**PRACTICAL: 1**

**AIM:** Study Blockchain Architecture and Block structure. Perform following tasks:

1. Create Blockchain which consists of 5 blocks. [First Block is genesis block].
2. Validate the blocks by implementing mining logic [Cryptographically Mathematical Puzzles]
3. Set the difficulty of blocks.
4. Keep timestamp, block version, data and previous hash as block parameters.
5. Verify the blocks.
6. Access all transaction.
7. Modify the block data.

Implement above all functionalities using Node js/Python/Java/C++/go lang.

**CODE:**

import hashlib

import time

import json

class Block:

def \_\_init\_\_(self, index, previous\_hash, data, difficulty=3):

self.index = index

self.timestamp = time.time()

self.version = "1.0"

self.data = data

self.previous\_hash = previous\_hash

self.nonce = 0

self.difficulty = difficulty

self.hash = self.mine\_block()

def calculate\_hash(self):

block\_string = json.dumps({

"index": self.index,

"timestamp": self.timestamp,

"version": self.version,

"data": self.data,

"previous\_hash": self.previous\_hash,

"nonce": self.nonce

}, sort\_keys=True)

return hashlib.sha256(block\_string.encode()).hexdigest()

def mine\_block(self):

prefix = "0" \* self.difficulty

while True:

self.hash = self.calculate\_hash()

if self.hash[:self.difficulty] == prefix:

break

self.nonce += 1

return self.hash

class Blockchain:

def \_\_init\_\_(self, difficulty=3):

self.chain = []

self.difficulty = difficulty

self.create\_genesis\_block()

def create\_genesis\_block(self):

genesis\_block = Block(0, "0" \* 64, "Genesis Block", self.difficulty)

self.chain.append(genesis\_block)

def add\_block(self, data):

previous\_block = self.chain[-1]

new\_block = Block(len(self.chain), previous\_block.hash, data, self.difficulty)

self.chain.append(new\_block)

def is\_valid(self):

for i in range(1, len(self.chain)):

current\_block = self.chain[i]

previous\_block = self.chain[i - 1]

# Validate hash

if current\_block.hash != current\_block.calculate\_hash():

print(f"Block {current\_block.index} has been tampered with!")

return False

# Validate previous hash linkage

if current\_block.previous\_hash != previous\_block.hash:

print(f"Block {current\_block.index} previous hash is incorrect!")

return False

print("Blockchain is valid!")

return True

def display\_chain(self):

for block in self.chain:

print(json.dumps(block.\_\_dict\_\_, indent=4))

def modify\_block(self, index, new\_data):

if index == 0:

print("Cannot modify the genesis block!")

return

self.chain[index].data = new\_data # Modify block data

self.chain[index].hash = self.chain[index].calculate\_hash() # Recalculate hash

# Driver Code

if \_\_name\_\_ == "\_\_main\_\_":

my\_blockchain = Blockchain(difficulty=4) # Set difficulty level

my\_blockchain.add\_block("Alice sends 1 BTC to Bob")

my\_blockchain.add\_block("Bob sends 0.5 BTC to Charlie")

my\_blockchain.add\_block("Charlie sends 0.2 BTC to Dave")

my\_blockchain.add\_block("Dave sends 0.1 BTC to Alice")

print("\nBlockchain:")

my\_blockchain.display\_chain()

print("\nValidating Blockchain:")

my\_blockchain.is\_valid()

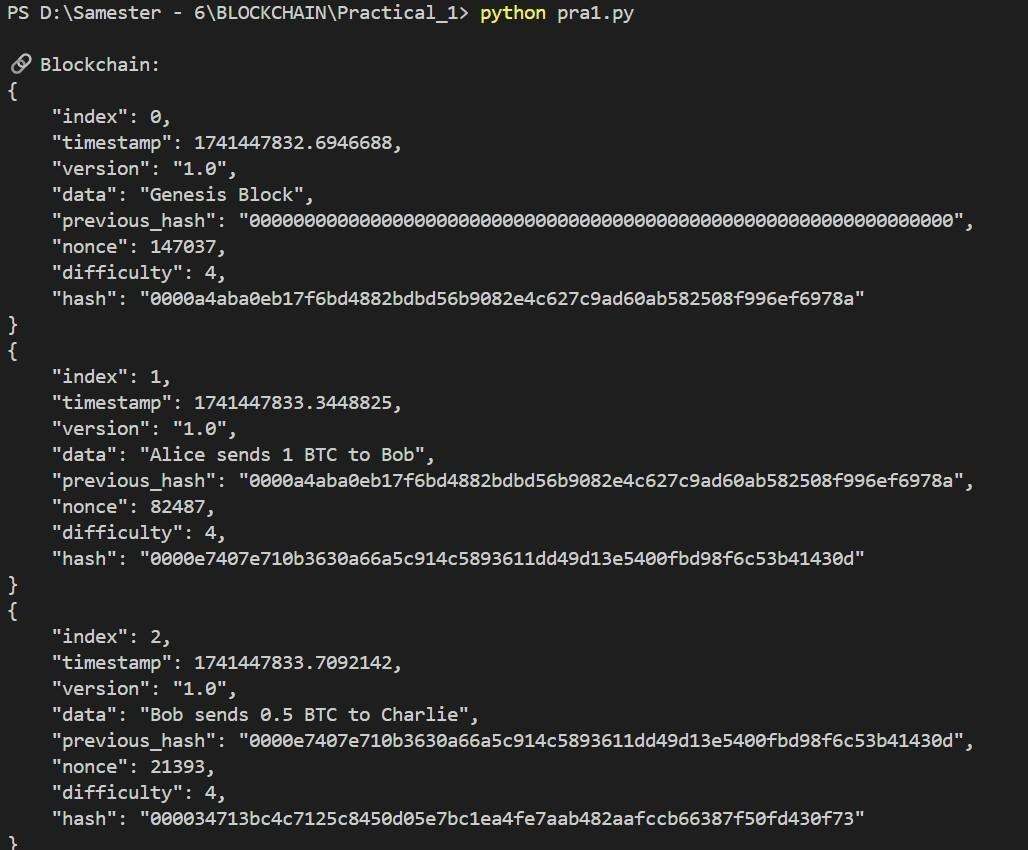
print("\nModifying Block 2...")

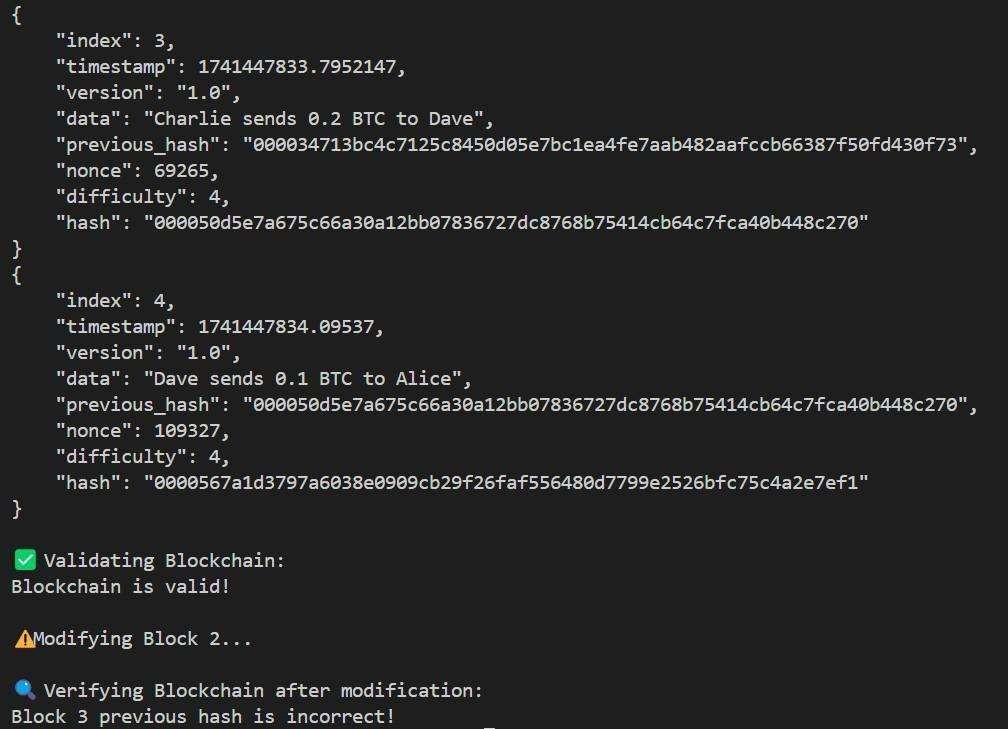
my\_blockchain.modify\_block(2, "Bob sends 10 BTC to Charlie")

print("\nVerifying Blockchain after modification:")

my\_blockchain.is\_valid()

**OUTPUT:**





**LEARNING OUTCOME:**

Understanding blockchain architecture is key to developing decentralized applications. It involves learning the block structure, which includes important components such as block header, timestamp, version, previous hash, and data. By creating a basic blockchain with multiple blocks, including the genesis block, developers implement mining logic to validate blocks using cryptographic mathematical puzzles. Difficulty levels are set to control block creation times, and the blockchain's integrity is ensured through previous hash linking. The timestamp ensures accurate records, while block versioning helps in handling updates and changes over time. Verifying blockchain transactions and modifying block data helps in understanding how to maintain the security and immutability of the blockchain. Transaction management, including adding, verifying, and removing transactions, is explored to ensure a solid understanding of blockchain operations.