Practical no: 14

Name: Bhairavi Narendra Rewatkar
Roll No.: DMET1221006
Subject: Blockchain Technology Laboratory

Title: Time stamping and block validation.
Aim: Write a program to modify the block structure to include timestamps and to implement a function that validates the chronological order of blocks based on their timestamps.

Source Code:

```
import java.security.MessageDigest;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
class Block {
   public int index;
   public String previousHash;
   public String hash;
   public String data;
   public long timestamp;
   private int nonce;
   public Block(int index, String previousHash, String data, long timestamp) {
      this.index = index;
      this.previousHash = previousHash;
      this.data = data;
      this.timestamp = timestamp;
      this.hash = calculateHash();
   }
   public String calculateHash() {
      String input = index + previousHash + data + timestamp + nonce;
      return applySHA256(input);
   }
   public static String applySHA256(String input) {
      try {
         MessageDigest digest = MessageDigest.getInstance("SHA-256");
         byte[] hashBytes = digest.digest(input.getBytes("UTF-8"));
         StringBuilder hexString = new StringBuilder();
         for (byte hashByte : hashBytes) {
            String hex = Integer.toHexString(0xff & hashByte);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
         }
         return hexString.toString();
      } catch (Exception e) {
         throw new RuntimeException(e);
      }
   }
   public void mineBlock(int difficulty) {
      String target = new String(new char[difficulty]).replace('\0', '0');
      while (!hash.startsWith(target)) {
```

```
            nonce++;
            hash = calculateHash();
        }
        System.out.println("Block mined: " + hash);
    }
}
class Blockchain {
    public List<Block> chain;
    private int difficulty;
    public Blockchain(int difficulty) {
        this.chain = new ArrayList<>();
        this.difficulty = difficulty;
        chain.add(createGenesisBlock());
    }
    private Block createGenesisBlock() {
        return new Block(0, "0", "Genesis Block", currentTimestamp());
    }
    public Block getLatestBlock() {
        return chain.get(chain.size() - 1);
    }
    public void addBlock(String data) {
        Block latestBlock = getLatestBlock();
        Block newBlock = new Block(latestBlock.index + 1, latestBlock.hash, data,
currentTimestamp());
        newBlock.mineBlock(difficulty);
        chain.add(newBlock);
    }
    public long currentTimestamp() {
        return new Date().getTime();
    }
    public boolean validateBlockchain() {
        for (int i = 1; i < chain.size(); i++) {
            Block currentBlock = chain.get(i);
            Block previousBlock = chain.get(i - 1);
            if (!currentBlock.hash.equals(currentBlock.calculateHash())) {
                System.out.println("Block " + i + " has been tampered with.");
                return false;
            }
            if (!currentBlock.previousHash.equals(previousBlock.hash)) {
                System.out.println("Block " + i + "'s previous hash doesn't match.");
                return false;
            }
            if (currentBlock.timestamp <= previousBlock.timestamp) {
                System.out.println("Block " + i + " has an invalid timestamp.");
                return false;
            }
        }
        return true;
    }
}
public class Main {
```

```
    public static void main(String[] args) {
        Blockchain blockchain = new Blockchain(4);
        blockchain.addBlock("Block 1 Data");
        blockchain.addBlock("Block 2 Data");
        blockchain.addBlock("Block 3 Data");
        System.out.println("Blockchain is valid: " + blockchain.validateBlockchain());
        for (Block block : blockchain.chain) {
            System.out.println("Block " + block.index + " [Hash: " + block.hash + ", Previous Hash: " +
block.previousHash + ", Timestamp: " + block.timestamp + "]");
        }
    }
}
```

Output: