Blockchain Lab

Aim - Create a blockchain using python.

Theory -

1. **What is blockchain**

Blockchain is a revolutionary technology that functions as a shared, immutable digital ledger. The name "blockchain" comes from its structure data is organized in blocks, with each new block linked to the one before it, forming a continuous chain.

Each block contains crucial data, such as a list of transactions, a timestamp, and a unique identifier called a cryptographic hash. This hash is generated from the block's contents and the hash of the previous block, ensuring that each block is tightly connected to the one before it.

- Blockchain's linked structure makes data tampering detectable by altering hashes and breaking the chain.
- It acts as a distributed database, storing transactions across the network.
- Each transaction is verified by the majority, ensuring legitimacy.
- This decentralization prevents any single party from manipulating the data.

Blockchain is decentralized and distributed, meaning no single authority controls it. Instead, multiple computers (nodes) on a network each have a copy of the blockchain, keeping the ledger synchronized. This setup ensures that once data, like a transaction, is recorded and confirmed, it becomes immutable almost impossible to alter or delete.

2. **What is block in blockchain**

A block in blockchain is a digital container that securely stores verified transaction data, forming a fundamental part of the linked chain; each block holds transaction details, a timestamp, its own unique cryptographic hash, and the hash of the previous block, creating an immutable, chronological record that's distributed across a network for security and transparency.

3. **What are the components of blockchain**

Key Components of a Block:

- **Transaction Data**: Records details like who, what, when, and how much was exchanged (e.g., cryptocurrency, assets).
- **Timestamp**: Marks the exact time the block was created, ensuring chronological order.
- **Hash**: A unique digital fingerprint (like a digital seal) generated from the block's data, making it tamper-proof.
- **Previous Hash:** A reference to the hash of the block that came before it, linking them together into a chain.

4. What is the process of mining

Mining is a fundamental process in blockchain technology used to validate transactions and add new blocks to the blockchain. It ensures security, transparency, and decentralization of the network. In cryptocurrencies like Bitcoin, mining follows the Proof of Work (PoW) mechanism. Mining is the process in which miners solve complex cryptographic puzzles to verify transactions and append them to the blockchain, in return for cryptocurrency rewards.

1. **Steps Involved in the Mining Process**

1.1 **Transaction Initiation**

- Users initiate transactions on the blockchain network.
- These transactions are broadcast to all network nodes.

1.2 **Transaction Pool (Mempool)**

- Unconfirmed transactions are stored in a temporary area called the mempool.
- Miners select transactions from this pool.

1.3 **Block Creation**

A new block is formed containing:

- Selected transactions
- Hash of the previous block
- Timestamp
- Nonce (random number)

1.4 **Hash Generation**

- The miner applies a cryptographic hash function (e.g., SHA-256).
- The output hash must satisfy the network's difficulty condition (leading zeros).

1.5 **Proof of Work**

- Miners repeatedly change the nonce and compute hashes.
- The first miner to find a valid hash provides Proof of Work.

1.6 **Block Verification**

- The mined block is broadcast to the network.
- Other nodes verify:
    - Hash validity
    - Transaction authenticity
    - Block structure

1.7 **Block Addition to Blockchain**

- Once verified, the block is permanently added to the blockchain.
- The chain remains immutable and synchronized.

1.8 **Reward Distribution**

- The successful miner receives:
    - Block reward (new coins)
    - Transaction fees

5. How to check validity of blocks in blockchain

Block validity checking is the process of verifying the correctness of a block's structure, transactions, hash value, and consensus rules before accepting it into the blockchain.

- **Block Structure Verification**
  The block is checked to ensure it contains all required components such as the block header, previous block hash, timestamp, Merkle root, nonce, and the list of transactions.

- **Previous Block Hash Validation**
  The previous hash stored in the block is compared with the hash of the latest block in the blockchain. This step ensures proper linkage and continuity of the blockchain.

- **Hash and Proof of Work Validation**
  The hash of the block is recalculated and verified. The hash must satisfy the difficulty level defined by the network, such as a required number of leading zeros, confirming valid Proof of Work.

- **Transaction Validation**
  Each transaction in the block is verified for correct digital signatures, sufficient balance, proper format, and absence of double spending.

- **Merkle Root Verification**
  The Merkle root is recalculated from all transactions in the block and compared with the Merkle root stored in the block header to ensure transaction integrity.

- **Timestamp Validation**
  The block timestamp is checked to ensure it is not too far in the future and is greater than the timestamp of the previous block.

- **Block Size and Rule Validation**
  The block size and transaction limits are verified to ensure compliance with blockchain protocol rules.

- **Consensus Rule Verification**
  The block is validated according to the network's consensus mechanism, such as Proof of Work or Proof of Stake, ensuring it follows all consensus rules.

