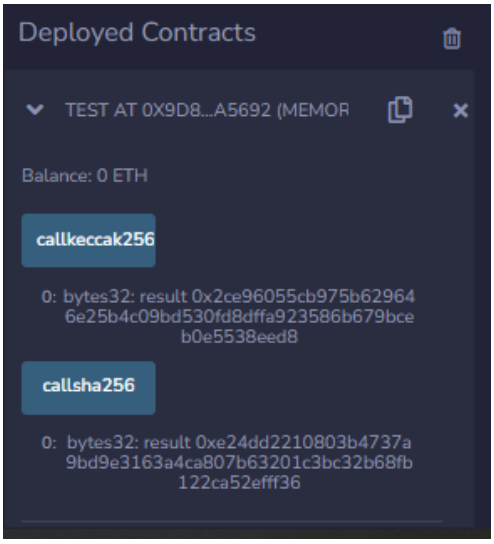
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Test{
    function callsha256() public pure returns (bytes32 result){
        return sha256("ronaldo");
    }

    function callkeccak256() public pure returns (bytes32 result){
        return keccak256("ronaldo");
    }

}
```

Program Output:






Deployed Contracts
TEST AT 0X9D8...A5692 (MEMOR
Balance: 0 ETH
callkeccak256
0: bytes32: result 0x2ce96055cb975b62964 6e25b4c09bd530fd8dffa923586b679bce b0e5538eed8
callsha256
0: bytes32: result 0xe24dd2210803b4737a 9bd9e3163a4ca807b63201c3bc32b68fb 122ca52efff36

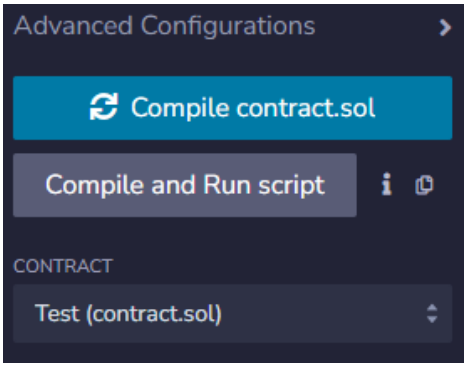
Practical No: 7.1

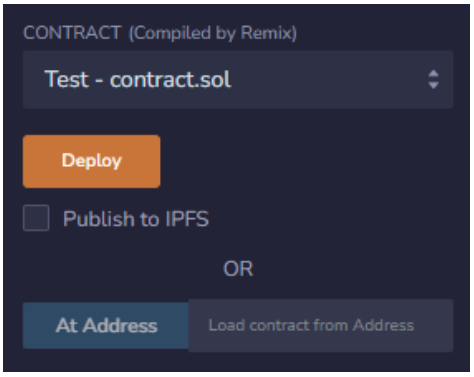
Aim: Implement and demonstrate the use of the following in Solidity: Contracts

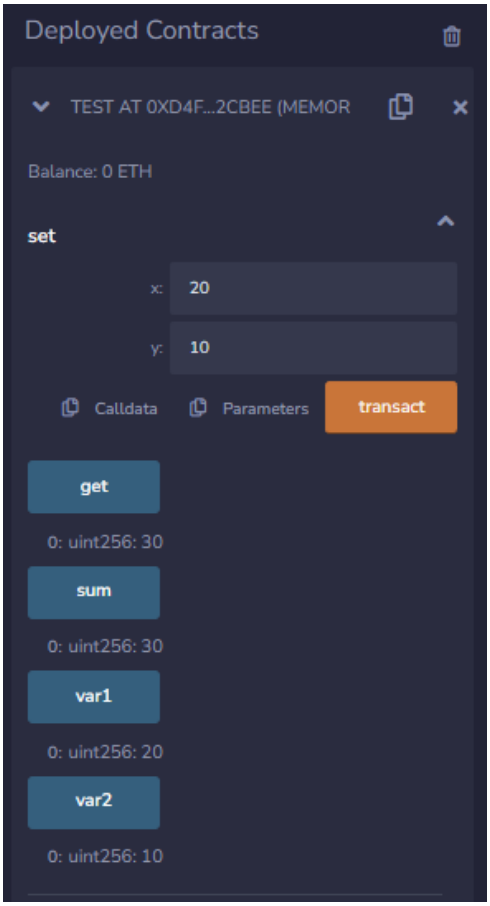
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract Test {
    uint public var1;
    uint public var2;
    uint public sum;

    function set(uint x, uint y) public {
        var1 = x;
        var2 = y;
        sum = var1 + var2;
    }

    function get() public view returns (uint) {
        return sum;
    }
}
```

Program Output:






Deployed Contracts
TEST AT 0XD4F...2CBEE (MEMOR
Balance: 0 ETH
set
x:
20
y:
10
Calldata
Parameters
transact
get
0: uint256: 30
sum
0: uint256: 30
var1
0: uint256: 20
var2
0: uint256: 10

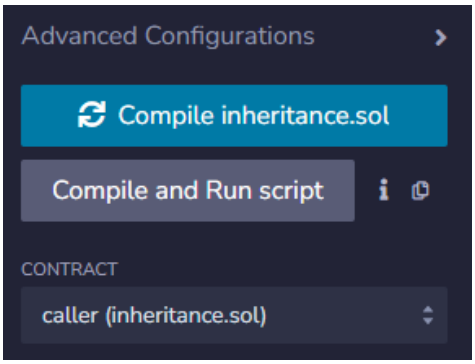
Practical No: 7.2

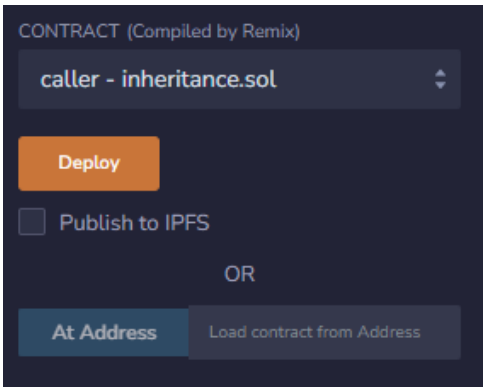
Aim: Implement and demonstrate the use of the following in Solidity: Inheritance

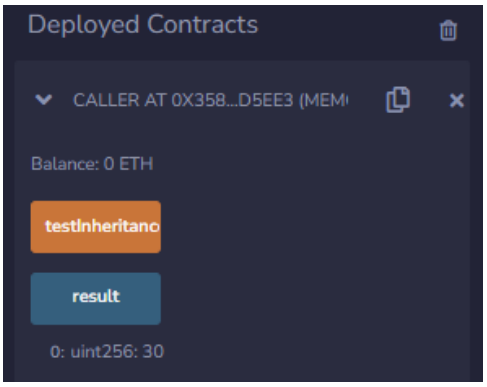
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract parent{
    uint internal sum;

    function setValue() external {
        uint a = 10;
        uint b = 20;
        sum = a + b;
    }

}

contract child is parent {
    function getValue() external view returns (uint){
        return sum;
    }
}

contract caller {
    child cc = new child();

    function testInheritance() public {
        cc.setValue();
    }

    function result() public view returns(uint){
        return cc.getValue();
    }
}
```

Program Output:


Advanced Configurations
Compile inheritance.sol
Compile and Run script
CONTRACT
caller (inheritance.sol)


CONTRACT (Compiled by Remix)
caller - inheritance.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address


Deployed Contracts
CALLER AT 0X358...D5EE3 (MEM
Balance: 0 ETH
testInheritano
result
0: uint256: 30

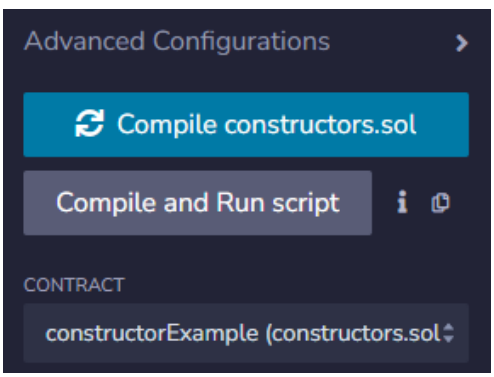
Practical No: 7.3

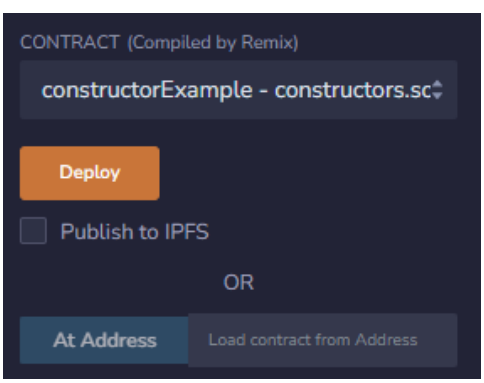
Aim: Implement and demonstrate the use of the following in Solidity: constructors

Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;
contract constructorExample{
    string str;

    constructor() public {
        str ="AbhishekMadhukarGawade";
    }

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

Program Output:






Deployed Contracts
CONSTRUCTOREXAMPLE AT 0X[
Balance: 0 ETH
getValue
0: string: AbhishekMadhukarGawade

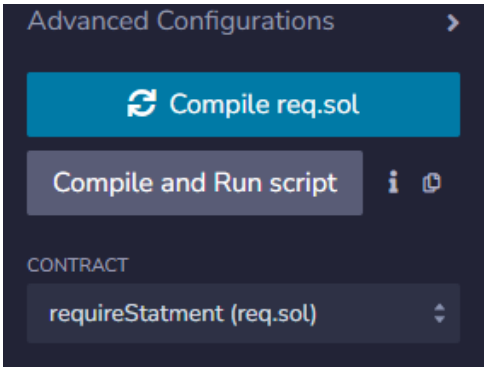
Practical No: 8.1.1

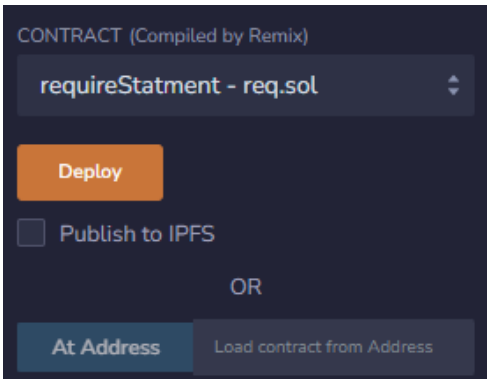
Aim: Implement and demonstrate the use of the following in Solidity: require statement

Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract requireStatment{
    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;
    }
}
```

Program Output:






Deployed Contracts
REQUIRESTATMENT AT 0X0FC...9
Balance: 0 ETH
checkInput
254
0: string: Input is uint8
Odd
253
0: bool: true

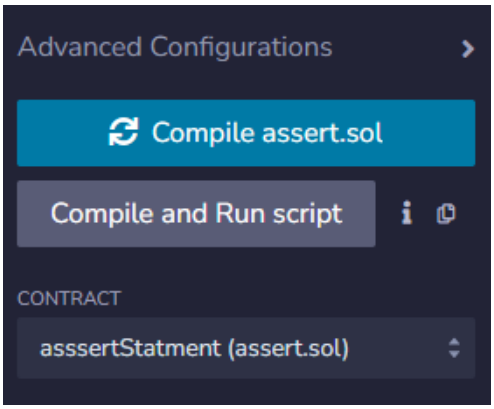
Practical No: 8.1.2

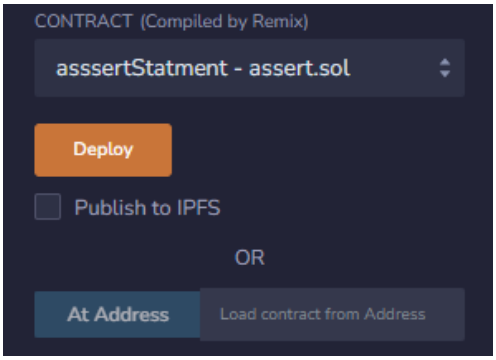
Aim: Implement and demonstrate the use of the following in Solidity: assert statement.1

Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract asssertStatment{
   bool result;

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

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

Program Output:




CONTRACT (Compiled by Remix)
asssertStatment - assert.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address


Deployed Contracts
ASSSERTSTATMENT AT 0XB27...(
Balance: 0 ETH
checkOverflov
24
getResult
0: string: Overflow exist

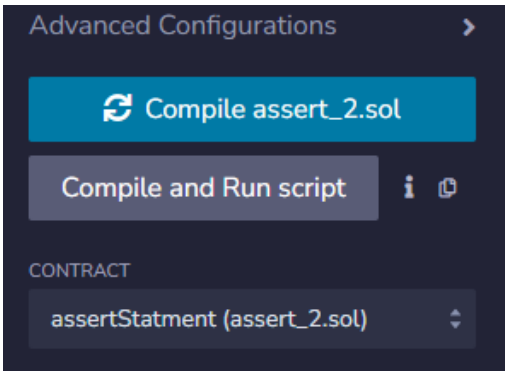
Practical No: 8.2

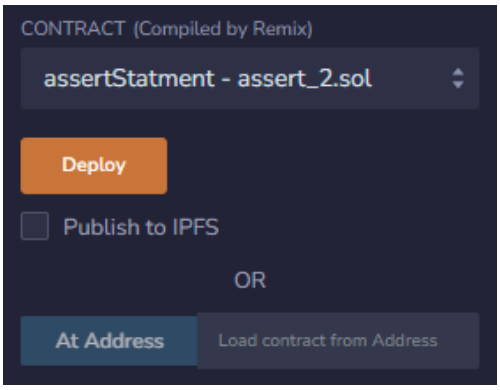
Aim: Implement and demonstrate the use of the following in Solidity: assert statement.2

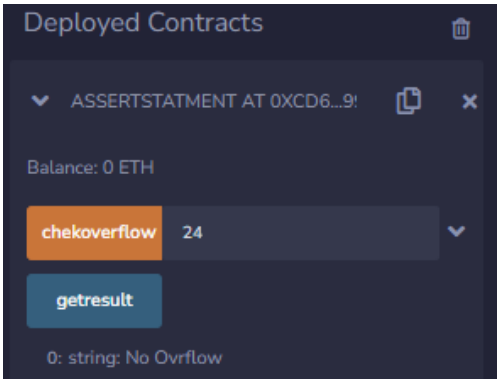
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

contract assertStatment {
   bool result;

   function chekoverflow(uint8 sum) public {
      assert(sum <= 255);
      result = true;
   }

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

Program Output:




CONTRACT (Compiled by Remix)
assertStatment - assert_2.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address


Deployed Contracts
ASSERTSTATMENT AT 0XCD6...9
Balance: 0 ETH
chekoverflow
24
getresult
0: string: No Ovrflow

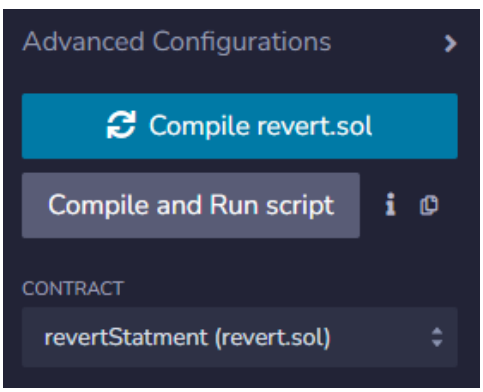
Practical No: 8.3

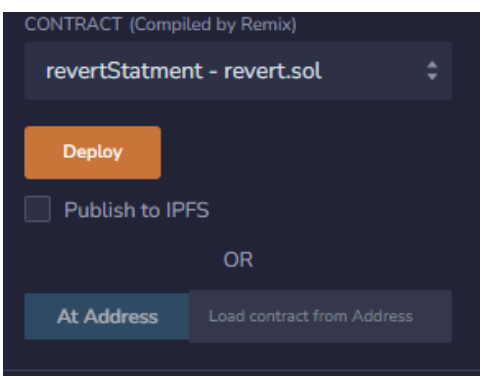
Aim: Implement and demonstrate the use of the following in Solidity: revert statement

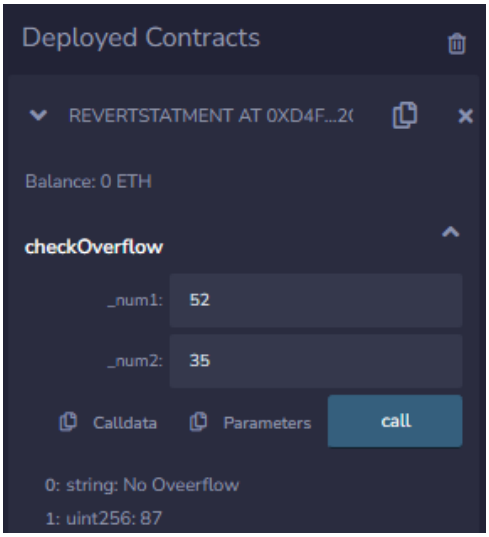
Program Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.5.0;

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

Program Output:






Deployed Contracts
REVERTSTATMENT AT 0XD4F...2(
Balance: 0 ETH
checkOverflow
_num1:
52
_num2:
35
Calldata
Parameters
call
0: string: No Oveerflow
1: uint256: 87

Keraleeya Samajam(Regd.) Dombivli's

MODEL COLLEGE

Re-Accredited Grade "A" by NAAC

Kanchan Goan Village, Khambalpada, Thakurli East – 421201
Contact No – 7045682157, 7045682158. www.model-college.edu.in

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr. /Miss _____

*Studying in Class*_____*Seat No.* _____

*Has completed the prescribed practicals in the subject*_____

*During the academic year*_____

Date : _____

External Examiner

Internal Examiner
M.Sc. Information Technology

Practical No	Title	Date	Signature
1.A	Convert the given text to speech.	11/3/2023	
1.B	Convert audio file Speech to Text.	11/3/2023	
2.A	Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like filelds, raw, words, sents, categories.	11/3/2023	
2.B	Create and use your own corpora (plaintext, categorical).	11/3/2023	
2.C	Study Conditional frequency distributions.	11/3/2023	
2D	Study of tagged corpora with methods like tagged_sents, tagged_words.	11/3/2023	
2.E	Write a program to find the most frequent noun tags.	11/3/2023	
2.F	Map Words to Properties Using Python Dictionaries	11/3/2023	
2.G.1	Study DefaultTagger.	11/3/2023	
2.G.2	Regular expression tagger.	11/3/2023	
2.G.3	UnigramTagger	11/3/2023	
2.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.	11/3/2023	
3.A	Study of Wordnet Dictionary with methods as synsets, definitions, examples,antonyms.	18/3/2023	
3.B	Study lemmas, hyponyms, hypernyms.	18/3/2023	
3.c	Write a program using python to find synonym and antonym of word "active" using Wordnet.	18/3/2023	
3.D	Compare two nouns.	18/3/2023	
3.E.1	Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List	18/3/2023	
3.E.2	Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List	18/3/2023	
3.E.3	Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List.	18/3/2023	
4.A	Tokenization using Python’s split() function.	18/3/2023	
4.B	Tokenization using Regular Expressions (RegEx)	18/3/2023	
4.C	Tokenization using NLTK	18/3/2023	
4.D	Tokenization using the spaCy library	18/3/2023	
4.E	Tokenization using Keras	18/3/2023	
4.F	Aim: Tokenization using Gensim	18/3/2023	
6	Illustrate part of speech tagging.		
6.A	Part of speech Tagging and chunking of user defined text.	8/4/2023	
6.B	Named Entity recognition using user defined text.	8/4/2023	
6.C	Named Entity recognition with diagram using NLTK corpus – treebank.	8/4/2023	
7	Finite state automata	8/4/2023	

Practical No	Title		
7.A	Define grammar using nltk. Analyze a sentence using the same.	8/4/2023	
7.B	Accept the input string with Regular expression of Finite Automaton: 101+.	8/4/2023	
7.C	Accept the input string with Regular expression of FA: (a+b)*bba.	8/4/2023	
7.D	Implementation of Deductive Chart Parsing using context free grammar and a given sentence.	8/4/2023	
8.	Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer, Study WordNetLemmatizer	15/4/2023	
9.	Implement Naive Bayes classifier	15/4/2023	
10.	Speech Tagging:		
10.A.1	Speech tagging using spacy	23/4/2023	
10.A.2	Speech tagging using nktl	23/4/2023	
10.B.1	Usage of Give and Gave in the Penn Treebank sample	23/4/2023	
10.B.2	probabilistic parser	23/4/2023	
10.C	Malt parsing: Parse a sentence and draw a tree using malt parsing.	23/4/2023	
11.A	Multiword Expressions in NLP	23/4/2023	
11.B	Normalized Web Distance and Word Similarity	29/4/2023	
11.C	Word Sense Disambiguation	29/4/2023	

Practical No: 1.A

Aim: Convert the given text to speech.

Program code:

```
from playsound import playsound

from gtts import gTTS

mytext = "hello every one we are doing Natrual Language Processing."

language = "en"

myobj = gTTS (text= mytext, lang= language, slow = False)

myobj.save("myfile_nlp.mp3")

playsound('myfile_nlp.mp3')
```

Program output:

Practical No:1.B

Aim: Convert audio file Speech to Text.

Program code:

```
import speech_recognition as sr

r = sr.Recognizer()

with sr.AudioFile('Ed-Sheeran-Shape-Of-You-Lyrics.wav') as source:

    audio_text = r.listen(source)

    text = r.recognize_google(audio_text, language="en",)

print(text)
```

Program output:

```
In [4]: runfile('D:/abhishek/model college/sem4/NLP/
pract1_2.py', wdir='D:/abhishek/model college/sem4/NLP')
I love you somebody like me
```

Practical No:2.A

Aim: Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like filelds, raw, words, sents, categories.

Program code:

```
import nltk

from nltk.corpus import brown

print('file ids of brown n corpus\n', brown.fileids())

ca01 = brown.words('ca01')

print('\n ca01 has following word:\n', ca01)

print('\n ca01 has ', len(ca01),'words')

print('\n \n categories oe file in brown corpus:\n')

print(brown.categories())

print('\n \n Static for each text: \n')

print('AvgWordLen\t AvgSentenceLen\t no.ofTimesachWordAppearsOnAvg\t\t FileName')

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('\t\t',int(num_chars/num_words),'\t\t\t',int(num_words/num_sents),'\t\t\t',
int(num_words/num_vocab),'\t\t\t\t\t\t',fileid)
```

Program output:

```
In [6]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_a.py', wdir='D:/abhishek/model college/sem4/NLP')
file ids of brown n corpus
 ['ca01', 'ca02', 'ca03', 'ca04', 'ca05', 'ca06', 'ca07',
'ca08', 'ca09', 'ca10', 'ca11', 'ca12', 'ca13', 'ca14', 'ca15',
'ca16', 'ca17', 'ca18', 'ca19', 'ca20', 'ca21', 'ca22', 'ca23',
'ca24', 'ca25', 'ca26', 'ca27', 'ca28', 'ca29', 'ca30', 'ca31',
'ca32', 'ca33', 'ca34', 'ca35', 'ca36', 'ca37', 'ca38', 'ca39',
'ca40', 'ca41', 'ca42', 'ca43', 'ca44', 'cb01', 'cb02', 'cb03',
'cb04', 'cb05', 'cb06', 'cb07', 'cb08', 'cb09', 'cb10', 'cb11',
'cb12', 'cb13', 'cb14', 'cb15', 'cb16', 'cb17', 'cb18', 'cb19',
```

```
'cp19', 'cp20', 'cp21', 'cp22', 'cp23', 'cp24', 'cp25', 'cp26',
'cp27', 'cp28', 'cp29', 'cr01', 'cr02', 'cr03', 'cr04', 'cr05',
'cr06', 'cr07', 'cr08', 'cr09']

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

 ca01 has  2242 words


 categories oe file in brown corpus:

['adventure', 'belles_lettres', 'editorial', 'fiction',
'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery',
'news', 'religion', 'reviews', 'romance', 'science_fiction']


 Static for each text:

AvgWordLen   AvgSentenceLen  no.ofTimesachWordAppearsOnAvg
FileName
         9           22              2                      ca01
         8           23              2                      ca02
         8           20              2                      ca03
         9           25              2                      ca04
```

Practical No: 2.B

Aim: Create and use your own corpora (plaintext, categorical).

Program code:

```
import nltk

from nltk.corpus import PlaintextCorpusReader

corpus_root ='D:/abhishek/model college/sem4/NLP'

filelist = PlaintextCorpusReader(corpus_root,'.*')

print('\n file list: \n')

print(filelist.fileids())

print(filelist.root)


print('\n \n static 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)
```

Program output:

```
In [3]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_b.py', wdir='D:/abhishek/model college/sem4/NLP')

 file list:

['4_a.py', 'Downloads/27-Jan-2023 at 114107 PM.pdf',
'Downloads/Abhishek resume.docx', 'Downloads/Acer Care
Center_Acer_4.00.3042_W10x64_A.zip', 'Ed-Sheeran-Shape-Of-You-
Lyrics.wav', 'Ed_Sheeran_Perfect_.wav', 'PRACT_7_D.py',
'a.txt', 'b.txt', 'c.txt', 'engmalt.linear-1.7.mco',
'maltparser-1.7.2/LICENSE', 'maltparser-1.7.2/NOTICE',
'maltparser-1.7.2/README', 'maltparser-1.7.2/appdata/
dataformat/bracketds.xml', 'maltparser-1.7.2/appdata/
dataformat/bracketps.xml', 'maltparser-1.7.2/appdata/
dataformat/conll2009.xml', 'maltparser-1.7.2/appdata/
dataformat/conllx.xml', 'maltparser-1.7.2/appdata/dataformat/
gramexds.xml', 'maltparser-1.7.2/appdata/dataformat/
gramexps.xml', 'maltparser-1.7.2/appdata/dataformat/
malttab.xml', 'maltparser-1.7.2/appdata/dataformat/
negrads.xml', 'maltparser-1.7.2/appdata/dataformat/
negraps.xml', 'maltparser-1.7.2/appdata/dataformat/pennds.xml',
'maltparser-1.7.2/appdata/dataformat/pennps.xml',
'maltparser-1.7.2/appdata/dataformat/tal05ds.xml',
'maltparser-1.7.2/appdata/dataformat/tal05ps.xml',
'maltparser-1.7.2/appdata/dataformat/tigerds.xml',
```

```
maltparser/parser/transition/package.html', 'maltparser-1.7.2/
src/org/maltparser/transform/pseudo/PseudoProjChartItem.java',
'maltparser-1.7.2/src/org/maltparser/transform/pseudo/
PseudoProjectivity.java', 'maltparser-1.7.2/src/org/maltparser/
transform/pseudo/TransformationException.java',
'maltparser-1.7.2/src/org/maltparser/transform/pseudo/
package.html', 'myfile.mp3', 'myfile_nlp.mp3', 'pract1_2.py',
'pract2_8.py', 'pract3.py', 'pract4_b.py', 'pract_10_a_2.py',
'pract_10_b_1.py', 'pract_10_b_2.py', 'pract_11_a.py',
'pract_11_b.py', 'pract_11_c.py', 'pract_1_a.py',
'pract_2_a.py', 'pract_2_b.py', 'pract_2_c.py', 'pract_2_d.py',
'pract_2_e.py', 'pract_2_f.py', 'pract_2_g_1.py',
'pract_2_g_2.py', 'pract_2_g_3.py', 'pract_2_h.py',
'pract_3_a.py', 'pract_3_b.py', 'pract_3_c.py', 'pract_3_d.py',
'pract_3_e_1.py', 'pract_3_e_2.py', 'pract_3_e_3.py',
'pract_4_c.py', 'pract_4_d.py', 'pract_4_e.py', 'pract_4_f.py',
'pract_5_1.py', 'pract_6_B.py', 'pract_6_a.py', 'pract_6_c.py',
'pract_7_a.py', 'pract_7_b.py', 'pract_7_c.py', 'pract_9.py',
'spam.csv', 'untitled8.py', 'word.txt']
D:\abhishek\model college\sem4\NLP


 static for each text

AvgWordLen  AvgSentenceLen  no.ofTimesEachWordAppearsOnAvg
FileName
```

Practical No:2.C

Aim: Study Conditional frequency distributions.

Program code:

```
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:])
cfd = nltk.ConditionalFreqDist(genre_word)
print(cfd)
print(cfd.conditions())
print(cfd['news'])
print(cfd['romance'])
print(list(cfd['romance']))
from nltk.corpus import inaugural
cfd = 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
language =
['Chickasaw','English','German_Deutsch','Greenlandic_Inuktikut','Hungarian_Magyar','Ibibio_Efik']
cfd = nltk.ConditionalFreqDist(
    (lang, len(word))
    for lang in language
    for word in udhr.words(lang + '-Latin1'))
cfd.tabulate(conditions=['English','German_Deutsch'], samples = range(10), cumulative = True)
```

Program output:

```
In [11]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_c.py', wdir='D:/abhishek/model college/sem4/NLP')
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>
[',', '.', 'the', 'and', 'to', 'a', 'of', '``', "''", 'was',
'I', 'in', 'he', 'had', '?', 'her', 'that', 'it', 'his', 'she',
'with', 'you', 'for', 'at', 'He', 'on', 'him', 'said', '!',
'--', 'be', 'as', ';', 'have', 'but', 'not', 'would', 'She',
'The', 'out', 'were', 'up', 'all', 'from', 'could', 'me',
```

```
'debate', 'raged', 'Financing', 'emerged', 'obstacle',
'contributed', 'maximum', 'underwrite', 'department', 'risk',
'risky', 'basis', 'capital', 'Heads', 'instinctively', 'onus',
'recriminations', 'broadcast', 'availing', 'in-laws', 'Sweat',
'forehead', 'disquietude', 'Across', 'saluted', 'jubilantly',
'encircled', 'forefinger', 'Spike-haired', 'burly', 'red-faced',
'decked', 'horn-rimmed', "officers'", 'expect', 'episode']
                 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
```

Practical No:2.D

Aim: Study of tagged corpora with methods like tagged_sents, tagged_words.

Program code:

```
import nltk

from nltk import tokenize

nltk.download('punkt')

nltk.download('words')

para = "Hello! My name is Abhishek Madhukar Gawade. Today you'll be learning NLTK under guidence prajakta mam."

sents = tokenize.sent_tokenize(para)

print("\n sentence tokenization\n ===================\n", sents)

print("\n words tokenization \n =====================\n")

for index in range(len(sents)):

    words = tokenize.word_tokenize(sents[index])

    print(words)

print('\n\n\n')
```

Program output:

```
In [12]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_d.py', wdir='D:/abhishek/model college/sem4/NLP')
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Aditi\AppData\Roaming\nltk_data...

 sentence tokenization
 ===================
 ['Hello!', 'My name is Abhishek Madhukar Gawade.', "Today
you'll be learning NLTK under guidence prajakta mam."]

 words tokenization
 =====================

['Hello', '!']
['My', 'name', 'is', 'Abhishek', 'Madhukar', 'Gawade', '.']
['Today', 'you', "'ll", 'be', 'learning', 'NLTK', 'under',
'guidence', 'prajakta', 'mam', '.']
```

Practical No:2.E

Aim: Write a program to find the most frequent noun tags.

Program code:

```
import nltk

from collections import defaultdict

text = nltk.word_tokenize("Abhishek likes to play foot ball. Abhishek does not like to play cricket")

tagged = nltk.pos_tag(text)

print(tagged)

addNounWord =[]

count = 0

for words in tagged:

    val = tagged[count][1]

    if (val == 'NN' or val =='NNS' or val =='NNP'):

        addNounWord.append(tagged[count][0])

    count+=1

print(addNounWord)

temp = defaultdict(int)

for sub in addNounWord:

    for wrd in sub.split():

        temp[wrd]+=1

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

print("word with maximum frequency:"+str(res))
```

Program output:

```
In [13]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_e.py', wdir='D:/abhishek/model college/sem4/NLP')
[('Abhishek', 'NNP'), ('likes', 'VBZ'), ('to', 'TO'), ('play',
'VB'), ('foot', 'RB'), ('ball', 'NN'), ('.', '.'), ('Abhishek',
'NNP'), ('does', 'VBZ'), ('not', 'RB'), ('like', 'VB'), ('to',
'TO'), ('play', 'VB'), ('cricket', 'NN')]
['Abhishek', 'ball', 'Abhishek', 'cricket']
word with maximum frequency:Abhishek
```

Practical No:2.F

Aim: Map Words to Properties Using Python Dictionaries

Program code:

```
thisdict = {
   "brand":"Ford",
   "model":"Mustang",
   "Year":1964
   }
print('\nList:\n',thisdict)
print('\nbrand from list:\n',thisdict["brand"])
print('\nLenfh of list:\n',len(thisdict))
print('\ntype of dict:\n',type(thisdict))
```

Program output:

```
In [14]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_f.py', wdir='D:/abhishek/model college/sem4/NLP')

List:
 {'brand': 'Ford', 'model': 'Mustang', 'Year': 1964}

brand from list:
 Ford

Lenfh of list:
 3

type of dict:
 <class 'dict'>
```

Practical No:2.G.1

Aim: Study DefaultTagger

Program code:

```
import nltk
from nltk.tag import DefaultTagger
```

```
exptager = DefaultTagger('NN')

from nltk.corpus import treebank

testsentence = treebank.tagged_sents()[1000:]

print(exptager.evaluate(testsentence))

from nltk.tag import DefaultTagger

exptager = DefaultTagger('NN')

print(exptager.tag_sents([['Hi',''],['How','are','you','?']]))
```

Program output:

```
In [15]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_g_1.py', wdir='D:/abhishek/model college/sem4/NLP')
d:\abhishek\model college\sem4\nlp\pract_2_g_1.py:13:
DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(exptager.evaluate(testsentence))
0.13198749536374715
[[('Hi', 'NN'), ('', 'NN')], [('How', 'NN'), ('are', 'NN'),
('you', 'NN'), ('?', 'NN')]]
```

Practical No:2.G.2

Aim: Regular expression tagger

Program 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'),

    (r'(The|the|A|a|An|an)$','AT'),

    (r'.*able$','JJ'),

    (r'.*ness$','NN'),

    (r'.*ly$','RB'),

    (r'.*s$','NNS'),

    (r'.*ing$','VBG'),
```

```
    (r'.*ed$','VBD'),
    (r'.*','NN')
    ])
print(regexp_tagger)
print(regexp_tagger.tag(test_sent))
```

Program output:

```
In [16]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_g_2.py', wdir='D:/abhishek/model college/sem4/NLP')
<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'), ('election', 'NN'),
('produced', 'VBD'), ('``', 'NN'), ('no', 'NN'), ('evidence',
'NN'), ("''", 'NN'), ('that', 'NN'), ('any', 'NN'),
('irregularities', 'NNS'), ('took', 'NN'), ('place', 'NN'),
('.', 'NN')]
```

Practical No: 2.G.3

Aim: UnigramTagger

Program code:

```
from nltk.tag import UnigramTagger
from nltk.corpus import treebank
train_sents = treebank.tagged_sents()[:10]
tagger = UnigramTagger(train_sents)
print(treebank.sents()[0])
print('\n', tagger.tag(treebank.sents()[0]))
tagger.tag(treebank.sents()[0])
tagger = UnigramTagger(model={'Pierre':'NN'})
print('\n',tagger.tag(treebank.sents()[0]))
```

Program output:

```
In [17]: runfile('D:/abhishek/model college/sem4/NLP/
pract_2_g_3.py', wdir='D:/abhishek/model college/sem4/NLP')
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will',
'join', 'the', 'board', '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), ('board', None), ('as', None),
('a', None), ('nonexecutive', None), ('director', None),
('Nov.', None), ('29', None), ('.', None)]
```

Practical No:2.H

Aim: 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.

Program code:

Program output:

Practical No:3.A

Aim: Study of Wordnet Dictionary with methods as synsets, definitions, examples,antonyms.

Program code:

```
import nltk

from nltk.corpus import wordnet

print(wordnet.synsets("computer"))

print(wordnet.synset("computer.n.01").definition())

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

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

Program output:

```
In [23]: runfile('D:/abhishek/model college/sem4/NLP/
pract_3_a.py', wdir='D:/abhishek/model college/sem4/NLP')
[Synset('computer.n.01'), Synset('calculator.n.01')]
a machine for performing calculations automatically
Examples: []
[Lemma('sell.v.01.sell')]
```

Practical No:3.B

Aim: Study lemmas, hyponyms, hypernyms.

Program code:

```
import nltk

from nltk.corpus import wordnet

print(wordnet.synsets("computer"))

print(wordnet.synset("computer.n.01").lemma_names())

for e in wordnet.synsets("computer"):

    print(f'{e} --> {e.lemma_names()}')

print(wordnet.synset("computer.n.01").lemmas())

print(wordnet.lemma('computer.n.01.computing_device').synset())

print(wordnet.lemma('computer.n.01.computing_device').name())

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

print(syn.hyponyms)

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

vehicle = wordnet.synset('vehicle.n.01')
```

car = wordnet.synset('car.n.01')

print(car.lowest_common_hypernyms(vehicle))

Program output:

```
In [24]: runfile('D:/abhishek/model college/sem4/NLP/
pract_3_b.py', wdir='D:/abhishek/model college/sem4/NLP')
[Synset('computer.n.01'), Synset('calculator.n.01')]
['computer', 'computing_machine', 'computing_device',
'data_processor', 'electronic_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'), Lemma('computer.n.01.computing_device'),
Lemma('computer.n.01.data_processor'), Lemma('computer.n.
01.electronic_computer'), Lemma('computer.n.
01.information_processing_system')]
Synset('computer.n.01')
computing_device
<bound method _WordNetObject.hyponyms of Synset('computer.n.
01')>
['analog_computer', 'analogue_computer', 'digital_computer',
'home_computer', 'node', 'client', 'guest', 'number_cruncher',
'pari-mutuel_machine', 'totalizer', 'totaliser', 'totalizator',
'totalisator', 'predictor', 'server', 'host', 'Turing_machine',
'web_site', 'website', 'internet_site', 'site']
[Synset('vehicle.n.01')]
```

Practical No:3.C

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

Program code:

from nltk.corpus import wordnet

print(wordnet.synsets("active"))

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

Program output:

```
In [25]: runfile('D:/abhishek/model college/sem4/NLP/
pract_3_c.py', wdir='D:/abhishek/model college/sem4/NLP')
[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('active.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')]
```

Practical No:3.D

Aim: Compare two nouns.

Program code:

import nltk

from nltk.corpus import wordnet

```
syn1 = wordnet.synsets('football')

syn2 = wordnet.synsets('soccer')

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()
```

Program output:

```
In [26]: runfile('D:/abhishek/model college/sem4/NLP/
pract_3_d.py', wdir='D:/abhishek/model college/sem4/NLP')
path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with
a ball (round or oval) in which two teams try to kick or carry
or propel the ball into each other's goal ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams
of 11 players try 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 American football ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams
of 11 players try to kick or head a ball into the opponents'
goal ]
is 0.05
```

Practical No:3.e Handling stopword: 1.

Aim: Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List

Program code:

```
import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

from nltk.tokenize import word_tokenize

text = "Yashesh like 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)

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)

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)
```

Program output:

```
In [27]: runfile('D:/abhishek/model college/sem4/NLP/
pract_3_e_1.py', wdir='D:/abhishek/model college/sem4/NLP')
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Aditi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['Yashesh', 'play', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'like', 'football', ',', 'however', 'fond',
'tennis', '.']
['Yashesh', 'like', 'football', ',', 'however', 'not', 'fond',
'tennis', '.']
```

Practical No:3.E.2

Aim: Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List

Program code:

```
import gensim

from gensim.parsing.preprocessing import remove_stopwords

from nltk.tokenize import word_tokenize

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

filtered_sentense = remove_stopwords(text)

print(filtered_sentense)

all_stpwords = gensim.parsing.preprocessing.STOPWORDS

print( all_stpwords)

from gensim.parsing.preprocessing import STOPWORDS

all_stpwords_genism = 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_stpwords]
```

```
print(tokens_without_sw)

print("========================================")

all_stpwords_genism = STOPWORDS

sw_list = {"not"}

all_stpwords_genism = 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_stpwords_genism]

print(all_stpwords_genism)

print(tokens_without_sw)
```

Program output:

```
In [29]: runfile('D:/abhishek/model college/sem4/NLP/
pract_3_e_2.py', wdir='D:/abhishek/model college/sem4/NLP')
Yashesh likes play football, fond tennis.
frozenset({'do', 'their', 'six', 'in', 'whereas', 'a',
'between', 'even', 'hereby', 'fire', 'please', 'our', 'very',
'becoming', 'is', 'amongst', 'her', 'un', 'more', 'by', 'one',
'whether', 'unless', 'had', 'nobody', 'been', 'somewhere',
'should', 'those', 'which', 'moreover', 'less', 'con', 'beyond',
```

```
'everyone', 'become', 'five', 'back', 'at', 'myself', 'indeed',
'using', 'already', 'behind', 'everything', 'hasnt', 'both',
'during', 'on', 'toward', 'us', 'amoungst', 'cant', 'doing',
'kg', 'two', 'cannot', 'move', 'find', 'all', 'such',
'themselves', 'are', 'anyone', 'de', 'never', 'almost',
'became', 'through', 'here', 'go', 'how'})
['Yashesh', 'likes', 'play', 'football', ',', 'fond', 'tennis',
'.']
```

```
'move', 'afterwards', 'and', 'find', 'all', 'such',
'themselves', 'are', 'anyone', 'de', 'ie', 'still', 'never',
'him', 'meanwhile', 'almost', 'became', 'amount', 'part', 'ten',
'through', 'various', 'you', 'ltd', 'am', 'here', 'go', 'how',
'nine'})
['Yashesh', 'likes', 'play', 'football', ',', 'not', 'fond',
'tennis', '.']
```

Practical No:3.E.3

Aim: Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List.

Program code:

```
import spacy

import nltk

from nltk.tokenize import word_tokenize

sp = spacy.load('en_core_web_sm')
```

```
all_stopwords = sp.Defaults.stop_words

all_stopwords.add('play')

text = "Yashesh like 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)

all_stopwords.remove('not')

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

print(tokens_without_sw)
```

Program output:

```
In [1]: runfile('D:/abhishek/model college/sem4/NLP/
pract_3_e_3.py', wdir='D:/abhishek/model college/sem4/NLP')
['Yashesh', 'like', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'like', 'football', ',', 'not', 'fond', 'tennis',
'.']
```

Practical No:4.A

Aim: Tokenization using Python's split() function.

Program code:

```
ext = "Monism is a thesis about oneness: that only one thing exists in a certain sense. The denial of monism is pluralism, the thesis that, in a certain sense, more than one thing exists.[7] There are many forms of monism and pluralism, but in relation to the world as a whole, two are of special interest: existence monism/pluralism and priority monism/pluralism."

data = text.split('.')

for i in data:

    print(i)
```

Program output:

```
In [2]: runfile('D:/abhishek/model college/sem4/NLP/4_a.py',
wdir='D:/abhishek/model college/sem4/NLP')
Monism is a thesis about oneness: that only one thing exists in
a certain sense
 The denial of monism is pluralism, the thesis that, in a
certain sense, more than one thing exists
[7] There are many forms of monism and pluralism, but in
relation to the world as a whole, two are of special interest:
existence monism/pluralism and priority monism/pluralism
```

Practical No:4.B

Aim: Tokenization using Regular Expressions (RegEx)

Program code:

```
import nltk

from nltk.tokenize import RegexpTokenizer

tk = RegexpTokenizer('\s+', gaps=True)

str = "I love to study Natrual Language processing in python"

tokkens = tk.tokenize(str)

print(tokkens)
```

Program output:

```
In [3]: runfile('D:/abhishek/model college/sem4/NLP/
pract4_b.py', wdir='D:/abhishek/model college/sem4/NLP')
['I', 'love', 'to', 'study', 'Natrual', 'Language',
'processing', 'in', 'python']
```

Practical No:4.C

Aim: Tokenization using NLTK

Program code:

```
import nltk

from nltk.tokenize import word_tokenize

str = "I love to study Natrual Language processing in python"

print(word_tokenize(str))
```

Program output:

```
In [4]: runfile('D:/abhishek/model college/sem4/NLP/
pract_4_c.py', wdir='D:/abhishek/model college/sem4/NLP')
['I', 'love', 'to', 'study', 'Natrual', 'Language',
'processing', 'in', 'python']
```

Practical No:4.D

Aim: Tokenization using the spaCy library

Program code:

```
import spacy

nlp = spacy.blank("en")

str = "I love to study Natrual Language processing in python"

doc = nlp(str)

words = [ word.text for word in doc]

print(words)
```

Program output:

```
In [5]: runfile('D:/abhishek/model college/sem4/NLP/
pract_4_d.py', wdir='D:/abhishek/model college/sem4/NLP')
['I', 'love', 'to', 'study', 'Natrual', 'Language',
'processing', 'in', 'python']
```

Practical No:4.E

Aim: Tokenization using Keras

Program code:

```
import keras

from keras.preprocessing.text import text_to_word_sequence

str = "I love to study Natrual Language processing in python"

tokens = text_to_word_sequence(str)

print(tokens)
```

Program output:

```
In [6]: runfile('D:/abhishek/model college/sem4/NLP/
pract_4_e.py', wdir='D:/abhishek/model college/sem4/NLP')
['i', 'love', 'to', 'study', 'natrual', 'language',
'processing', 'in', 'python']
```

Practical No:4.F

Aim: Tokenization using Gensim

Program code:

```
from gensim.utils import tokenize

str = "I love to study Natrual Language processing in python"

print(list(tokenize(str)))
```

Program output:

```
In [7]: runfile('D:/abhishek/model college/sem4/NLP/
pract_4_f.py', wdir='D:/abhishek/model college/sem4/NLP')
['I', 'love', 'to', 'study', 'Natrual', 'Language',
'processing', 'in', 'python']
```

Practical No:6 Illustrate part of speech tagging.1

Aim: Part of speech Tagging and chunking of user defined text.

Program 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 Kapdia. Today you'll be learning NTK."
sents = tokenize.sent_tokenize(para)
print('\n sensentense to tokenize\n ============\n', sents )
print('\n word to tokenize\n ============\n', sents )
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)
tagged_words =[]
for index in range(len(sents)):
    tagged_words.append(tag.pos_tag(words))
print("\n POS Tagging \n =========\n",tagged_words)
tree =[]
for index in range(len(sents)):
    tree.append(chunk.ne_chunk(tagged_words[index]))
print('\n chunking\n==========\n')
print(tree)
```

Program output:

```
In [8]: runfile('D:/abhishek/model college/sem4/NLP/
pract_6_a.py', wdir='D:/abhishek/model college/sem4/NLP')
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Aditi\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Aditi\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-
to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\Aditi\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\Aditi\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!

 sensentense to tokenize
 ============
 ['Hello!', 'My name is Beena Kapdia.', "Today you'll be
learning NTK."]

 word to tokenize
 ============
 ['Hello!', 'My name is Beena Kapdia.', "Today you'll be
```

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

 POS Tagging
 =========
 [[('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'),
('learning', 'VBG'), ('NTK', 'NNP'), ('.', '.')], [('Today',
'NN'), ('you', 'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning',
'VBG'), ('NTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('you',
'PRP'), ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'),
('NTK', 'NNP'), ('.', '.')]]

 chunking
==========

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

Practical No:6.B

Aim: Named Entity recognition using user defined text.

Program code:

```
import spacy

nlp = spacy.load("en_core_web_sm")

text = ("when seabastian Thrun started working on self-driving cars at"

"Google in 2007, few people oustside of the company took him"

"Seroiusly,I can tell you very senior CEO's of major Amercan"

"car companies would shake my hand and turn away beacause I wasn't"

"worth talking to,said Thurn , in an interview with recorder earlier")
```

```
doc = nlp(text)

print("nouns:\n",[chunk.text for chunk in doc.noun_chunks])

print("verbs",[token.lemma_ for token in doc if token.pos_ =="VERB"])
```

Program output:

```
In [9]: runfile('D:/abhishek/model college/sem4/NLP/
pract_6_B.py', wdir='D:/abhishek/model college/sem4/NLP')
nouns:
 ['seabastian Thrun', 'self-driving cars atGoogle', 'few
people', 'the company', 'I', 'you', 'very senior CEO', 'major
Amercancar companies', 'my hand', 'I', 'Thurn', 'an interview',
'recorder']
verbs ['start', 'work', 'drive', 'take', 'tell', 'shake',
'turn', 'talk', 'say']
```

Practical No:6.C

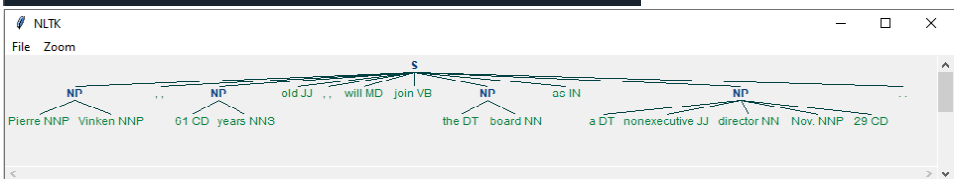
Aim: Named Entity recognition with diagram using NLTK corpus – treebank.

Program code:

```
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()
```

Program outpu**t:**

```
In [10]: runfile('D:/abhishek/model college/sem4/NLP/
pract_6_c.py', wdir='D:/abhishek/model college/sem4/NLP')
[nltk_data] Downloading package treebank to
[nltk_data]     C:\Users\Aditi\AppData\Roaming\nltk_data...
[nltk_data]   Package treebank is already up-to-date!
```



Practical No:7.A: Finite state automata

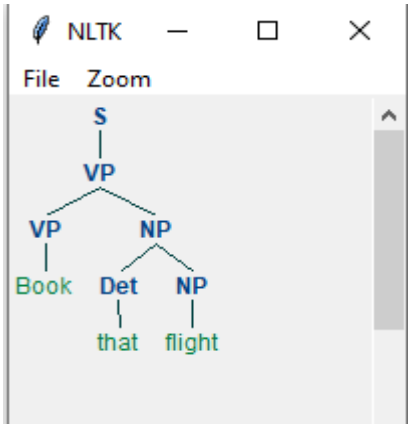
Aim: Define grammar using nltk. Analyze a sentence using the same.

Program 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()
```

Program output:

```
In [11]: runfile('D:/abhishek/model college/sem4/NLP/
pract_7_a.py', wdir='D:/abhishek/model college/sem4/NLP')
['Book', 'that', 'flight']
(S (VP (VP Book) (NP (Det that) (NP flight))))
```



