Index

Sr. No	Topic	Date	Sign
1	**Practical 1: Write the following programs for Blockchain in Python:** 1. A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it 2. A transaction class to send and receive money and test it.		
2	**Practical 2: Write the following programs for Blockchain in Python:** 1. Create multiple transactions and display them. 2. Create a blockchain, a genesis block and execute it.		
3	**Practical 3: Write the following programs for Blockchain in Python:** 1. Create a mining function and test it. 2. Add blocks to the miner and dump the blockchain.		
4	**Practical 4: Implement and demonstrate the use of the following in Solidity:** 1. Variable 2. Operators 3. Loops 4. Decision Making 5. Strings		
5	**Practical 5: Implement and demonstrate the use of the following in Solidity:** 1. Arrays 2. Enums 3. Structs 4. Mappings		
6	**Practical 6: Implement and demonstrate the use of the following in Solidity:** 1. Functions 2. View Functions 3. Pure Functions 4. Fallback Functions 5. Function Overloading 6. Mathematical Functions 7. Cryptographic Functions		
7	**Practical 7: Implement and demonstrate the use of the following in Solidity:** 1. Contracts 2. Inheritance 3. Constructors 4. Abstract Class 5. Interfaces		
8	**Practical 8: Implement and demonstrate the use of the following in Solidity:** 1. Libraries 2. Assembly 3. Events 4. Error Handling		

PRACTICAL 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.

```
# import libraries
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
 # following imports are required by PKI
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
import binascii
class Client:
  def __init__(self):
    random = Crypto.Random.new().read
    self._private_key = RSA.generate(1024, random)
```

```
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)
    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')
class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()
    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
        return collections.OrderedDict({
            'sender': identity,
            'recipient': self.recipient,
            'value': self.value,
            'time' : self.time})
    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')
Dinesh = Client()
Ramesh = Client()
t = Transaction(Dinesh,Ramesh.identity,5.0)
```

signature = t.sign_transaction()

print (signature)

OUTPUT:

```
2e856a639ce96bfcefde832706fc6769ebe745c128bf68851f7d66811307fb84e618c0e5bb1a0796
457bf027f32ef31dc974179e5972ec0c341badea9649c83308f147a3a9a001a5838b0e3bafc83aa2
e0215590f60d0b4fe63d547791a421b88c959b239bd47d6b47854017f40c421ccbb31d7ff5de9c79
727b57e909c92043
```

PRACTICAL 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.

```
def display_transaction(transaction):
  #for transaction in transactions:
  dict = transaction.to_dict()
  print ("sender: " + dict['sender'])
  print ('-----')
  print ("recipient: " + dict['recipient'])
  print ('-----')
  print ("value: " + str(dict['value']))
  print ('-----')
  print ("time: " + str(dict['time']))
  print ('-----')
transactions = []
Dinesh = Client()
Ramesh = Client()
Seema = Client()
Vijay = Client()
t1 = Transaction(
  Dinesh,
  Ramesh.identity,
  15.0
)
t1.sign_transaction()
transactions.append(t1)
t2 = Transaction(
```

```
    Dinesh,
    Seema.identity,
    6.0
)
t2.sign_transaction()
transactions.append(t2)
t3 = Transaction(
    Ramesh,
    Vijay.identity,
    2.0
)
t3.sign_transaction()
transactions.append(t3)
t4 = Transaction(
    Seema,
    Ramesh.identity,
    4.0
)
t4.sign_transaction()
transactions.append(t4)
t5 = Transaction(
    Vijay,
    Seema.identity,
    7.0
)
t5.sign_transaction()
transactions.append(t5)
t6 = Transaction(
    Ramesh,
```

```
    Seema.identity,
    3.0
)
t6.sign_transaction()
transactions.append(t6)
t7 = Transaction(
    Seema,
    Dinesh.identity,
    8.0
)
t7.sign_transaction()
transactions.append(t7)
t8 = Transaction(
    Seema,
    Ramesh.identity,
    1.0
)
t8.sign_transaction()
transactions.append(t8)
t9 = Transaction(
    Vijay,
    Dinesh.identity,
    5.0
)
t9.sign_transaction()
transactions.append(t9)
t10 = Transaction(
    Vijay,
    Ramesh.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)
```