Output -

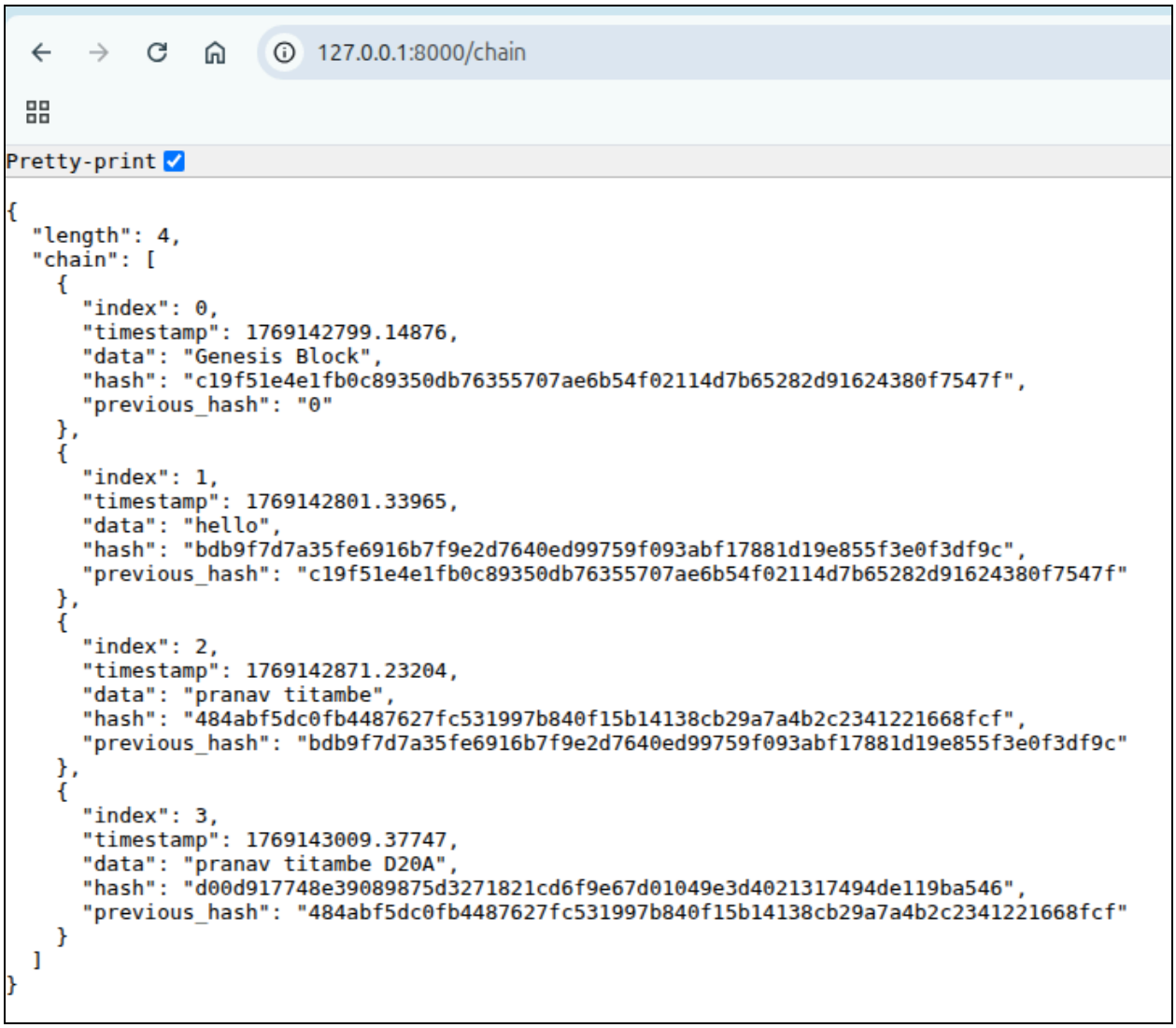
```
from fastapi import FastAPI
from hashlib import sha256
from time import time
from typing import List

app = FastAPI(title="Simple Blockchain with FastAPI")
# ---------------- BLOCK STRUCTURE ----------------
class Block:
    def __init__(self, index, timestamp, data, previous_hash):
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.calculate_hash()

    def calculate_hash(self):
                                                        block_string        =
f"{self.index}{self.timestamp}{self.data}{self.previous_hash}"
        return sha256(block_string.encode()).hexdigest()


# ---------------- BLOCKCHAIN ----------------
class Blockchain:
    def __init__(self):
        self.chain: List[Block] = []
        self.create_genesis_block()

    def create_genesis_block(self):
        genesis_block = Block(0, time(), "Genesis Block", "0")
        self.chain.append(genesis_block)

    def get_last_block(self):
        return self.chain[-1]

    def add_block(self, data):
```

```
        last_block = self.get_last_block()
        new_block = Block(
            index=last_block.index + 1,
            timestamp=time(),
            data=data,
            previous_hash=last_block.hash
        )
        self.chain.append(new_block)
        return new_block

    def is_chain_valid(self):
        for i in range(1, len(self.chain)):
            current = self.chain[i]
            previous = self.chain[i - 1]

            # Check hash integrity
            if current.hash != current.calculate_hash():
                return False

            # Check previous hash link
            if current.previous_hash != previous.hash:
                return False

        return True


# ---------------- INIT BLOCKCHAIN ----------------
blockchain = Blockchain()

# ---------------- ROUTES ----------------

@app.get("/")
def home():
    return {"message": "Blockchain API is running 🚀"}


@app.get("/chain")
def get_chain():
    chain_data = []
    for block in blockchain.chain:
```

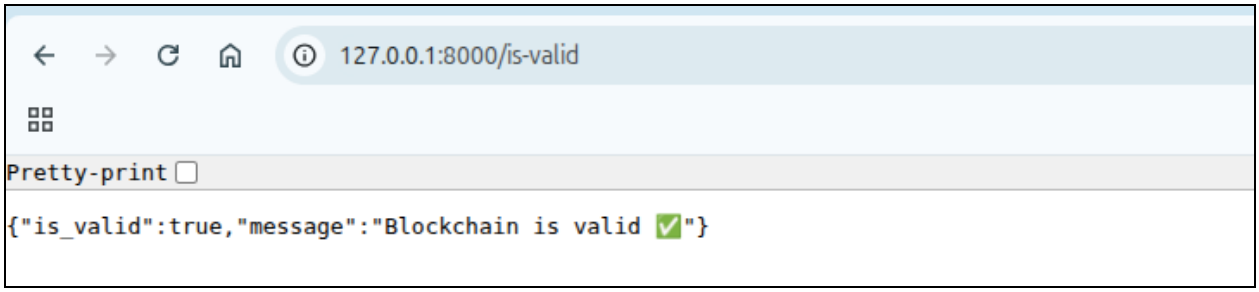
```
        chain_data.append({
            "index": block.index,
            "timestamp": block.timestamp,
            "data": block.data,
            "hash": block.hash,
            "previous_hash": block.previous_hash
        })

    return {
        "length": len(chain_data),
        "chain": chain_data
    }
@app.post("/block")
def create_block(data: str):
    block = blockchain.add_block(data)
    return {
        "message": "New block added ✅",
        "block": {
            "index": block.index,
            "timestamp": block.timestamp,
            "data": block.data,
            "hash": block.hash,
            "previous_hash": block.previous_hash
        }
    }
@app.get("/is-valid")
def check_validity():
    is_valid = blockchain.is_chain_valid()
    return {
        "is_valid": is_valid,
        "message": "Blockchain is valid ✅" if is_valid else "Blockchain is
NOT valid ❌"
    }
```

Pranav Titambe, D20A, 60


```
{
  "message": "New block added ✅",
  "block": {
    "index": 3,
    "timestamp": 1769143009.37747,
    "data": "pranav titambe D20A",
    "hash": "d00d917748e39089875d3271821cd6f9e67d01049e3d4021317494de119ba546",
    "previous_hash": "484abf5dc0fb4487627fc531997b840f15b14138cb29a7a4b2c2341221668fcf"
  }
}
```


```
{
  "length": 4,
  "chain": [
    {
      "index": 0,
      "timestamp": 1769142799.14876,
      "data": "Genesis Block",
      "hash": "c19f51e4e1fb0c89350db76355707ae6b54f02114d7b65282d91624380f7547f",
      "previous_hash": "0"
    },
    {
      "index": 1,
      "timestamp": 1769142801.33965,
      "data": "hello",
      "hash": "bdb9f7d7a35fe6916b7f9e2d7640ed99759f093abf17881d19e855f3e0f3df9c",
      "previous_hash": "c19f51e4e1fb0c89350db76355707ae6b54f02114d7b65282d91624380f7547f"
    },
    {
      "index": 2,
      "timestamp": 1769142871.23204,
      "data": "pranav titambe",
      "hash": "484abf5dc0fb4487627fc531997b840f15b14138cb29a7a4b2c2341221668fcf",
      "previous_hash": "bdb9f7d7a35fe6916b7f9e2d7640ed99759f093abf17881d19e855f3e0f3df9c"
    },
    {
      "index": 3,
      "timestamp": 1769143009.37747,
      "data": "pranav titambe D20A",
      "hash": "d00d917748e39089875d3271821cd6f9e67d01049e3d4021317494de119ba546",
      "previous_hash": "484abf5dc0fb4487627fc531997b840f15b14138cb29a7a4b2c2341221668fcf"
    }
  ]
}
```



Conclusion - The concept of mining and validating the blockchain was understood via this experiment.