Practical No:7.B

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

Program code:

```
def FA(s):
    if len(s)<3:
        return "rejected"
    if s[0] =="1":
        if s[1]=="0":
            if s[0]=="1":
                for i in range(3,len(s)):
                    if s[i]!="1":
                        return "rejected"
                return "accepted"
            return "rejected"
        return "rejected"
    return "rejected"
inputs = ['1','10101','101','1011','01010','100','10111101','1011111',' ']
for i in inputs:
    print(FA(i))
```

Program output:

```
In [1]: runfile('D:/abhishek/model college/sem4/NLP/
pract_7_b.py', wdir='D:/abhishek/model college/sem4/
NLP')
rejected
rejected
accepted
accepted
rejected
accepted
rejected
accepted
rejected
```

Practical No:7.C

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

Program code:

```
def FA(s):
    size = 0
    for i in s:
        if i =='a' or i =='b':
            size+=1
        else:
            return "rejected"
    if size>=3:
        if s[size-3] =='b':
            if s[size-2] =='b':
                if s[size-1] =='a':
                    return "accepted"
                return "rejected"
            return "rejected"
        return "rejected"
    return "rejected"
inputs = ['bba','ababba','abba','abb','baba','bbb']
for i in inputs:
    print(FA(i))
```

Program output:

```
In [2]: runfile('D:/abhishek/model college/sem4/NLP/
pract_7_c.py', wdir='D:/abhishek/model college/sem4/
NLP')
accepted
accepted
accepted
rejected
rejected
rejected
```

Practical No:7.d

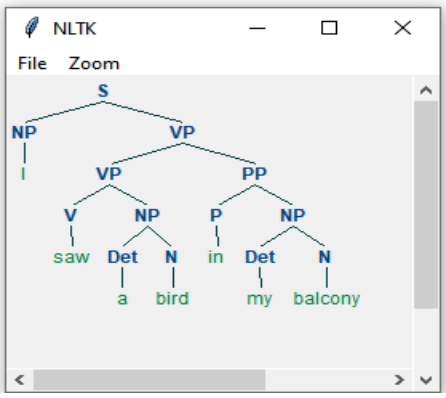
Aim: Implementation of Deductive Chart Parsing using context free grammar and a given sentence.

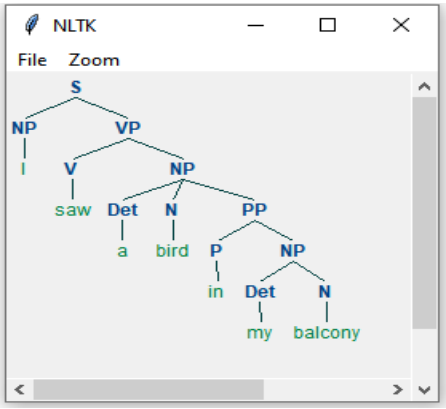
Program code:

```
import nltk

from nltk import tokenize

grammar = 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)

parser = nltk.ChartParser(grammar)

for tree in parser.parse(all_tokens):

   print(tree)

   tree.draw()
```

Program output:

```
In [1]: runfile('D:/abhishek/model college/sem4/
NLP/PRACT_7_D.py', wdir='D:/abhishek/model college/
sem4/NLP')
['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
(S
  (NP I)
  (VP
    (VP (V saw) (NP (Det a) (N bird)))
    (PP (P in) (NP (Det my) (N balcony)))))
(S
  (NP I)
  (VP
    (V saw)
    (NP (Det a) (N bird) (PP (P in) (NP (Det my) (N
balcony))))))
```





Practical No:8

Aim: Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer ,Study WordNetLemmatizer

Program code:

```
import nltk

from nltk.stem import PorterStemmer

word_stemm = PorterStemmer()

print(word_stemm.stem('writing'))

import nltk

from nltk.stem import LancasterStemmer

lanc_stemm = LancasterStemmer()

print(lanc_stemm.stem('writing'))

import nltk

from nltk.stem import RegexpStemmer

Reg_stemm = RegexpStemmer('ing$|s$|e$|able$', min=4)

print(Reg_stemm.stem('writing'))

import nltk

from nltk.stem import SnowballStemmer

english_stemm = SnowballStemmer('english')

print(english_stemm.stem('writing'))

import nltk

from nltk.stem import WordNetLemmatizer

lemetizer = WordNetLemmatizer()

print("word:\t lemma")

print("rocks:", lemetizer.lemmatize("rocks"))

print("corpa:",lemetizer.lemmatize("corpora"))
```

Program output:

```
In [1]: runfile('C:/Users/Aditi/pract_8.py', wdir='C:/Users/
Aditi')
write
writ
writ
write
word:    lemma
rocks: rock
corpa: corpus
```

Practical No:9

Aim: Implement Naive Bayes classifier

Program code:

```
import pandas as pd

import numpy as np

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

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

stemm = PorterStemmer()

corpus = []

for i in range(0, len(data)):

    s1 = re.sub('[a-zA-Z]',repl='', string=data['v2'][i])

    s1.lower()

    s1 = s1.split()

    s1 = [stemm.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

Countvector = CountVectorizer()

x = Countvector.fit_transform(corpus).toarray()

print(x)

y = 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)

from sklearn.naive_bayes import MultinomialNB

multmomial = MultinomialNB()
```

```
multmomial.fit(x_train, y_train)

y_pred = multmomial.predict(x_test)

print(y_pred)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(classification_report(y_test, y_pred))

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

Program output:

```
In [6]: runfile('D:/abhishek/model college/sem4/NLP/
pract_9.py', wdir='D:/abhishek/model college/sem4/NLP')
[[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]]
['ham' 'ham' 'spam' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
              precision    recall  f1-score   support

         ham       0.91      1.00      0.95      1448
        spam       0.97      0.40      0.56       224

    accuracy                           0.92      1672
   macro avg       0.94      0.70      0.76      1672
weighted avg       0.92      0.92      0.90      1672

accuracy_score: 0.9174641148325359
```

Practical No:10a. Speech Tagging:1

Aim: Speech tagging using spacy

Program code:

```
import spacy
sp = spacy.load('en_core_web_sm')
sen = sp(" I like to play football. Ihated it 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('can you google it?')
word = sen[2]
print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')
sen = sp('can you search it on google?')
word = sen[5]
print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')
sen = sp(" I like to play football. Ihated it 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}')
from spacy import displacy
sen = sp(" I like to play football. Ihated it my childhood though")
displacy.serve(sen, style ='dep', options ={'distance':120})
```

Program output:

```
In [10]: runfile('C:/Users/Aditi/untitled3.py', wdir='C:/Users/Aditi')
 I like to play football. Ihated it my childhood though
VERB
VBD
verb, past tense
             SPACE      _SP      whitespace
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      .        punctuation mark, sentence closer
Ihated       VERB       VBD      verb, past tense
it           PRON       PRP      pronoun, personal
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
86.ADV     :1
92.NOUN    :2
94.PART    :1
95.PRON    :3
97.PUNCT   :1
100.VERB    :3
103.SPACE   :1

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

Practical No:10.a.2:

Aim: Speech tagging using nktl

Program code:

```
import nltk

from nltk.corpus import state_union

from nltk.tokenize import PunktSentenceTokenizer

train_text = state_union.raw("D:/abhishek/model college/sem4/NLP/b.txt")

sample_text = state_union.raw("D:/abhishek/model college/sem4/NLP/c.txt")

custom_sent_tokenizer = PunktSentenceTokenizer(train_text)

tokanized = custom_sent_tokenizer.tokenize(sample_text)

def proces():

    try:

        for i in tokanized[:2]:

            words = nltk.word_tokenize(i)

            tagged = nltk.pos_tag(words)

            print(tagged)


    except Exception as e:

        print(str(e))
```

proces()

Program output:

```
In [7]: runfile('D:/abhishek/model college/sem4/NLP/
pract_10_a_2.py', wdir='D:/abhishek/model college/sem4/
NLP')
[('#', '#'), ('-', ':'), ('*', 'SYM'), ('-', ':'),
('coding', 'NN'), (':', ':'), ('utf-8', 'JJ'), ('-',
':'), ('*', 'SYM'), ('-', ':'), ("'''", "'''"), ("'''",
"'''"), ("'''", "'''"), ('Created', 'VBN'), ('on', 'IN'),
('Sat', 'NNP'), ('Apr', 'NNP'), ('29', 'CD'),
('14:39:23', 'CD'), ('2023', 'CD'), ('@', 'NN'),
('author', 'NN'), (':', ':'), ('Aditi', 'NN'), ("'''",
"'''"), ("'''", "'''"), ("'''", "'''"), ('Generally', 'RB'),
(',', ','), ('there', 'EX'), ('is', 'VBZ'), ('no',
'DT'), ('review', 'NN'), ('by', 'IN'), ('a', 'DT'),
('moderator', 'NN'), ('or', 'CC'), ('gatekeeper', 'NN'),
('before', 'IN'), ('modifications', 'NNS'), ('are',
'VBP'), ('accepted', 'VBN'), ('and', 'CC'), ('thus',
'RB'), ('lead', 'VB'), ('to', 'TO'), ('changes', 'NNS'),
('on', 'IN'), ('the', 'DT'), ('website', 'NN'), ('.',
'.')]
[('Many', 'JJ'), ('wikis', 'NNS'), ('are', 'VBP'),
('open', 'JJ'), ('to', 'TO'), ('alteration', 'NN'),
('by', 'IN'), ('the', 'DT'), ('general', 'JJ'),
('public', 'NN'), ('without', 'IN'), ('requiring',
'VBG'), ('registration', 'NN'), ('of', 'IN'), ('user',
'JJ'), ('accounts', 'NNS'), ('.', '.')]
```

Practical No:10.B. Statistical parsing:1

Aim: Usage of Give and Gave in the Penn Treebank sample

Program code:

```
import nltk

import nltk.parse.viterbi

import nltk.parse.pchart


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):

    op = "%s %s: %s/ %s: %s "%(sent(t[0]), t[1].label(), sent(t[1]), t[2].label(), sent(t[2]))

    if len(op)> width:

        op = op[:width] + "..."

    print(op)
```

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

Program output:

```
In [8]: runfile('D:/abhishek/model college/sem4/NLP/
pract_10_b_1.py', wdir='D:/abhishek/model college/sem4/
NLP')
gave NP: thechefs/ NP: astandingovation
gave NP: them/ NP: similarhelp
gave NP: Mitsui/ NP: accesstoahigh-techmedicalproduct
gave NP: quickapproval/ PP-DTV:
to$3.18billioninsupplementalappropriatio...
gave NP: Mr.Thomas/ NP:
onlya``qualified''rating,ratherthan``wellqualifi...
```

Practical No:10.B.2

Aim: probabilistic parser

Program code:

```
import nltk
from nltk import PCFG
gramm = 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]|"yong"[0.6]
                CC -> "and"[0.9] |"or"[0.1]
               ''' )
print(gramm)
viterbi_parser = nltk.ViterbiParser(gramm)
token = "old men and women".split()
obj = viterbi_parser.parse(token)
print("Outpu:t")
for x in obj:
    print(x)
```

Program output:

```
In [9]: runfile('D:/abhishek/model college/sem4/NLP/
pract_10_b_2.py', wdir='D:/abhishek/model college/sem4/
NLP')
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 -> 'yong' [0.6]
    CC -> 'and' [0.9]
    CC -> 'or' [0.1]
Outpu:t
(NP (JJ old) (NNS (NNS men) (CC and) (NNS women)))
(p=0.000864)
```

Practical No:10.C Malt parsing:

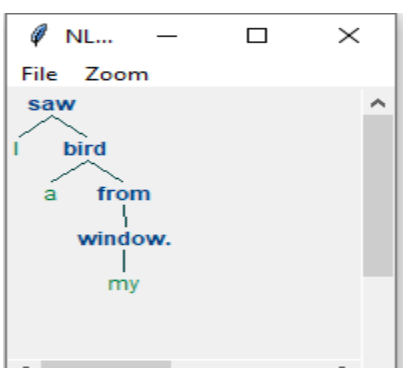
Aim: Parse a sentence and draw a tree using malt parsing.