OUTPUT:

```
9cfb905700b0d54bab243e9a04613e7c50277cc29d1c8ea923afe7c3ece47ffbd8e27e1cdd799fe1a3e3019d
09fb98481aaec375e8b7773de13fb7e9570504a84fe88ada587dd4203db7fbe92c51b91d0b37ffa06f42df65
3c232f974e3dd519d5ca69e9d9006e4be4f306ac649f6956a9d3694287e6c0e562d451902d715b3e
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100af6e0b363cb8fee715a3db
e31dbafe0992d2794753c70fa2ea4018e671b9bf4e1f03432c55911b6ce62c70b33e7cf78ed2a027afde9a59
7e13191a583a63f9e3eca758ea0f3d1662f439077e3584b013960955837a604e85fa9d10bdac23e47f121ea2
2a80e9fc253ce513c892f1c00b7a8881e5858430dfdf55f9ea7b79abf70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100be5be235223aa1ada9d
a69bbcd7b6100d873b2fdf69b8a736d0fa6c5b124c1965869478de41671c2a8aded16d14486e8c8b04f3738d
fad382f38940eef9154d1ce4a373e2aa16628703609d1f89fd38b37ee3031ad2ba207b915bb2aee450b05e5f
7d446ebab5bc8eef3bf1234cf1e15addb7b0d8d35bda00981325e6db256f50203010001
-----
value: 15.0
-----
time: 2025-03-06 11:18:10.342471
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100af6e0b363cb8fee715a3db
e31dbafe0992d2794753c70fa2ea4018e671b9bf4e1f03432c55911b6ce62c70b33e7cf78ed2a027afde9a59
7e13191a583a63f9e3eca758ea0f3d1662f439077e3584b013960955837a604e85fa9d10bdac23e47f121ea2
2a80e9fc253ce513c892f1c00b7a8881e5858430dfdf55f9ea7b79abf70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100aa9d0a008dd28ba1d53
c0ebd3c291e0ed9d85641107bfc3c41a4431465785b52ae5af7e0dbcc07d4e72336dabc023cba3e308e611f6
87014d66b1c22d670be815e94748f023fd21ef5b54eae71a3ce65aa74dac5a7bcb652ca41b602b03df851fd1
4428b9f06dfceac923ede2ae5fc209fb05b580b58685eb9c2e0d98c5ef3e70203010001
-----
value: 6.0
-----
time: 2025-03-06 11:18:10.345992
```

PRACTICAL 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.

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

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

block.previous_block_hash = last_block_hash
block.Nonce = mine (block, 2)
digest = hash (block)

TPCoins.append (block)
last_block_hash = digest
dump_blockchain(TPCoins)
```

OUTPUT:

