AndrewID: muyut

Name: Muyu Tong

Task 0 Execution

```
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
0
Current size of chain: 1
Difficulty of most recent block: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 2234
Expected total hashes required for the whole chain: 256.000000
Nonce for most recent block: 500
Chain hash: 00C5FEA351D4678902F296E17675AA9D355C845F1F07A7DA0E4087728E70919C
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
2
Enter transaction
Alice pays Bob 100 DScoin
Total execution time to add this block was 28 milliseconds
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
2
Enter transaction
Bob pays Carol 50 DScoin
Total execution time to add this block was 22 milliseconds
0. View basic blockchain status.
```

```
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
2
Enter transaction
Carol pays Andy 10 DScoin
Total execution time to add this block was 26 milliseconds
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
2
Total execution time to add this block was 1 milliseconds
Chain verification: TRUE
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
3
View the block chain
BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:29:34.895, data: '', difficulty: 2, previousHash: '', nonce: 500}
, {index: 1, timestamp: 2022-03-19 16:29:45.629, data: 'Alice pays Bob 100 DScoin', difficulty: 2, previousHash: '00C5FEA351D4678902F296E17675AA9D355C845F1F07A7DA0E4087728E70919C', nonce: 19}
, {index: 2, timestamp: 2022-03-19 16:30:23.917, data: 'Bob pays Carol 50 DScoin', difficulty: 2, previousHash: '00746F2541D68122AD760E6917A85DBB5B4BB7B3B12E49107A94E93AF6563B24', nonce: 116}
, {index: 3, timestamp: 2022-03-19 16:30:45.4, data: 'Carol pays Andy 10 DScoin', difficulty: 2, previousHash: '00F3CED4D03CD9B66B6E39C04A8DBB5E8B24807DC03F492897101A84EC293728', nonce: 85}
], chainHash: '006427688CFDE049B2C1AFAB910284BD99BA83547D8BD0329ED4A1A144BB9551'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
4
```

```
Corrupt the Blockchain
Enter block ID of block to corrupt
1
Enter new data for block 1
Alice pays Bob 76 DScoin
Block 1 now holds Alice pays Bob 76 DScoin
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
3
View the block chain
BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:29:34.895, data: '', difficulty: 2, previousHash: '', nonce: 500}
, {index: 1, timestamp: 2022-03-19 16:29:45.629, data: 'Alice pays Bob 76 DScoin', difficulty: 2, previousHash: '00C5FEA351D4678902F296E17675AA9D355C845F1F07A7DA0E4087728E70919C', nonce: 19}
, {index: 2, timestamp: 2022-03-19 16:30:23.917, data: 'Bob pays Carol 50 DScoin', difficulty: 2, previousHash: '00746F2541D68122AD760E6917A85DBB5B4BB7B3B12E49107A94E93AF6563B24', nonce: 116}
, {index: 3, timestamp: 2022-03-19 16:30:45.4, data: 'Carol pays Andy 10 DScoin', difficulty: 2, previousHash: '00F3CED4D03CD9B66B6E39C04A8DBB5E8B24807DC03F492897101A84EC293728', nonce: 85}
], chainHash: '006427688CFDE049B2C1AFAB910284BD99BA83547D8BD0329ED4A1A144BB9551'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
2
Improper hash on node 1 Does not begin with 00
Total execution time to verify the chain was 7 milliseconds
Improper hash on node 2 Previous Hash is incorrect
Total execution time to verify the chain was 8 milliseconds
Chain verification: FALSE
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
5
Total execution time required to repair the chain was 5 milliseconds
0. View basic blockchain status.
1. Add a transaction to the blockchain.
```

```
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
2
Total execution time to add this block was 9 milliseconds
Chain verification: TRUE
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
4
Enter transaction
Andy pays Sean 25 DScoin
Total execution time to add this block was 187 milliseconds
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
0
Current size of chain: 5
Difficulty of most recent block: 4
Total difficulty for all blocks: 12
Approximate hashes per second on this machine: 2234
Expected total hashes required for the whole chain: 66560.000000
Nonce for most recent block: 31761
Chain hash: 00008E64732C7E00F814F2A0EB9ED25CE3CA5439699F69BD17448F8C60B38807
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
6

Process finished with exit code 0
```

Task 0 Block.java

```
import java.math.BigInteger;
```

```
import java.sql.Timestamp;

public class Block {
    int index;
    Timestamp timestamp;
    String data;
    int difficulty;
    String previousHash;
    BigInteger nonce;

    public Block(int index, Timestamp timestamp, String data, int difficulty) {
        this.index = index;
        this.timestamp = timestamp;
        this.data = data;
        this.difficulty = difficulty;
        this.previousHash = "";
        this.nonce = new BigInteger("0");
    }

    public String calculateHash() {
        String hash = "";
        hash =
Utils.getHashString(index+timestamp.toString()+data+previousHash+nonce+difficulty);
        return hash;
    }

    public BigInteger getNonce() {
        return nonce;
    }

    public String proofOfWork() {
        String hash = calculateHash();
        while (!hash.matches("0".repeat(difficulty)+".*")) {
            nonce = nonce.add(new BigInteger("1"));
            hash = calculateHash();
        }
        return hash;
    }

    public int getDifficulty() {
        return difficulty;
    }

    public void setDifficulty(int difficulty) {
        this.difficulty = difficulty;
    }

    @Override
    public String toString() {
        return "{" +
                "index: " + index +
```

```
                ", timestamp: " + timestamp +
                ", data: '" + data + '\'' +
                ", difficulty: " + difficulty +
                ", previousHash: '" + previousHash + '\'' +
                ", nonce: " + nonce +
                "}\n";
    }

    public String getPreviousHash() {
        return previousHash;
    }

    public void setPreviousHash(String previousHash) {
        this.previousHash = previousHash;
    }

    public int getIndex() {
        return index;
    }

    public void setIndex(int index) {
        this.index = index;
    }

    public Timestamp getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(Timestamp timestamp) {
        this.timestamp = timestamp;
    }

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }
}

```

Task 0 BlockChain.java

```
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Scanner;

public class BlockChain {
    List<Block> chain;
    String chainHash;
    int hashPerSec;

    public BlockChain() {
        chain = new ArrayList<>();
        chainHash = "";
    }

    public void addBlock(Block newBlock) {
        String testHash = newBlock.calculateHash();
        if (newBlock.previousHash.equals(chainHash) &&
testHash.matches("0".repeat(newBlock.difficulty)+".*")) {
            chain.add(newBlock);
            chainHash = testHash;
        } else {
            System.out.println("Illegal block!");
        }
    }

    public String getChainHash() {
        return chainHash;
    }

    public Timestamp getTime() {
        return new Timestamp(System.currentTimeMillis());
    }

    public Block getLatestBlock() {
        return chain.get(chain.size()-1);
    }

    public int getChainSize() {
        return chain.size();
    }

    public void computeHashesPerSecond() {
        String s = "00000000";
        byte[] bytes = s.getBytes(StandardCharsets.UTF_8);
        byte[] hs;
        long start = System.currentTimeMillis();
        MessageDigest md = null;
        try {
            md = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        for (int i = 0; i < 2000000; i++) {
```

```
            hs = md.digest(bytes);
        }
        long end = System.currentTimeMillis();
        hashPerSec = (int) (2000000/(end-start));
    }

    public int getHashesPerSecond() {
        return hashPerSec;
    }

    @Override
    public String toString() {
        return "BlockChain: {" +
                "chain: " + chain +
                ", chainHash: '" + chainHash + '\'' +
                '}';
    }

    public Block getBlock(int i) {
        return chain.get(i);
    }

    public int getTotalDifficulty() {
        int total = 0;
        for (Block block: chain) {
            total += block.getDifficulty();
        }
        return total;
    }

    public double getTotalExpectedHashes() {
        double total = 0;
        for (Block block: chain) {
            total += Math.pow(16, block.getDifficulty());
        }

        return total;
    }

    public String isChainValid() {
        long start = System.currentTimeMillis();
        long end;
        String block1Hash = ""; // previous block hash
        String block2Hash = ""; // current block hash
        for (Block block : chain) { // each round test if current block hash match previous one
            block2Hash = block.calculateHash();
            if (!block2Hash.matches("0".repeat(block.getDifficulty()) + ".*")){
                System.out.printf("Improper hash on node %d Does not begin with " + "0".repeat(block.getDifficulty()) + "\n", block.getIndex());
                end = System.currentTimeMillis();
```

```
                System.out.printf("Total execution time to verify the chain was %d milliseconds\n", end-start);
            } else if (!block.getPreviousHash().equals(block1Hash)) {
                System.out.printf("Improper hash on node %d Previous Hash is incorrect\n", block.getIndex());
                end = System.currentTimeMillis();
                System.out.printf("Total execution time to verify the chain was %d milliseconds\n", end-start);
                return "FALSE";
            }
            block1Hash = block2Hash;
        }

        //check if the last block hash equals chain hash
        if (!block1Hash.equals(chainHash)) {
            System.out.println("The chain hash is incorrect");
            return "FALSE";
        }
        end = System.currentTimeMillis();
        System.out.printf("Total execution time to add this block was %d milliseconds\n", end-start);
        return "TRUE";
    }

    public void repairChain() {
        long start = System.currentTimeMillis();
        String block1Hash = ""; // previous block hash
        String block2Hash = ""; // current block hash
        boolean repair = false;
        int i;
        for (i = 0;  i < chain.size(); i++) { // each round test if current block hash match previous one
            Block block = chain.get(i);
            block2Hash = block.calculateHash();

            if (!block2Hash.matches("0".repeat(block.getDifficulty()) + ".*")
                    || !block.getPreviousHash().equals(block1Hash)) {
                break;
            }
            block1Hash = block2Hash; //set previous hash
        }
        if (i == chain.size() && !block1Hash.equals(chainHash)) i = chain.size()-1; // the last block hash not equals to chain hash
        else if (i == chain.size()) return; // the chain is correct

        for (; i < chain.size(); i++ ) {//repair the chain from the corrupt block
            Block block = chain.get(i);
            block.setPreviousHash(i==0?"":chain.get(i-1).calculateHash()); // previous hash is empty for the first block
            block.proofOfWork();
            chain.set(i, block);
```

```
150         }
151 
152         chainHash = chain.get(chain.size()-1).calculateHash(); // update the chain hash in
    the end
153         long end = System.currentTimeMillis();
154         System.out.printf("Total execution time required to repair the chain was %d
    milliseconds\n", end-start);
155     }
156 
157     public static void main(String[] args) {
158         BlockChain chain = new BlockChain();
159         Block gen = new Block(0, new Timestamp(System.currentTimeMillis()),"", 2);
160         gen.proofOfWork();
161         chain.addBlock(gen);
162         chain.computeHashesPerSecond();
163 
164         Scanner scanner = new Scanner(System.in);
165         int option;
166         do {
167             System.out.println("0. View basic blockchain status.\n" +
168                     "1. Add a transaction to the blockchain.\n" +
169                     "2. Verify the blockchain.\n" +
170                     "3. View the blockchain.\n" +
171                     "4. corrupt the chain.\n" +
172                     "5. Hide the corruption by repairing the chain.\n" +
173                     "6. Exit");
174             option = Integer.parseInt(scanner.nextLine());
175             switch (option) {
176                 case 0:
177                     System.out.printf("Current size of chain: %d\n", chain.getChainSize());
178                     System.out.printf("Difficulty of most recent block: %d\n",
    chain.getLatestBlock().getDifficulty());
179                     System.out.printf("Total difficulty for all blocks: %d\n",
    chain.getTotalDifficulty());
180                     System.out.printf("Approximate hashes per second on this machine: %d\n",
    chain.getHashesPerSecond());
181                     System.out.printf("Expected total hashes required for the whole chain:
    %f\n", chain.getTotalExpectedHashes());
182                     System.out.printf("Nonce for most recent block: %s\n",
    chain.getLatestBlock().getNonce());
183                     System.out.printf("Chain hash: %s\n", chain.getChainHash());
184                     break;
185                 case 1:
186                     System.out.println("Enter difficulty > 0");
187                     int diff = Integer.parseInt(scanner.nextLine());
188                     System.out.println("Enter transaction");
189                     String transaction = scanner.nextLine();
190                     long start = System.currentTimeMillis();
191                     Block newBlock = new Block(chain.getChainSize(), new
    Timestamp(System.currentTimeMillis()), transaction, diff);
192                     newBlock.setPreviousHash(chain.chainHash);
```

```
                    newBlock.proofOfWork();
                    chain.addBlock(newBlock);
                    long end = System.currentTimeMillis();
                    System.out.printf("Total execution time to add this block was %d milliseconds\n", end-start);
                    break;
                case 2:
                    String res = chain.isChainValid();
                    System.out.println("Chain verification: " + res);
                    break;
                case 3:
                    System.out.println("View the block chain");
                    System.out.println(chain);
                    break;
                case 4:
                    System.out.println("Corrupt the Blockchain");
                    System.out.println("Enter block ID of block to corrupt");
                    int id = Integer.parseInt(scanner.nextLine());
                    System.out.printf("Enter new data for block %d\n", id);
                    String d = scanner.nextLine();
                    chain.getBlock(id).setData(d);
                    System.out.printf("Block %d now holds %s\n", id, d);
                    break;
                case 5:
                    chain.repairChain();
                    break;
                default:
                    break;
            }
        } while (option != 6);
    }
}

```

Task 1 Client Side Execution

```
Client running
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
0
{selection:0, size:1,
chainHash:'000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F',
totalHashes:256.0, totalDiff:2, recentNonce:61, hps:2188,
response:''}
0. View basic blockchain status.
```

```
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
2
Enter transaction
Alice pays Bob 100 DScoin
{selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Total execution time to add this block was 32 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
2
Enter transaction
Bob pays Carol 50 DScoin
{selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Total execution time to add this block was 21 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
2
Enter transaction
Carol pays Andy 10DScoin
{selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Total execution time to add this block was 29 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
```

```
2
{selection:2, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Chain verification: TRUE
Total execution time to verify the chain was 3 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
3
{selection:3, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: '', difficulty: 2, previousHash: '', nonce: 61}
, {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 100 DScoin', difficulty: 2, previousHash: '000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce: 104}
, {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin', difficulty: 2, previousHash: '00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce: 135}
, {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty: 2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce: 234}
], chainHash: '00614422F0012102AACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
4
Corrupt the Blockchain
Enter block ID of block to corrupt
1
Enter new data for block 1
Alice pays Bob 76 DScoin
{selection:4, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Block 1 now holds Alice pays Bob 76 DScoin'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
3
```

```
{selection:3, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: '', difficulty: 2, previousHash: '', nonce: 61}
, {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 76 DScoin', difficulty: 2, previousHash: '000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce: 104}
, {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin', difficulty: 2, previousHash: '00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce: 135}
, {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty: 2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce: 234}
], chainHash: '00614422F0012102AACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
2
{selection:2, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Chain verification: FALSE
Total execution time to verify the chain was 11 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
5
{selection:5, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Total execution time required to repair the chain was 45 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
2
{selection:2, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Chain verification: TRUE
Total execution time to verify the chain was 9 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
```

```
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
1
Enter difficulty > 0
4
Enter transaction
Andy pays Sean 25 DSCoin
{selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:'Total execution time to add this block was 192 milliseconds'}
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
6
{selection:6, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null, hps:0,
response:''}

Process finished with exit code 0

```

Task 1 Server Side Execution

```
Blockchain server running
We have a visitor
Setting response to
{"selection":0,"size":1,"chainHash":"000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F","totalHashes":256.0,"totalDiff":2,"recentNonce":61,"hps":2188,"response":""}
Adding a block
Setting response to Total execution time to add this block was 32 milliseconds
{"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution time to add this block was 32 milliseconds"}
Adding a block
Setting response to Total execution time to add this block was 21 milliseconds
{"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution time to add this block was 21 milliseconds"}
Adding a block
Setting response to Total execution time to add this block was 29 milliseconds
{"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution time to add this block was 29 milliseconds"}
Chain verification: TRUE
Setting response to Chain verification: TRUE
Total execution time to verify the chain was 3 milliseconds

```

{"selection":2,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Chain verification: TRUE\nTotal execution time to verify the chain was 3 milliseconds"}
View the Blockchain
Setting response to BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: '', difficulty: 2, previousHash: '', nonce: 61}
, {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 100 DScoin', difficulty: 2, previousHash: '000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce: 104}
, {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin', difficulty: 2, previousHash: '00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce: 135}
, {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty: 2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce: 234}
], chainHash: '00614422F0012102AACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}
{"selection":3,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: \u0027\u0027, difficulty: 2, previousHash: \u0027\u0027, nonce: 61}\n, {index: 1, timestamp: 2022-03-19 16:53:47.658, data: \u0027Alice pays Bob 100 DScoin\u0027, difficulty: 2, previousHash: \u0027000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F\u0027, nonce: 104}\n, {index: 2, timestamp: 2022-03-19 16:53:56.887, data: \u0027Bob pays Carol 50 DScoin\u0027, difficulty: 2, previousHash: \u002700072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27\u0027, nonce: 135}\n, {index: 3, timestamp: 2022-03-19 16:54:07.05, data: \u0027Carol pays Andy 10DScoin\u0027, difficulty: 2, previousHash: \u002700ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90\u0027, nonce: 234}\n], chainHash: \u002700614422F0012102AACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C\u0027}"}
Corrupt the Blockchain
Setting response to Block 1 now holds Alice pays Bob 76 DScoin
{"selection":4,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Block 1 now holds Alice pays Bob 76 DScoin"}
View the Blockchain
Setting response to BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: '', difficulty: 2, previousHash: '', nonce: 61}
, {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 76 DScoin', difficulty: 2, previousHash: '000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce: 104}
, {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin', difficulty: 2, previousHash: '00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce: 135}
, {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty: 2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce: 234}
], chainHash: '00614422F0012102AACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}

```
{"selection":3,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"BlockChain:
{chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: \u0027\u0027, difficulty: 2,
previousHash: \u0027\u0027, nonce: 61}\n, {index: 1, timestamp: 2022-03-19 16:53:47.658, data:
\u0027Alice pays Bob 76 DScoin\u0027, difficulty: 2, previousHash:
\u0027000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F\u0027, nonce: 104}\n,
{index: 2, timestamp: 2022-03-19 16:53:56.887, data: \u0027Bob pays Carol 50 DScoin\u0027,
difficulty: 2, previousHash:
\u002700072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27\u0027, nonce: 135}\n,
{index: 3, timestamp: 2022-03-19 16:54:07.05, data: \u0027Carol pays Andy 10DScoin\u0027,
difficulty: 2, previousHash:
\u002700ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90\u0027, nonce: 234}\n],
chainHash: \u002700614422F0012102AACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C\u0027}"}
Improper hash on node 1 Does not begin with 00
Improper hash on node 2 Previous Hash is incorrect
Chain verification: FALSE
Setting response to Chain verification: FALSE
Total execution time to verify the chain was 11 milliseconds
{"selection":2,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Chain
verification: FALSE\nTotal execution time to verify the chain was 11 milliseconds"}
Repairing the entire chain
Setting response to Total execution time required to repair the chain was 45 milliseconds
{"selection":5,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution
time required to repair the chain was 45 milliseconds"}
Chain verification: TRUE
Setting response to Chain verification: TRUE
Total execution time to verify the chain was 9 milliseconds
{"selection":2,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Chain
verification: TRUE\nTotal execution time to verify the chain was 9 milliseconds"}
Adding a block
Setting response to Total execution time to add this block was 192 milliseconds
{"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution
time to add this block was 192 milliseconds"}
Setting response to
{"selection":6,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":""}
```

Task 1 Client Source Code

```java
import com.google.gson.Gson;

import java.io.*;
import java.net.Socket;
import java.util.Scanner;

//https://github.com/CMU-Heinz-95702/lab4-http-server/tree/master/Example-Socket-Code
public class BlockchainClient {
    public void init() throws IOException {
        System.out.println("Client running");
        Gson gson = new Gson();
        Socket socket = new Socket("localhost", 7777);
```

```
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())));
        Scanner scanner = new Scanner(System.in);
        int option;
        do {
            RequestMessage req = new RequestMessage();
            System.out.println("0. View basic blockchain status.\n" +
                    "1. Add a transaction to the blockchain.\n" +
                    "2. Verify the blockchain.\n" +
                    "3. View the blockchain.\n" +
                    "4. corrupt the chain.\n" +
                    "5. Hide the corruption by repairing the chain.\n" +
                    "6. Exit");
            option = Integer.parseInt(scanner.nextLine());
            switch (option) {
                case 0:
                    req.setSelection(0);
                    break;
                case 1:
                    System.out.println("Enter difficulty > 0");
                    int diff = Integer.parseInt(scanner.nextLine());
                    System.out.println("Enter transaction");
                    String transaction = scanner.nextLine();
                    req = new RequestMessage();
                    req.setSelection(1);
                    req.setDifficulty(diff);
                    req.setMessage(transaction);
                    break;
                case 2:
                    req.setSelection(2);
                    break;
                case 3:
                    req.setSelection(3);
                    break;
                case 4:
                    System.out.println("Corrupt the Blockchain");
                    System.out.println("Enter block ID of block to corrupt");
                    int id = Integer.parseInt(scanner.nextLine());
                    System.out.printf("Enter new data for block %d\n", id);
                    String d = scanner.nextLine();
                    req.setSelection(4);
                    req.setId(id);
                    req.setMessage(d);
                    break;
                case 5:
                    req.setSelection(5);
                    break;
                case 6:
                    req.setSelection(6);
```

```
                    break;
                default:
                    break;
            }
            out.println(gson.toJson(req));
            out.flush();
            ResponseMessage res = gson.fromJson(in.readLine(), ResponseMessage.class);
            System.out.println(res);
        } while (option != 6);
    }
    public static void main(String[] args) {
        BlockchainClient blockchainClient = new BlockchainClient();
        try {
            blockchainClient.init();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Task 1 Server Source Code

```
import com.google.gson.Gson;

import java.io.BufferedWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Timestamp;
import java.util.Scanner;

//https://github.com/CMU-Heinz-95702/lab4-http-server/tree/master/Example-Socket-Code
public class BlockchainServer {
    BlockChain chain;
    Gson gson;
    public BlockchainServer() {
        gson = new Gson();
    }

    public ResponseMessage exec(RequestMessage req) {
        int selection = req.selection;
        ResponseMessage res = new ResponseMessage();
        res.setSelection(req.getSelection());
        String resMes = "";
        long start;
        long end;
        switch (selection) {
            case 0:
```

```
                res.setChainHash(chain.getChainHash());
                res.setHps(chain.getHashesPerSecond());
                res.setRecentNonce(chain.getLatestBlock().getNonce());
                res.setSize(chain.getChainSize());
                res.setTotalDiff(chain.getTotalDifficulty());
                res.setTotalHashes(chain.getTotalExpectedHashes());
                break;
            case 1:
                System.out.println("Adding a block");
                start = System.currentTimeMillis();
                Block newBlock = new Block(chain.getChainSize(), new
Timestamp(System.currentTimeMillis()),
                        req.getMessage(), req.getDifficulty());
                newBlock.setPreviousHash(chain.chainHash);
                newBlock.proofOfWork();
                chain.addBlock(newBlock);
                end = System.currentTimeMillis();
                resMes = "Total execution time to add this block was "+ (end-start)+"
milliseconds";
                break;
            case 2:
                start = System.currentTimeMillis();
                resMes = chain.isChainValid();
                resMes =  "Chain verification: " + resMes;
                System.out.println(resMes);
                end = System.currentTimeMillis();
                resMes += "\n"+"Total execution time to verify the chain was "+(end-start)+"
milliseconds";
                break;
            case 3:
                System.out.println("View the Blockchain");
                resMes = chain.toString();
                break;
            case 4:
                System.out.println("Corrupt the Blockchain");
                int id = req.getId();
                String d = req.getMessage();
                chain.getBlock(id).setData(d);
                resMes = "Block "+id+" now holds "+d;
                break;
            case 5:
                start = System.currentTimeMillis();
                System.out.println("Repairing the entire chain");
                chain.repairChain();
                end = System.currentTimeMillis();
                resMes = "Total execution time required to repair the chain was "+(end-
start)+" milliseconds";
                break;
            default:
                break;
        }
```

```
        System.out.println("Setting response to " + resMes);
        res.setResponse(resMes);
        System.out.println(gson.toJson(res));
        return res;
    }

    public void init() {
        Socket clientSocket = null;
        chain = new BlockChain();
        try {
            int serverPort = 7777;
            ServerSocket listenSocket = new ServerSocket(serverPort);
            System.out.println("Blockchain server running");
            while (true) {
                String request;
                clientSocket = listenSocket.accept();
                Scanner scanner = new Scanner(clientSocket.getInputStream());
                PrintWriter outToSocket = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                System.out.println("We have a visitor");
                do {
                    request = scanner.nextLine();
                    RequestMessage req = gson.fromJson(request, RequestMessage.class);
                    ResponseMessage res = exec(req);
                    outToSocket.println(gson.toJson(res));
                    outToSocket.flush();
                    if (req.getSelection() == 6) break;
                } while (scanner.hasNextLine());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        BlockchainServer blockchainServer = new BlockchainServer();
        blockchainServer.init();
    }
}

```