Program code:

```
import nltk

from nltk.parse import malt

mp = malt.MaltParser('D:/abhishek/model college/sem4/NLP/maltparser-1.7.2','D:/abhishek/model college/sem4/NLP/engmalt.linear-1.7.mco')

t = mp.parse_one('I saw a bird from my window.'.split()).tree()

print(t)

t.draw()
```

Program output:

```
In [2]: runfile('C:/Users/Aditi/untitled1.py', wdir='C:/Users/
Aditi')
(saw I (bird a (from (window. my))))
```



Practical No:11.A

Aim: Multiword Expressions in NLP

Program code:

```
from nltk.tokenize import MWETokenizer

from nltk import sent_tokenize, word_tokenize

s  = "A typical wiki contains multiple pages for the subjects or scope of the project, and could be either open to the public or limited to use within an organization for maintaining its internal knowledge base."

mwe = MWETokenizer([('New','York'),('Hong','Kong')],separator='_')

for sent in sent_tokenize(s):

    print(mwe.tokenize(word_tokenize(sent)))
```

Program output:

```
In [10]: runfile('D:/abhishek/model college/sem4/NLP/
pract_11_a.py', wdir='D:/abhishek/model college/sem4/
NLP')
['A', 'typical', 'wiki', 'contains', 'multiple',
'pages', 'for', 'the', 'subjects', 'or', 'scope', 'of',
'the', 'project', ',', 'and', 'could', 'be', 'either',
'open', 'to', 'the', 'public', 'or', 'limited', 'to',
'use', 'within', 'an', 'organization', 'for',
'maintaining', 'its', 'internal', 'knowledge', 'base',
'.']
```

Practical No:11.B

Aim: Normalized Web Distance and Word Similarity

Program code:

```
import numpy as np

import re

import textdistance

import sklearn

from sklearn.cluster import AgglomerativeClustering

texts =['Reliance supermarket', 'Reliance hypermarket', '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']

def normalize(texts):

    return re.sub('[^a-z0-9]+','',texts.lower())

def grp_text (texts):
```

```
    normalixe_txt = np.array([normalize(text) for text in texts])

    dist = 1 - np.array([[textdistance.jaro_winkler(one, another) for one in normalixe_txt] for another in normalixe_txt])

    clustering = AgglomerativeClustering(distance_threshold=0.4, affinity="precomputed", linkage="complete", n_clusters=None).fit(dist)

    centers = dict()

    for clust_id in set(clustering.labels_):

        ind = clustering.labels_ == clust_id

        center = dist[:,ind][ind].sum(axis=1)

        centers[clust_id] = normalixe_txt[ind][center.argmin()]

    return [centers[i] for i in clustering.labels_]

print(grp_text(texts))
```

Program output:

```
In [11]: runfile('D:/abhishek/model college/sem4/NLP/
pract_11_b.py', wdir='D:/abhishek/model college/sem4/
NLP')
D:\ANA\lib\site-
packages\sklearn\cluster\_agglomerative.py:983:
FutureWarning: Attribute `affinity` was deprecated in
version 1.2 and will be removed in 1.4. Use `metric`
instead
  warnings.warn(
['reliance', 'reliance', 'reliance', 'reliance',
'reliance', 'reliance', 'mumbai', 'mumbai', 'mumbai',
'mumbai', 'kmtrading', 'kmtrading', 'kmtrading',
'kmtrading', 'kmtrading']
```

Practical No:11.C

Aim: Word Sense Disambiguation

Program code:

```
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('blank')
```

```
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))
```

Program output:

```
In [12]: runfile('D:/abhishek/model college/sem4/NLP/
pract_11_c.py', wdir='D:/abhishek/model college/sem4/
NLP')
<bound method Synset.name of Synset('space.n.05')>:
<bound method Synset.definition of Synset('space.n.05')>
<bound method Synset.name of Synset('set.n.01')>: <bound
method Synset.definition of Synset('set.n.01')>
<bound method Synset.name of Synset('put.v.01')>: <bound
method Synset.definition of Synset('put.v.01')>
```

Keraleeya Samajam(Regd.) Dombivli's

MODEL COLLEGE

Re-Accredited Grade "A" by NAAC

Kanchan Goan Village, Khambalpada, Thakurli East – 421201
Contact No – 7045682157, 7045682158. www.model-college.edu.in

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr. /Miss _____

*Studying in Class*_____*Seat No.* _____

*Has completed the prescribed practicals in the subject*_____

*During the academic year*_____

Date : _____

External Examiner

Internal Examiner
M.Sc. Information Technology

Sr.No.	Practical	Date	Signature
1.	Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow.	25/3/2023	
2.	Solving XOR problem using deep feed forward network.	25/3/2023	
3.	Solving XOR problem using deep feed forward network.	1/4/2023	
4.A	Using deep feed forward network with two hidden layers for performing multiclass classification and predicting the class.	1/4/2023	
4.B	Using a deep feed forward network with two hidden layers for performing classification and predicting the probability of class.	1/4/2023	
4.C	Using a deep feed forward network with two hidden layers for performing linear regression and predicting values.	1/4/2023	
5.B	Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.	15/4/2023	
6.	Implementing regularization to avoid overfitting in binary classification.	15/4/2023	
7.	Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.	23/4/2023	
8.	Performing encoding and decoding of images using deep autoencoder.	23/4/2023	
9.	Implementation of convolutional neural network to predict numbers from number images.	29/4/2023	
10.	Denoising images using autoencoder.	29/4/2023	

Practical No: 1

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

Program Code:

```
import tensorflow as tf

print('Matrix Multiplication Demo')

x = tf.constant([1,2,3,4,5,6], shape=[2,3])

print(x)

y = tf.constant([7,8,9,10,11,12], shape=[3,2])

print(y)

z = tf.matmul(x, y)

print('Product:',z)

e_matrix_A = tf.random.uniform([2,2],minval=3,maxval=10,dtype=tf.float32, name='matrixA')

print('Matrix A:\n{}\n\n'.format(e_matrix_A))

eigen_values_A, eigeen_vactor_A = tf.linalg.eigh(e_matrix_A)

print("Eigeen vactor:\n{}\n\n Eigen values:\n{}\n".format(eigeen_vactor_A,eigen_values_A))
```

Program Output:

```
In [7]: runfile('D:/abhishek/model college/sem4/DL/
Pract1.py', wdir='D:/abhishek/model college/sem4/DL')
Matrix Multiplication Demo
tf.Tensor(
[[1 2 3]
 [4 5 6]], shape=(2, 3), dtype=int32)
tf.Tensor(
[[ 7  8]
 [ 9 10]
 [11 12]], shape=(3, 2), dtype=int32)
Product: tf.Tensor(
[[ 58  64]
 [139 154]], shape=(2, 2), dtype=int32)
Matrix A:
[[7.783489  3.592496 ]
 [7.411567  6.7761736]]


Eigeen vactor:
[[-0.6827154   0.73068434]
 [ 0.73068434  0.6827154 ]]

 Eigen values:
[-0.14882919 14.70849   ]
```

Practical No: 2

Aim: Solving XOR problem using deep feed forward network.

Program Code:

```
import numpy as np

from keras.layers import Dense

from keras.models import Sequential

model = Sequential()

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

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

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

print(model.summary())

print(model.get_weights())

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

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

model.fit(x,y, epochs=1000, batch_size =4)

print(model.get_weights())

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

Program Output:

```
In [5]: runfile('D:/abhishek/model college/sem4/DL/pract2.py',
wdir='D:/abhishek/model college/sem4/DL')
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 2)                 6

 dense_7 (Dense)             (None, 1)                 3

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
None
[array([[ 0.701226  ,  0.79736507],
       [-0.77499914, -0.6906795 ]], dtype=float32), array([0.,
0.], dtype=float32), array([[ 0.27735293],
       [-1.3896614 ]], dtype=float32), array([0.],
dtype=float32)]
Epoch 1/1000
1/1 [==============================] - 1s 703ms/step - loss:
0.8147 - accuracy: 0.5000
Epoch 2/1000
1/1 [==============================] - 0s 0s/step - loss: 0.8140
- accuracy: 0.5000
Epoch 3/1000
```

```
- accuracy: 0.7500
Epoch 996/1000
1/1 [==============================] - 0s 16ms/step - loss:
0.5592 - accuracy: 0.7500
Epoch 997/1000
1/1 [==============================] - 0s 0s/step - loss: 0.5591
- accuracy: 0.7500
Epoch 998/1000
1/1 [==============================] - 0s 16ms/step - loss:
0.5590 - accuracy: 0.7500
Epoch 999/1000
1/1 [==============================] - 0s 16ms/step - loss:
0.5589 - accuracy: 0.7500
Epoch 1000/1000
1/1 [==============================] - 0s 0s/step - loss: 0.5587
- accuracy: 0.7500
[array([[-0.7788038 , -0.16185868],
       [ 0.7787089 , -0.67428535]], dtype=float32),
array([-0.00015648,  0.        ], dtype=float32),
array([[1.8830456],
       [0.7679335]], dtype=float32), array([-0.29425845],
dtype=float32)]
1/1 [==============================] - 0s 47ms/step
[[0.42696166]
 [0.7634687 ]
 [0.42696166]
 [0.42696166]]
```

Practical No: 3

Aim: Implementing deep neural network for performing binary classification task.

Program Code:

```
from numpy import loadtxt

from keras.models import Sequential

from keras.layers import Dense

data = loadtxt('pima-indians-diabetes.csv',delimiter=',')

data

x = data[:,0:8]

y = data[:,8]

x

y

model = Sequential()

model.add(Dense(12, input_dim=8,activation ='relu'))

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

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

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

model.fit(x,y, epochs=150,batch_size=10)

_,accuracy = model.evaluate(x,y)

print('Acurracy of model is',(accuracy*100))

prediction = model.predict(x)

exec("for i in range(5):print(x[i].tolist(),prediction[i],y[i])")
```

Program Output:

```
In [7]: runfile('D:/abhishek/model college/sem4/DL/prac3.py',
wdir='D:/abhishek/model college/sem4/DL')
Epoch 1/150
77/77 [==============================] - 1s 2ms/step - loss:
11.1088 - accuracy: 0.4701
Epoch 2/150
77/77 [==============================] - 0s 2ms/step - loss:
2.8592 - accuracy: 0.4622
Epoch 3/150
77/77 [==============================] - 0s 2ms/step - loss:
1.7996 - accuracy: 0.5117
Epoch 4/150
77/77 [==============================] - 0s 2ms/step - loss:
1.3623 - accuracy: 0.5690
Epoch 5/150
77/77 [==============================] - 0s 2ms/step - loss:
1.0873 - accuracy: 0.5508
Epoch 6/150
77/77 [==============================] - 0s 2ms/step - loss:
0.9781 - accuracy: 0.5964
Epoch 7/150
77/77 [==============================] - 0s 2ms/step - loss:
0.9378 - accuracy: 0.5794
Epoch 8/150
77/77 [==============================] - 0s 2ms/step - loss:
0.8382 - accuracy: 0.6146
Epoch 9/150
77/77 [==============================] - 0s 2ms/step - loss:
```

```
Epoch 145/150
77/77 [==============================] - 0s 2ms/step - loss:
0.5460 - accuracy: 0.7500
Epoch 146/150
77/77 [==============================] - 0s 2ms/step - loss:
0.6729 - accuracy: 0.6901
Epoch 147/150
77/77 [==============================] - 0s 2ms/step - loss:
0.5345 - accuracy: 0.7552
Epoch 148/150
77/77 [==============================] - 0s 2ms/step - loss:
0.5485 - accuracy: 0.7487
Epoch 149/150
77/77 [==============================] - 0s 2ms/step - loss:
0.4980 - accuracy: 0.7643
Epoch 150/150
77/77 [==============================] - 0s 2ms/step - loss:
0.5388 - accuracy: 0.7448
24/24 [==============================] - 0s 2ms/step - loss:
0.5445 - accuracy: 0.7448
Acurracy of model is 74.47916865348816
24/24 [==============================] - 0s 2ms/step
[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0] [0.93576294] 1.0
[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0] [0.11852439] 0.0
[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0] [0.9844859] 1.0
[1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0] [0.12498722] 0.0
[0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0] [0.9622241] 1.
0
```

Practical No: 4.A

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

Program Code:

```
from keras.models import Sequential

from keras.layers import Dense

from sklearn.datasets import make_blobs

from sklearn.preprocessing import MinMaxScaler

x,y = make_blobs(n_samples=100, centers=2, n_features =2, random_state=1)

scalar = MinMaxScaler()

scalar.fit(x)

x = scalar.transform(x)

model = Sequential()

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

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

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

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

model.fit(x,y, epochs=500)

xnew,yreal = make_blobs(n_samples=3, centers=2, n_features=2, random_state=1)

xnew = scalar.transform(xnew)

ynew = model.predict(xnew)

for i in range(len(xnew)):

    print('X=%s.predicted=%s, Desired%s'%(xnew[i],ynew[i],yreal[i]))
```

Program Output:

```
4/4 [==============================] - 0s 5ms/step - loss: 0.0031
Epoch 490/500
4/4 [==============================] - 0s 5ms/step - loss: 0.0031
Epoch 491/500
4/4 [==============================] - 0s 0s/step - loss: 0.0030
Epoch 492/500
4/4 [==============================] - 0s 0s/step - loss: 0.0030
Epoch 493/500
4/4 [==============================] - 0s 0s/step - loss: 0.0030
Epoch 494/500
4/4 [==============================] - 0s 0s/step - loss: 0.0030
Epoch 495/500
4/4 [==============================] - 0s 0s/step - loss: 0.0030
Epoch 496/500
4/4 [==============================] - 0s 0s/step - loss: 0.0030
Epoch 497/500
4/4 [==============================] - 0s 0s/step - loss: 0.0029
Epoch 498/500
4/4 [==============================] - 0s 5ms/step - loss: 0.0029
Epoch 499/500
4/4 [==============================] - 0s 5ms/step - loss: 0.0029
Epoch 500/500
4/4 [==============================] - 0s 5ms/step - loss: 0.0029
1/1 [==============================] - 0s 156ms/step
X=[0.89337759 0.65864154].predicted=[0.00432069], Desired0
X=[0.29097707 0.12978982].predicted=[0.9971629], Desired1
X=[0.78082614 0.75391697].predicted=[0.00526416], Desired0
```

Practical No: 4.B

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

Program Code:

```
from keras.models import Sequential

from keras.layers import Dense

from sklearn.datasets import make_blobs

from sklearn.preprocessing import MinMaxScaler

x,y = make_blobs(n_samples=100, n_features=2, random_state=1)

scalar = MinMaxScaler()

scalar.fit(x)

x = scalar.transform(x)

model = Sequential()

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

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

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

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

model.fit(x,y, epochs=500)

xnew, yreal = make_blobs(n_samples=3,n_features=2, centers=2, random_state=1)

xnew = scalar.transform(xnew)

yclass = model.predict(xnew)

ynew= model.predict_on_batch(xnew)

for i in range(len(xnew)):

    print("x=%s.predicted_probablity=%s, predicted_class%s"%(xnew[i],ynew[i],yclass[i]))
```

Program Output:

```
4/4 [==============================] - 0s 5ms/step - loss:
-46.8294
Epoch 495/500
4/4 [==============================] - 0s 5ms/step - loss:
-47.0932
Epoch 496/500
4/4 [==============================] - 0s 5ms/step - loss:
-47.3511
Epoch 497/500
4/4 [==============================] - 0s 5ms/step - loss:
-47.6506
Epoch 498/500
4/4 [==============================] - 0s 5ms/step - loss:
-47.9673
Epoch 499/500
4/4 [==============================] - 0s 0s/step - loss:
-48.2521
Epoch 500/500
4/4 [==============================] - 0s 0s/step - loss:
-48.5630
1/1 [==============================] - 0s 94ms/step
x=[0.89337759 0.74250702].predicted_probablity=[1.],
predicted_class[1.]
x=[0.29097707 0.3435844 ].predicted_probablity=[1.],
predicted_class[1.]
x=[0.78082614 0.81437503].predicted_probablity=[0.999955],
predicted_class[0.999955]
```

Practical No: 4.c

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

Program Code:

```
from keras.models import Sequential

from keras.layers import Dense

from sklearn.datasets import make_regression

from sklearn.preprocessing import MinMaxScaler

x,y = make_regression(n_samples=100, n_features=2,noise=0.1,random_state=1)

scalarx, scalary = MinMaxScaler(), MinMaxScaler()

scalarx.fit(x)

scalary.fit(y.reshape(100, 1))

x = scalarx.transform(x)

y = scalary.transform(y.reshape(100, 1))

model = Sequential()

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

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

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

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

model.fit(x,y, epochs=1000, verbose=0)

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

xnew = scalarx.transform(xnew)

ynew = model.predict(xnew)

for i in range(len(xnew)):

    print("x =%s, Predicted=%s"%(xnew[i], ynew[i]))
```

Program Output:

```
In [3]: runfile('D:/abhishek/model college/sem4/DL/pract4c.py',
wdir='D:/abhishek/model college/sem4/DL')
1/1 [==============================] - 0s 82ms/step
x =[0.29466096 0.30317302], Predicted=[0.18272518]
x =[0.39445118 0.79390858], Predicted=[0.75779283]
x =[0.02884127 0.6208843 ], Predicted=[0.39201856]
```

Practical No: 5.A

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

Program Code:

Program Output:

Practical No: 5.B

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

Program Code:

```
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
df = pd.read_csv('D:/abhishek/model college/sem4/DL/flowers.csv')
print(df)
x = df.iloc[:,0:4].astype(float)
y = df.iloc[:,4]
print(x)
print(y)
encoder = LabelEncoder()
encoder.fit(y)
encoder_y = encoder.transform(y)
print(encoder_y)
dummy = np_utils.to_categorical(encoder_y)
print(dummy)
def baseline_model():
    model = Sequential()
    model.add(Dense(8, input_dim=4, activation='relu'))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
    return model
```

```
estimator = baseline_model()

estimator.fit(x, dummy, epochs=100, shuffle=True)

action = estimator.predict(x)

for i in range(25):

    print(dummy[i])

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

for i in range(25):

    print(action[i])
```

Program Output:

```
In [45]: runfile('D:/abhishek/model college/sem4/DL/untitled2.py',
wdir='D:/abhishek/model college/sem4/DL')
     sepal length  sepal width  petal length  petal width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

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

```
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
^^^^^^^^^^^^^^^^^^
[0.5007003  0.16068313 0.3386165 ]
[0.5321626  0.16876343 0.2990739 ]
[0.505182   0.16843778 0.32638016]
[0.48901486 0.18401408 0.32697108]
[0.48258665 0.16292574 0.3544876 ]
[0.4283684  0.17038709 0.40124458]
[0.45275995 0.17857163 0.36866838]
[0.49356377 0.1698191  0.33661705]
[0.5000019  0.185414   0.31458414]
[0.53005666 0.17110258 0.29884082]
[0.49744782 0.1564966  0.34605563]
[0.46835616 0.18175757 0.34988627]
```

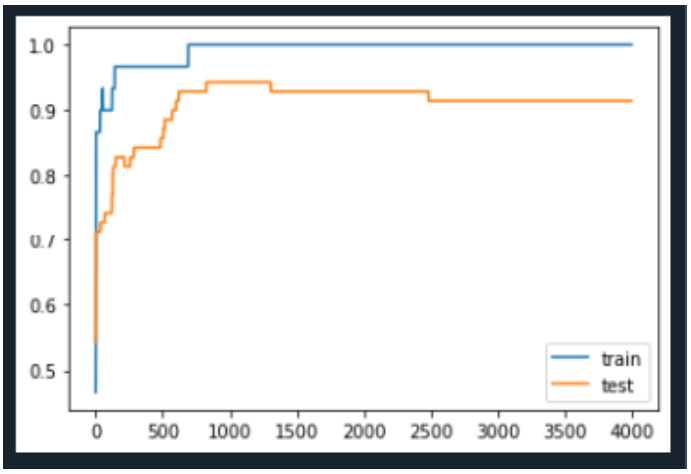
Practical No: 6

Aim: Implementing regularization to avoid overfitting in binary classification.

Program Code:

```
import matplotlib.pyplot as plt

from sklearn.datasets import make_moons

from keras.models import Sequential

from keras.layers import Dense

x,y = make_moons(n_samples=100, noise=0.2, random_state=1)

n_train = 30

x_train, x_test = x[:n_train,:],x[n_train:]

y_train, y_test = y[:n_train],y[n_train:]

model = Sequential()

model.add(Dense(500,input_dim=2,activation='relu'))

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

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

history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=4000)

plt.plot(history.history['accuracy'], label='train')

plt.plot(history.history['val_accuracy'],label='test')

plt.legend()

plt.show()
```

Program Output:



```
In [4]: runfile('D:/abhishek/model college/sem4/DL/pract6.py',
wdir='D:/abhishek/model college/sem4/DL')
Epoch 1/4000
1/1 [==============================] - 1s 1s/step - loss: 0.7011
- accuracy: 0.5000 - val_loss: 0.6905 - val_accuracy: 0.4714
Epoch 2/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.6845 - accuracy: 0.5333 - val_loss: 0.6797 - val_accuracy:
0.5000
Epoch 3/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.6683 - accuracy: 0.7000 - val_loss: 0.6691 - val_accuracy:
0.6286
Epoch 4/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.6526 - accuracy: 0.8333 - val_loss: 0.6590 - val_accuracy:
0.6857
Epoch 5/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.6373 - accuracy: 0.8333 - val_loss: 0.6492 - val_accuracy:
0.6857
Epoch 6/4000
1/1 [==============================] - 0s 31ms/step - loss:
0.6224 - accuracy: 0.8333 - val_loss: 0.6398 - val_accuracy:
0.6857
Epoch 7/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.6079 - accuracy: 0.8333 - val_loss: 0.6308 - val_accuracy:
```

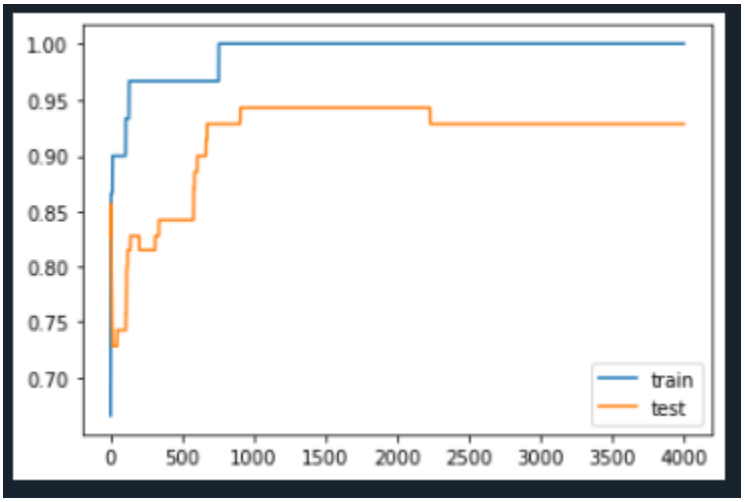
L2 regularization

Program Code:

```
import matplotlib.pyplot as plt

from sklearn.datasets import make_moons

from keras.models import Sequential

from keras.layers import Dense

from keras.regularizers import l2

x,y = make_moons(n_samples=100, noise=0.2, random_state=1)

n_train = 30

x_train, x_test = x[:n_train,:],x[n_train:]

y_train, y_test = y[:n_train],y[n_train:]

model = Sequential()

model.add(Dense(500,input_dim=2,activation='relu', kernel_regularizer=l2(0.0001)))

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

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

history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=4000)

plt.plot(history.history['accuracy'], label='train')
```

```
plt.plot(history.history['val_accuracy'],label='test')

plt.legend()

plt.show()
```

Program Output:




```
Epoch 3995/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.0048 - accuracy: 1.0000 - val_loss: 0.3416 - val_accuracy:
0.9286
Epoch 3996/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.0048 - accuracy: 1.0000 - val_loss: 0.3417 - val_accuracy:
0.9286
Epoch 3997/4000
1/1 [==============================] - 0s 62ms/step - loss:
0.0048 - accuracy: 1.0000 - val_loss: 0.3417 - val_accuracy:
0.9286
Epoch 3998/4000
1/1 [==============================] - 0s 62ms/step - loss:
0.0048 - accuracy: 1.0000 - val_loss: 0.3417 - val_accuracy:
0.9286
Epoch 3999/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.0048 - accuracy: 1.0000 - val_loss: 0.3416 - val_accuracy:
0.9286
Epoch 4000/4000
1/1 [==============================] - 0s 47ms/step - loss:
0.0048 - accuracy: 1.0000 - val_loss: 0.3416 - val_accuracy:
0.9286
```

Practical No: 7

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

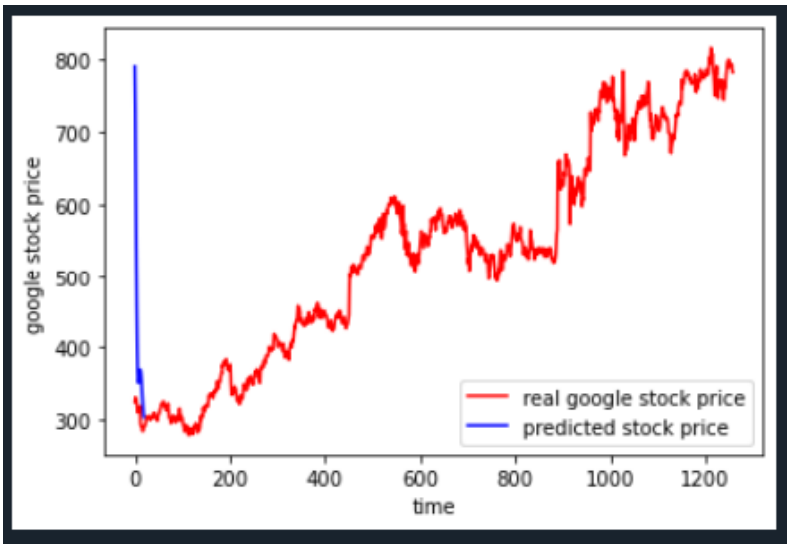
Program Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv('D:/abhishek/model college/sem4/DL/Google_Stock_Price_Train.csv')
train_set = df.iloc[:,1:2].values
sc = MinMaxScaler(feature_range=(0,1))
train_set_scal = sc.fit_transform(train_set)
x_train =[]
y_train =[]
for i in range(60, 1258):
    x_train.append(train_set_scal[i - 60:i,0])
    y_train.append(train_set_scal[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)
print(x_train)
print('*************************')
print(y_train)
x_train = np.reshape(x_train,(x_train.shape[0], x_train.shape[1],1))
print('*************************')
print(x_train)
regrssor = Sequential()
regrssor.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
```