```
b990ab9f17edbe757bc7167ce776b8f7fe1d3e569ba9b10b751cfa5b20e58101b2ce851170ef52519b36748791a7ff4182d88be8e4f623221d5256096bd4ea7763f753eb400175013cb642b5c95262f89157fe12eb4ffc6d1b0f0ae6
e632d4dc47954ab39e22c90827d15bda960b26e7b51fd87821a12574ece064b3c2cedc54
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b0407b8a1cc8680635a26abc8ba2109b32da4eccf7de91145dbd33066472a9167f5328edd8eebf35e1ab90445ab46165aac3742604480f7df3866f
3427cac227987878ada842b236d52c4a2291acf26e74a6028180f475d32e045324de9c204ea7b5788ee5cd6a628fc7d97984b9e882f6f2e5e5f60e5b262ecb1b2d1b2426c70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100dd888fe475bca816c3ded5bf52c45d4e1ad62608ef76ba98f22ba8677c74a03ba63d62949a947fa5bd4a362425101708a1ff0416c07e9b9095f
67d996a367ccd5a2cb75c890cca589a1040bcfac49d8da154af08a5ad8804489f638c9b8fb1231fb6140881ec430b6976a7758db6918e2c55cb09fe1e669d4e6cd652f143ff0d0203010001
-----
value: 15.0
-----
time: 2025-03-06 11:35:19.425027
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b0407b8a1cc8680635a26abc8ba2109b32da4eccf7de91145dbd33066472a9167f5328edd8eebf35e1ab90445ab46165aac3742604480f7df3866f
3427cac227987878ada842b236d52c4a2291acf26e74a6028180f475d32e045324de9c204ea7b5788ee5cd6a628fc7d97984b9e882f6f2e5e5f60e5b262ecb1b2d1b2426c70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ff
d2eff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
value: 6.0
-----
time: 2025-03-06 11:35:19.427540
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100dd888fe475bca816c3ded5bf52c45d4e1ad62608ef76ba98f22ba8677c74a03ba63d62949a947fa5bd4a362425101708a1ff0416c07e9b9095f67d
996a367ccd5a2cb75c890cca589a1040bcfac49d8da154af08a5ad8804489f638c9b8fb1231fb6140881ec430b6976a7758db6918e2c55cb09fe1e669d4e6cd652f143ff0d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100dc1dc151afedb461ce2e4d8e2233f1d8b17896d4f1a5b1b0c5382d029360ea4c218b00afef3f2eda61362eaaeab6fd70ab0d0f42d092edadd9c
40b308f9a4cd7f0fc0da99f22c18f966cd4ef8c27a6e4ad407783573f9ed3a873e666f784d3897b7cddf6cec7360418031076387a9bb60432bb19e58db3c8a4efb01e1f16e1890203010001
-----
value: 2.0
-----
time: 2025-03-06 11:35:19.429553
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ffd2e
ff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100dd888fe475bca816c3ded5bf52c45d4e1ad62608ef76ba98f22ba8677c74a03ba63d62949a947fa5bd4a362425101708a1ff0416c07e9b9095f
67d996a367ccd5a2cb75c890cca589a1040bcfac49d8da154af08a5ad8804489f638c9b8fb1231fb6140881ec430b6976a7758db6918e2c55cb09fe1e669d4e6cd652f143ff0d0203010001
-----
value: 4.0
```

```
time: 2025-03-06 11:35:19.433082
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100dc1dc151afedb461ce2e4d8e2233f1d8b17896d4f1a5b1b0c5382d029360ea4c218b00afef3f2eda61362eaaeab6fd70ab0d0f42d092edadd9c40b
308f9a4cd7f0fc0da99f22c18f966cd4ef8c27a6e4ad407783573f9ed3a873e666f784d3897b7cddf6cec7360418031076387a9bb60432bb19e58db3c8a4efb01e1f16e1890203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ff
d2eff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
value: 7.0
-----
time: 2025-03-06 11:35:19.435095
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100dd888fe475bca816c3ded5bf52c45d4e1ad62608ef76ba98f22ba8677c74a03ba63d62949a947fa5bd4a362425101708a1ff0416c07e9b9095f67d
996a367ccd5a2cb75c890cca589a1040bcfac49d8da154af08a5ad8804489f638c9b8fb1231fb6140881ec430b6976a7758db6918e2c55cb09fe1e669d4e6cd652f143ff0d0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ff
d2eff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
value: 3.0
-----
time: 2025-03-06 11:35:19.437107
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ffd2e
ff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b0407b8a1cc8680635a26abc8ba2109b32da4eccf7de91145dbd33066472a9167f5328edd8eebf35e1ab90445ab46165aac3742604480f7df38
66f3427cac227987878ada842b236d52c4a2291acf26e74a6028180f475d32e045324de9c204ea7b5788ee5cd6a628fc7d97984b9e882f6f2e5e5f60e5b262ecb1b2d1b2426c70203010001
-----
value: 8.0
-----
time: 2025-03-06 11:35:19.439118
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ffd2e
ff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100dd888fe475bca816c3ded5bf52c45d4e1ad62608ef76ba98f22ba8677c74a03ba63d62949a947fa5bd4a362425101708a1ff0416c07e9b9095f
67d996a367ccd5a2cb75c890cca589a1040bcfac49d8da154af08a5ad8804489f638c9b8fb1231fb6140881ec430b6976a7758db6918e2c55cb09fe1e669d4e6cd652f143ff0d0203010001
-----
value: 1.0
-----
time: 2025-03-06 11:35:19.441132
-----
```