```
regrssor.add(Dropout(0.2))
regrssor.add(LSTM(units=50, return_sequences=True))
regrssor.add(Dropout(0.2))
regrssor.add(LSTM(units=50, return_sequences=True))
regrssor.add(Dropout(0.2))
regrssor.add(LSTM(units=50))
regrssor.add(Dropout(0.2))
regrssor.add(Dense(units=1))
regrssor.compile(optimizer='adam', loss='mean_squared_error')
regrssor.fit(x_train, y_train, epochs=100, batch_size=32)
dataset_test = pd.read_csv('D:/abhishek/model college/sem4/DL/Google_Stock_Price_Train.csv')
real_stock_price = dataset_test.iloc[:,1:2].values
dataset_total = pd.concat((df['Open'],dataset_test['Open']), axis=0)
iputs = dataset_total[len(dataset_total)-len(dataset_test)-60:].values
iputs = iputs.reshape(-1,1)
iputs= sc.transform(iputs)
x_test =[]
for i in range(60,80):
    x_test.append(iputs[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 = regrssor.predict(x_test)
predicted_stock_price= sc.inverse_transform(predicted_stock_price)
plt.plot(real_stock_price, color='red',label='real google stock price')
plt.plot(predicted_stock_price, color='blue', label='predicted stock price')
plt.xlabel('time')
plt.ylabel('google stock price')
plt.legend()
plt.show()
```

Program Output:

```
In [46]: runfile('D:/abhishek/model college/sem4/DL/pract_7.py', wdir='D:/
abhishek/model college/sem4/DL')
[[0.08581368 0.09701243 0.09433366 ... 0.07846566 0.08034452 0.08497656]
 [0.09701243 0.09433366 0.09156187 ... 0.08034452 0.08497656 0.08627874]
 [0.09433366 0.09156187 0.07984225 ... 0.08497656 0.08627874 0.08471612]
 ...
 [0.92106928 0.92438053 0.93048218 ... 0.95475854 0.95204256 0.95163331]
 [0.92438053 0.93048218 0.9299055  ... 0.95204256 0.95163331 0.95725128]
 [0.93048218 0.9299055  0.93113327 ... 0.95163331 0.95725128 0.93796041]]
*************************
[0.08627874 0.08471612 0.07454052 ... 0.95725128 0.93796041 0.93688146]
*************************
[[[0.08581368]
  [0.09701243]
  [0.09433366]
  ...
  [0.07846566]
  [0.08034452]
  [0.08497656]]

 [[0.09701243]
  [0.09433366]
  [0.09156187]
  ...
  [0.08034452]
  [0.08497656]
  [0.08627874]]

 [[0.09433366]
```



Practical No: 8

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

Program Code:

```
import keras

from keras import layers

from keras.datasets import mnist

import numpy as np

encoding_dim =32

input_img = keras.Input(shape=(784,))

encoded = layers.Dense(encoding_dim, activation='relu')(input_img)

decoded = layers.Dense(784, activation='sigmoid')(encoded)

autoencoder = keras.Model(input_img, decoded)

encoder = keras.Model(input_img, encoded)

encoded_input = keras.Input(shape=(encoding_dim,))

decoded_layer = autoencoder.layers[-1]

decoder = keras.Model(encoded_input, decoded_layer(encoded_input))

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

(x_train,_),(x_test,_)=mnist.load_data()

x_train=x_train.astype('float32')/255.

x_test=x_test.astype('float32')/255.

x_train = x_train.reshape((len(x_train),np.prod(x_train.shape[1:])))

x_test = x_test.reshape((len(x_test),np.prod(x_test.shape[1:])))

print(x_train.shape)

print(x_test.shape)

autoencoder.fit(x_train, x_train, epochs=50, batch_size = 256, shuffle=True,
validation_data=(x_test,x_test))

encoded_imgs = encoder.predict(x_test)

decoded_imgs = decoder.predict(encoded_imgs)

import matplotlib.pyplot as plt
```

```
n = 10
plt.figure(figsize=(40,4))
for i in range(10):
    ax = plt.subplot(3, 20, i+1)
    plt.imshow(x_test[i].reshape(28,28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    ax= plt.subplot(3,20,i+1+20)
    plt.imshow(encoded_imgs[i].reshape(8,4))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    ax= plt.subplot(3,20,2*20+i+1)
    plt.imshow(decoded_imgs[i].reshape(28,28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

Program Output:

```
235/235 [==============================] - 2s 9ms/step -
loss: 0.0928 - val_loss: 0.0917
Epoch 45/50
235/235 [==============================] - 2s 8ms/step -
loss: 0.0927 - val_loss: 0.0917
Epoch 46/50
235/235 [==============================] - 2s 8ms/step -
loss: 0.0927 - val_loss: 0.0916
Epoch 47/50
235/235 [==============================] - 2s 8ms/step -
loss: 0.0927 - val_loss: 0.0917
Epoch 48/50
235/235 [==============================] - 2s 8ms/step -
loss: 0.0927 - val_loss: 0.0916
Epoch 49/50
235/235 [==============================] - 2s 8ms/step -
loss: 0.0927 - val_loss: 0.0916
Epoch 50/50
235/235 [==============================] - 2s 8ms/step -
loss: 0.0927 - val_loss: 0.0916
313/313 [==============================] - 1s 2ms/step
313/313 [==============================] - 1s 2ms/step
```

Practical No: 9

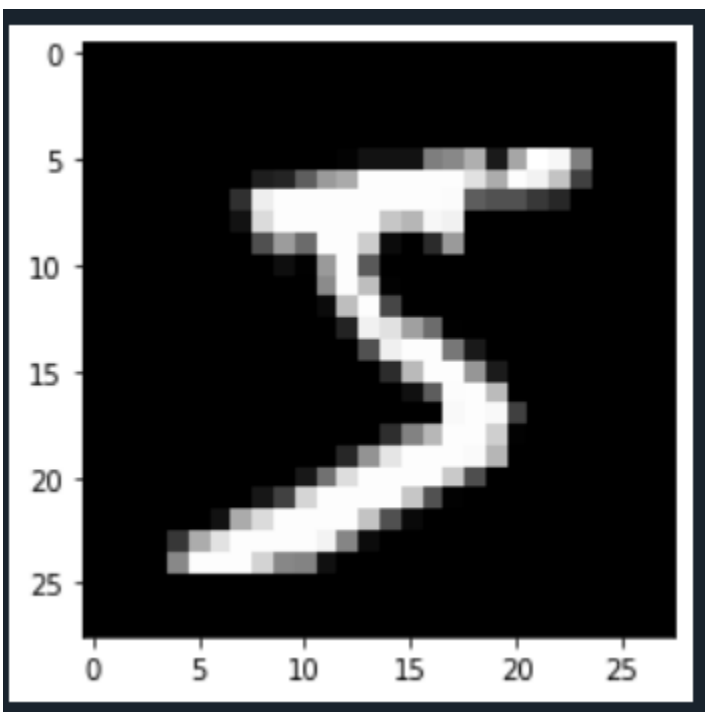
Aim: Implementation of convolutional neural network to predict numbers from number images.

Program Code:

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

(x_train, y_train),(x_test,y_test)= mnist.load_data()
plt.imshow(x_train[0])
plt.show()
print(x_train[0].shape)
x_train = x_train.reshape(60000,28,28,1)
x_test = x_test.reshape(10000,28,28,1)
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_train[0]
print(y_train[0])
model = Sequential()
model.add(Conv2D(64, kernel_size=3 , activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics =['accuracy'])
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=3)
print(model.predict(x_test[:4]))
print(y_test[:4])
```

Program Output:

```
In [56]: runfile('D:/abhishek/model college/sem4/DL/untitled5.py', wdir='D:/
abhishek/model college/sem4/DL')
(28, 28)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
Epoch 1/3
1875/1875 [==============================] - 178s 93ms/step - loss: 0.2411 -
accuracy: 0.9503 - val_loss: 0.1181 - val_accuracy: 0.9626
Epoch 2/3
1875/1875 [==============================] - 181s 97ms/step - loss: 0.0693 -
accuracy: 0.9784 - val_loss: 0.0873 - val_accuracy: 0.9754
Epoch 3/3
1875/1875 [==============================] - 208s 111ms/step - loss: 0.0462
- accuracy: 0.9854 - val_loss: 0.0920 - val_accuracy: 0.9751
1/1 [==============================] - 0s 414ms/step
[[4.5597323e-10 6.7393092e-14 4.9627706e-09 6.4531257e-08 2.0324411e-12
  1.4048982e-13 1.4055956e-16 9.9999988e-01 1.5282581e-09 2.3608365e-08]
 [1.5571613e-09 1.3101253e-07 9.9999988e-01 2.7155525e-10 2.3047625e-12
  6.6412994e-13 1.4332450e-08 3.0330026e-17 6.5343841e-09 3.3060760e-16]
 [3.4105799e-08 9.9737287e-01 3.2505812e-07 7.6057154e-11 6.9392280e-04
  2.8499589e-06 2.4637137e-08 2.9466214e-08 1.9299509e-03 3.6389749e-11]
 [1.0000000e+00 3.7858819e-13 1.1553168e-08 6.1621876e-15 3.4441179e-11
  1.0434328e-09 9.9566000e-10 4.4413364e-15 8.2206941e-12 2.5335983e-10]]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```



Practical No: 10

Aim: Denoising images using autoencoder.

Program Code:

```
import keras
from keras.datasets import mnist
from keras import layers
import numpy as np
from keras.callbacks import TensorBoard
import matplotlib.pyplot as plt

(x_train,_),(x_test,_)= mnist.load_data()
x_train = x_train.astype('float32')/255.
x_test = x_test.astype('float32')/255.
x_train = np.reshape(x_train, (len(x_train),28,28,1))
x_test = np.reshape(x_test, (len(x_test),28,28,1))

noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_test.shape)
x_train_noisy = np.clip(x_train_noisy,0.,1.)
x_test_noisy = np.clip(x_test_noisy,0.,1.)
n = 10
plt.figure(figsize=(20,2))
for i in range(1, n +1):
   ax = plt.subplot(1,n,i)
   plt.imshow(x_test_noisy[i].reshape(28,28))
   plt.gray()
   ax.get_xaxis().set_visible(False)
   ax.get_yaxis().set_visible(False)
```

```
plt.show()


input_im = keras.Input(shape=(28,28,1))

x = layers.Conv2D(33,(3,3), activation='relu', padding='same')(input_im)

x = layers.MaxPooling2D((2,2), padding='same')(x)

x = layers.Conv2D(32,(3,3), activation='relu', padding='same')(x)

encoded = layers.MaxPooling2D((2,2), padding='same')(x)

x = layers.Conv2D(32,(3,3), activation='relu',padding='same')(encoded)

x = layers.UpSampling2D((2,2))(x)

x = layers.Conv2D(32,(3,3), activation='relu', padding='same')(x)

x = layers.UpSampling2D((2,2))(x)

decoded = layers.Conv2D(1,(3,3), activation='sigmoid', padding='same')(x)

autoencoder = keras.Model(input_im, decoded)

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

autoencoder.fit(x_train_noisy, x_train, epochs=3, batch_size = 128,
shuffle=True,validation_data=(x_test_noisy,x_test), callbacks=[TensorBoard(log_dir='/tmo/tb',
histogram_freq=0, write_graph=False)])

predictions = autoencoder.predict(x_test_noisy)

m = 10

plt.figure(figsize=(20,2))

for i in range(1, m+1):

    ax = plt.subplot(1, m ,i)

    plt.imshow(predictions[i].reshape(28,28))

    plt.gray()

    ax.get_xaxis().set_visible(False)

    ax.get_yaxis().set_visible(False)

plt.show()
```

Program Output:

```
In [49]: runfile('D:/abhishek/model college/sem4/DL/untitled3.py',
wdir='D:/abhishek/model college/sem4/DL')
Epoch 1/3
469/469 [==============================] - 123s 248ms/step - loss: 0.1670
- val_loss: 0.1170
Epoch 2/3
469/469 [==============================] - 127s 271ms/step - loss: 0.1139
- val_loss: 0.1109
Epoch 3/3
469/469 [==============================] - 123s 262ms/step - loss: 0.1082
- val_loss: 0.1054
313/313 [==============================] - 6s 18ms/step
```