```
308f9a4cd7f0fc0da99f22c18f966cd4ef8c27a6e4ad407783573f9ed3a873e666f784d3897b7cddf6cec7360418031076387a9bb60432bb19e58db3c8a4efb01e1f16e1890203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b0407b8a1cc8680635a26abc8ba2109b32da4eccf7de91145dbd33066472a9167f5328edd8eebf35e1ab90445ab46165aac3742604480f7df38
66f3427cac227987878ada842b236d52c4a2291acf26e74a6028180f475d32e045324de9c204ea7b5788ee5cd6a628fc7d97984b9e882f6f2e5e5f60e5b262ecb1b2d1b2426c70203010001
-----
value: 5.0
-----
time: 2025-03-06 11:35:19.443144
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100dc1dc151afedb461ce2e4d8e2233f1d8b17896d4f1a5b1b0c5382d029360ea4c218b00afef3f2eda61362eaaeab6fd70ab0d0f42d092edadd9c40b
308f9a4cd7f0fc0da99f22c18f966cd4ef8c27a6e4ad407783573f9ed3a873e666f784d3897b7cddf6cec7360418031076387a9bb60432bb19e58db3c8a4efb01e1f16e1890203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100dd888fe475bca816c3ded5bf52c45d4e1ad62608ef76ba98f22ba8677c74a03ba63d62949a947fa5bd4a362425101708a1ff0416c07e9b9095f
67d996a367ccd5a2cb75c890cca589a1040bcfac49d8da154af08a5ad8804489f638c9b8fb1231fb6140881ec430b6976a7758db6918e2c55cb09fe1e669d4e6cd652f143ff0d0203010001
-----
value: 3.0
-----
time: 2025-03-06 11:35:19.445154
-----
--------------
Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b0415c04c96249541731a337c6fc1cf95e26d64f881a57b603b596e2a15ee0aea70339d881b85c37c0bc2a58613f3fc0e21501ffc0758b4d98e
086f8021991d39ca14292b66d84f4eb046ecb052ee4640015559a778d9fd4260412d4642fc063ee0c92e2a7fc8eb94dd382e01a3147b4bb811fa0ce1991dc262f67f359e6ef470203010001
-----
value: 500.0
-----
time: 2025-03-06 11:35:21.749563
-----
--------------
====================================
Number of blocks in the chain: 4
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b0415c04c96249541731a337c6fc1cf95e26d64f881a57b603b596e2a15ee0aea70339d881b85c37c0bc2a58613f3fc0e21501ffc0758b4d98e
086f8021991d39ca14292b66d84f4eb046ecb052ee4640015559a778d9fd4260412d4642fc063ee0c92e2a7fc8eb94dd382e01a3147b4bb811fa0ce1991dc262f67f359e6ef470203010001
-----
value: 500.0
-----
time: 2025-03-06 11:35:21.749563
-----
--------------
====================================
```

```
block # 3
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ffd2e
ff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b0407b8a1cc8680635a26abc8ba2109b32da4eccf7de91145dbd33066472a9167f5328edd8eebf35e1ab90445ab46165aac3742604480f7df38
66f3427cac227987878ada842b236d52c4a2291acf26e74a6028180f475d32e045324de9c204ea7b5788ee5cd6a628fc7d97984b9e882f6f2e5e5f60e5b262ecb1b2d1b2426c70203010001
-----
value: 8.0
-----
time: 2025-03-06 11:35:19.439118
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b30f954f377efb7f88c858d1dbef8bc7a4630b2cb9d530fa5022bb695ebfab9d30c74f1df4e624917eb52c54eaccd6fcaceaa5cceb886f203ffd2e
ff996193316f37bd8137ce7c3c4a605af0e0d548782dc99fb49df47f61e451dca292697f4a21ad48aed75b0433297cdf798281344371509c46f5580ef5293fc3cae51e49e90203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100dd888fe475bca816c3ded5bf52c45d4e1ad62608ef76ba98f22ba8677c74a03ba63d62949a947fa5bd4a362425101708a1ff0416c07e9b9095f
67d996a367ccd5a2cb75c890cca589a1040bcfac49d8da154af08a5ad8804489f638c9b8fb1231fb6140881ec430b6976a7758db6918e2c55cb09fe1e669d4e6cd652f143ff0d0203010001
-----
value: 1.0
-----
time: 2025-03-06 11:35:19.441132
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100dc1dc151afedb461ce2e4d8e2233f1d8b17896d4f1a5b1b0c5382d029360ea4c218b00afef3f2eda61362eaaeab6fd70ab0d0f42d092edadd9c40b
308f9a4cd7f0fc0da99f22c18f966cd4ef8c27a6e4ad407783573f9ed3a873e666f784d3897b7cddf6cec7360418031076387a9bb60432bb19e58db3c8a4efb01e1f16e1890203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b0407b8a1cc8680635a26abc8ba2109b32da4eccf7de91145dbd33066472a9167f5328edd8eebf35e1ab90445ab46165aac3742604480f7df38
66f3427cac227987878ada842b236d52c4a2291acf26e74a6028180f475d32e045324de9c204ea7b5788ee5cd6a628fc7d97984b9e882f6f2e5e5f60e5b262ecb1b2d1b2426c70203010001
-----
value: 5.0
-----
time: 2025-03-06 11:35:19.443144
-----
--------------
====================================
```

PRACTICAL 4

Implement and demonstrate the use of the following in Solidity:

(I). Variable

(II). Operators

(III). Loops

(IV). Decision Making

(V). Strings

(I). Variable

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract SolidityTest {
uint storedData;
constructor() public {
    storedData = 10;
}
function getResult() public view returns(uint){
uint a=1;
uint b=2;
uint result=a+b;
return result;

}
}
```

OUTPUT:

SOLIDITY COMPILER
COMPILER
0.8.26+commit.8a97fa7a
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile variable.sol
Compile and Run script
CONTRACT
SolidityTest (variable.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
ABI
Bytecode
Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
--> variable.sol:5:1:
Home
variable.sol
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract SolidityTest {
uint storedData;
constructor() public {
storedData = 10;
}
function getResult() public view returns(uint){
uint a=1;
uint b=2;
uint result=a+b;
return result;
}
}
To exit full screen, press and hold Esc
Listen on all transactions
Filter with transaction hash or address
web3.js
ethers.js
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Scam Alert
main
Did you know? You can learn Solidity basics and more using the Learneth plugin.
RemixAI Copilot (enabled)

SOLIDITYTEST AT 0X2E9...B28I
Balance: 0 ETH
getResult
0: uint256: 3
Low level interactions
CALLDATA
Transact

b) State Variable

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract SolidityVarTest {
    uint8 public state_var;
    constructor() {
        state_var = 16;
    }
}
```

OUTPUT:

c) Global Variable

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    // Defining a variable
    address public admin;
    // Creating a constructor to use Global variable
    constructor() {
        admin = msg.sender;
    }
}
```

Output:

(II). Operators

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract SolidityTest {
    uint16 public a = 20;
    uint16 public b = 10;
    uint public sum;
    uint public diff;
    uint public mul;
    uint public div;
    uint public mod;
    uint public dec;
    uint public inc;
    constructor() {
```

```
            sum = a + b;
            diff = a - b;
            mul = a * b;
            div = a / b;
            mod = a % b;
            dec = --b;
            inc = ++a;
        }
    }
}
```

Output:

(IV). Decision Making

// **Solidity program to demonstrate the use of 'if statement'**

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Types {
    // Declaring state variable
    uint public i = 10;
    // Defining function to demonstrate use of 'if statement'
    function decisionMaking() public view returns(bool) {
        if (i < 10) {
            return true;
        } else {
            return false;
        }
    }
}
```

Output:

SOLIDITY COMPILER
0.8.26+commit.8a97fa7a
Compile ifstatement.sol
Compile and Run script
Types (ifstatement.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Types {
// Declaring state variable
uint public i = 10;
// Defining function to demonstrate use of 'if statement'
function decisionMaking() public view returns(bool) {
if (i < 10) {
return true;
} else {
return false;
}
}
}

Deployed Contracts 1
TYPES AT 0XB54...5EEEB (MEM
Balance: 0 ETH
decisionMaking
0: bool: false
i
0: uint256: 10

```
// Solidity program to demonstrate the use of 'if...else' statement
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Types {
    // Declaring state variables
    uint public i = 10;
    bool public even;
    // Defining function to demonstrate the use of 'if...else statement'
    function decisionMaking() public {
        if (i % 2 == 0) {
            even = true;
        } else {
            even = false;
        }
    }
    // Function to return the result
    function getResult() public view returns(bool) {
        return even;
    }
}
```

OUTPUT:

(III) Strings

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    string str;
    constructor(string memory str_in) {
        str = str_in;
    }
    function strOut() public view returns (string memory) {
```

```
        return str;
    }
}
```

OUTPUT:

PRACTICAL 5

Implement and demonstrate the use of the following in Solidity :

(I). Arrays

(II). Enums

(III). Structs

(IV). Mappings

(I). Arrays

```
// Solidity program to demonstrate
// creating a fixed-size array
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Types {
    // Declaring state variables of type array
    uint[6] public data1;
    int[5] public data;
    // Defining function to add values to an array
    function arrayExample() public returns (int[5] memory, uint[6] memory) {
        data = [int(50), -63, 77, -28, 90];
        data1 = [uint(10), 20, 30, 40, 50, 60];
        return (data, data1);
    }
    // Function to return the arrays
    function getResult() public view returns (int[5] memory, uint[6] memory) {
        return (data, data1);
    }
}
```

OUTPUT:

SOLIDITY COMPILER
COMPILER
0.8.26+commit.8a97fa7a
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile Arrays.sol
Compile and Run script
CONTRACT
Types (Arrays.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
ABI
Bytecode
strings.sol
Arrays.sol
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Types {
// Declaring state variables of type array
uint[6] public data1;
int[5] public data;
// Defining function to add values to an array
function arrayExample() public returns (int[5] memory, uint[6] memory) {
data = [int(50), -63, 77, -28, 90];
data1 = [uint(10), 20, 30, 40, 50, 60];
return (data, data1);
}
// Function to return the arrays
function getResult() public view returns (int[5] memory, uint[6] memory) {
return (data, data1);
}
}
Listen on all transactions
Filter with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Test.strOut() data: 0xb36...c771e
Debug
Scam Alert
main
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
RemixAI Copilot (enabled)

Deployed Contracts 1
TYPES AT 0XB31...4E9CA (MEM
Balance: 0 ETH
arrayExample
data
uint256
data1
uint256
getResult
getResult - call
0: int256[5]: 50,-63,77,-28,90
1: uint256[6]: 10,20,30,40,50,60

(II). Enums

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    enum FreshJuiceSize { SMALL, MEDIUM, LARGE }
    FreshJuiceSize public choice;
    FreshJuiceSize public constant defaultChoice = FreshJuiceSize.MEDIUM;
    function setLarge() public {
        choice = FreshJuiceSize.LARGE;
    }
    function getChoice() public view returns (FreshJuiceSize) {
        return choice;
    }
    function getDefaultChoice() public pure returns (uint) {
        return uint(defaultChoice);
    }
}
```

OUTPUT:

(III). Structs

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    struct Book {
        string title;
        string author;
        uint book_id;
    }
    Book public book;
    function setBook() public {
        book = Book('Learn Java', 'TP', 1);
    }
    function getBookId() public view returns (uint) {
        return book.book_id;
    }
```

}

OUTPUT:

SOLIDITY COMPILER
0.8.26+commit.8a97fa7a
Compile structs.sol
Compile and Run script
Test (structs.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
struct Book {
string title;
string author;
uint book_id;
}
Book public book;
function setBook() public {
book = Book('Learn Java', 'TP', 1);
}
function getBookId() public view returns (uint) {
return book.book_id;
}
}

evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 11
Deployed Contracts 1
TEST AT 0X2F8...B2593 (MEMC
Balance: 0 ETH
setBook
book
0: string: title Learn Java
1: string: author TP
2: uint256: book_id 1
getBookId
0: uint256: 1

(IV). Mappings

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract LedgerBalance {
    mapping(address => uint) public balance;
    function updateBalance() public returns (uint) {
        balance[msg.sender] = 20;
        return balance[msg.sender];
    }
}
```

OUTPUT:

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract LedgerBalance {
    mapping(address => string) public name;
    function updateBalance() public returns (string memory) {
        name[msg.sender] = "Dip";
        return name[msg.sender];
    }
    function printSender() public view returns (address) {
        return msg.sender;
    }
}
```

Output:

SOLIDITY COMPILER
0.8.26+commit.8a97fa7a
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile Mapping2.sol
Compile and Run script
CONTRACT
LedgerBalance (Mapping2.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
ABI
Bytecode
strings.sol
Arrays.sol
Enums.sol
structs.sol
mapping.sol
Mapping2.sol
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract LedgerBalance {
mapping(address => string) public name;
function updateBalance() public returns (string memory) {
name[msg.sender] = "Dip";
return name[msg.sender];
}
function printSender() public view returns (address) {
return msg.sender;
}
}
infinite gas
401 gas
Listen on all transactions
Filter with transaction hash or address
"0": "uint256: 20"
logs
raw logs
call to LedgerBalance.balance
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: LedgerBalance.balance(address) data: 0xe3d...eddc4
Debug
Scam Alert
main
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
RemixAI Copilot (enabled)

LedgerBalance - Mapping2.sol
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 20
Deployed Contracts 1
LEDGERBALANCE AT 0X398...(
Balance: 0 ETH
updateBalance
name
address
printSender
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

PRACTICAL 6

Implement and demonstrate the use of the following in Solidity:

(I). Functions

(II). View Functions

(III). Pure Functions

(IV). Function Overloading

(V). Mathematical Functions

(VI). Cryptographic Functions

(I). Functions

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract SolidityTest {
    function testPgmResult() public pure returns (uint) {
        uint a = 1000; // local variable
        uint b = 2000; // local variable
        uint result = a + b;
        return result; // return the result
    }
}
```

OUTPUT:

(II). View Functions

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
```

```
contract Test {
    function getResult() public pure returns (uint product, uint sum) {
        uint a = 1; // local variable
        uint b = 2;
        product = a * b;
        sum = a + b;
    }
}
```

OUTPUT:

(III). Pure Functions

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract C {
    // Private state variable
    uint private data;
    // Public state variable
    uint public info;
    // Constructor
    constructor() {
        info = 10;
    }
    // Private function
```

```
    function increment(uint a) private pure returns (uint) {
        return a + 1;
    }
    // Public function
    function updateData(uint a) public {
        data = a;
    }
    function getData() public view returns (uint) {
        return data;
    }
    function compute(uint a, uint b) internal pure returns (uint) {
        return a + b;
    }
}
// Derived Contract
contract E is C {
    uint private result;
    C private c;
    // Constructor
    constructor() {
        c = new C();
    }
    function getComputedResult() public {
        result = compute(3, 5);
    }
    function getResult() public view returns (uint) {
        return result;
    }
    function getDataFromC() public view returns (uint) {
```

```
        return c.info();
    }
}
```

OUTPUT:

(IV). Function Overloading

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    // Function to get the sum of two arguments
    function getSumTwo(uint a, uint b) public pure returns (uint) {
        return a + b;
    }
    // Function to get the sum of three arguments
    function getSumThree(uint a, uint b, uint c) public pure returns (uint) {
        return a + b + c;
    }
```

```
    // Function to call sum with two arguments
    function callSumWithTwoArguments() public pure returns (uint) {
        return getSumTwo(1, 2);
    }
    // Function to call sum with three arguments
    function callSumWithThreeArguments() public pure returns (uint) {
        return getSumThree(1, 2, 3);
    }
}
```

Output:

(V). Mathematical Functions

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    function callAddMod() public pure returns (uint) {
        return addmod(4, 5, 3);
    }
```

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

OUTPUT:

(VI). Cryptographic Functions

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    // Function to compute sha256 hash
    function callSha256() public pure returns (bytes32 result) {
        return sha256(abi.encodePacked("ronaldo"));
    }
    // Function to compute keccak256 hash
    function callKeccak256() public pure returns (bytes32 result) {
        return keccak256(abi.encodePacked("ronaldo"));
    }
}
```

OUTPUT:

CONTRACT
Test - cryptofunc.sol
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 28
Deployed Contracts 1
TEST AT 0XA42...A40EA (MEM
Balance: 0 ETH
callKeccak256
0: bytes32: result 0x2ce96055cb975b629646e25b4c09bd530fd8dffa923586b679bceb0e5538eed8
callSha256
0: bytes32: result 0xe24dd2210803b4737a9bd9e3163a4ca807b63201c3bc32b68fb122ca52efff36
Low level interactions
CALLDATA
Transact

PRACTICAL 7

Implement and demonstrate the use of the following in Solidity:

(I). Contracts

(II). Inheritance

(III). Constructors

(IV). Abstract Class

(V). Interfaces

(I). Contracts

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
    // Declaring state variables
    uint public var1;
    uint public var2;
    uint public sum;
    // Defining public function that sets the value of the state variables
    function set(uint x, uint y) public {
        var1 = x;
        var2 = y;
        sum = var1 + var2;
    }
    // Defining function to return the sum of state variables
    function get() public view returns (uint) {
        return sum;
    }
}
```

OUTPUT:

SOLIDITY COMPILER
COMPILER
0.8.26+commit.8a97fa7a
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile contracts.sol
Compile and Run script
CONTRACT
Test (contracts.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
ABI
Bytecode
mathsfunc.sol
cryptofunc.sol
contracts.sol
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Test {
// Declaring state variables
uint public var1;
uint public var2;
uint public sum;
// Defining public function that sets the value of the state variables
function set(uint x, uint y) public {
var1 = x;
var2 = y;
sum = var1 + var2;
}
// Defining function to return the sum of state variables
function get() public view returns (uint) {
return sum;
}
}
Listen on all transactions
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Test.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x16a...d8db0
call to Test.callKeccak256
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Test.callKeccak256() data: 0x5b4...aa3ee
call to Test.callSha256
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Test.callSha256() data: 0x01c...74044
Debug
Scam Alert
main
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
RemixAI Copilot (enabled)

Deployed Contracts 1
TEST AT 0X091...81367 (MEMO
Balance: 0 ETH
set
x: 23
y: 36
Calldata
Parameters
transact
get
0: uint256: 59
sum
0: uint256: 59
var1
0: uint256: 23
var2
0: uint256: 36

(II). Inheritance

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract Parent {
    // Declaring internal state variable
    uint internal sum;
    // Defining public function to set value of internal state variable sum
    function setValue() public {
        uint a = 10;
        uint b = 20;
        sum = a + b;
    }
}
// Defining child contract
contract Child is Parent {
    // Defining public function to return value of internal state variable sum
    function getValue() public view returns (uint) {
        return sum;
    }
}
// Defining calling contract
contract Caller {
    // Creating child contract object
    Child cc = new Child();
    // Defining function to call setValue and getValue functions
    function testInheritance() public {
        cc.setValue();
    }
    function result() public view returns (uint) {
```

```
        return cc.getValue();
    }
}
```

OUTPUT:

(III). Constructors

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract ConstructorExample {
    // Declaring state variable
    string str;
    // Creating a constructor to set value of 'str'
    constructor() {
        str = "GeeksForGeeks";
    }
    // Defining function to return the value of 'str'
    function getValue() public view returns (string memory) {
        return str;
    }
}
```

OUTPUT:

Transactions recorded 34
Deployed Contracts 1
CONSTRUCTOREXAMPLE AT (
Balance: 0 ETH
getValue
0: string: Vinish
Low level interactions
CALLDATA
Transact

PRACTICAL 8

Implement and demonstrate the use of the following in Solidity:

(I). Libraries

(II). Assembly

(III). Events

(IV). Error Handling

```
// Solidity program to demonstrate require statement
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract RevertStatement {
    function checkOverflow(uint _num1, uint _num2) public pure returns (string memory, uint) {
        uint sum = _num1 + _num2;
        if (sum > 255) {
            revert("Overflow Exist");
        } else {
            return ("No Overflow", sum);
        }
    }
}
```

OUTPUT:

// **Solidity program to demonstrate assert statement**

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract AssertStatement {
    // Defining a state variable
    bool result;
    // Defining a function to check condition
    function checkOverflow(uint8 sum) public {
        assert(sum <= 255);  // Assertion to check if sum is within uint8 range
        result = true;
    }
    // Defining a function to print result of assert statement
```

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

OUTPUT:

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract assertStatement {
bool result;
function checkOverflow(uint8 sum) public {
    assert(sum <= 255);
    result = true;
```

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

OUTPUT:

```
// Solidity program to demonstrate revert statement
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
contract revertStatement {
    // Defining a function to check condition
    function checkOverflow(uint256 _num1, uint256 _num2)
        public
```

```
        view
        returns (string memory, uint256)
    {
        uint256 sum = _num1 + _num2;
        if (sum < 0 || sum > 255) {
            revert(" Overflow Exist");
        } else {
            return ("No Overflow", sum);
        }
    }
}
```

OUTPUT: