# Project 3

Ruidi Chang, [ruidic@andrew.cmu.edu](mailto:ruidic@andrew.cmu.edu)

Table of Contents

[Project 3 1](#_Toc118146898)

[Task 0 1](#_Toc118146899)

[Task 0 Execution 1](#_Toc118146900)

[Task 0 Block.java 5](#_Toc118146901)

[Task 0 BlockChain.java 8](#_Toc118146902)

[Task 1 13](#_Toc118146903)

[Task 1 Client Side Execution 13](#_Toc118146904)

[Task 1 Server Side Execution 18](#_Toc118146905)

[Task 1 Client Source Code 19](#_Toc118146906)

[Task 1 Server Source Code 22](#_Toc118146907)

## Task 0

### 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: 2136

Expected total hashes required for the whole chain: 256.0

Nonce for the most recent block: 202

Chain hash: 008D30E5371C15964CC2D5BE9B8918CC79AE648D41DDBF2CA79C3BE263B37A9A

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 19 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 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 Donna 10 DSCoin

Total execution time to add this block was 10 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

Chain verification: TRUE

Total execution time to verify 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

3

View the block chain

{"ds\_chain":[{"index":0,"timestamp":"Oct 29, 2022, 11:13:27 PM","Tx":"Genesis","nonce":202,"difficulty":2},{"index":1,"timestamp":"Oct 29, 2022, 11:13:43 PM","Tx":"Alice pays Bob 100 DSCoin","previousHash":"008D30E5371C15964CC2D5BE9B8918CC79AE648D41DDBF2CA79C3BE263B37A9A","nonce":209,"difficulty":2},{"index":2,"timestamp":"Oct 29, 2022, 11:13:52 PM","Tx":"Bob pays Carol 50 DSCoin","previousHash":"00CE1B4152871601AFB8FA7C54213078841E9772F01663064636F8F75936B464","nonce":279,"difficulty":2},{"index":3,"timestamp":"Oct 29, 2022, 11:14:02 PM","Tx":"Carol pays Donna 10 DSCoin","previousHash":"00A932007414D647559A23685036886F7E6AEAC90B5D62EED2AC238727106BDE","nonce":97,"difficulty":2}],"chainHash":"00D2F6CDFB5B3B119773BA4E86251D2A8D0EE5503034999EA33C938619A29318","hashesPerSecond":2136}

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

{"ds\_chain":[{"index":0,"timestamp":"Oct 29, 2022, 11:13:27 PM","Tx":"Genesis","nonce":202,"difficulty":2},{"index":1,"timestamp":"Oct 29, 2022, 11:13:43 PM","Tx":"Alice pays Bob 76 DSCoin","previousHash":"008D30E5371C15964CC2D5BE9B8918CC79AE648D41DDBF2CA79C3BE263B37A9A","nonce":209,"difficulty":2},{"index":2,"timestamp":"Oct 29, 2022, 11:13:52 PM","Tx":"Bob pays Carol 50 DSCoin","previousHash":"00CE1B4152871601AFB8FA7C54213078841E9772F01663064636F8F75936B464","nonce":279,"difficulty":2},{"index":3,"timestamp":"Oct 29, 2022, 11:14:02 PM","Tx":"Carol pays Donna 10 DSCoin","previousHash":"00A932007414D647559A23685036886F7E6AEAC90B5D62EED2AC238727106BDE","nonce":97,"difficulty":2}],"chainHash":"00D2F6CDFB5B3B119773BA4E86251D2A8D0EE5503034999EA33C938619A29318","hashesPerSecond":2136}

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

Chain verification: Improper hash on node 1 Does not begin with 00

Total execution time to verify the chain was 10 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

Total execution time required to repair 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

2

Chain verification: TRUE

Total execution time to verify the chain was 4 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

Donna pays Sean 25 DSCoin

Total execution time to add this block was 54 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: 2136

Expected total hashes required for the whole chain: 66560.0

Nonce for the most recent block: 5663

Chain hash: 00009E446B5A3FF39B9F5B556802B31B1EF0B97B57536B6B73E86C26911E7FBC

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

package edu.cmu.ruidic;  
  
import java.math.BigInteger;  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
import java.sql.Timestamp;  
  
  
  
import com.google.gson.Gson;  
import jakarta.xml.bind.DatatypeConverter;  
  
*/\*\*  
 \* Block class  
 \** ***@author*** *Ruidi Chang  
 \** ***@email*** *ruidic@andrew.cmu.edu  
 \*/*public class Block {  
 private int index;  
 private Timestamp timestamp;  
 private String Tx;  
 private String previousHash;  
 private BigInteger nonce = BigInteger.*ZERO*;  
 private int difficulty;  
  
  
 */\*\*  
 \* This the Block constructor.  
 \** ***@param*** *index  
 \** ***@param*** *timestamp  
 \** ***@param*** *Tx  
 \** ***@param*** *difficulty  
 \*/* public Block(int index, Timestamp timestamp, String Tx, int difficulty){  
 this.index = index;  
 this.timestamp = timestamp;  
 this.Tx = Tx;  
 this.difficulty = difficulty;  
 }  
  
 */\*\*  
 \* This method computes a hash of the concatenation of the index, timestamp, data, previousHash, nonce, and difficulty.  
 \** ***@return*** *a String holding Hexadecimal characters  
 \** ***@throws*** *NoSuchAlgorithmException  
 \*/* public String calculateHash() throws NoSuchAlgorithmException {  
 String message = index + timestamp.toString() + Tx + previousHash + nonce + difficulty;  
 MessageDigest md = MessageDigest.*getInstance*("SHA-256");  
 md.update(message.getBytes());  
 return DatatypeConverter.*printHexBinary*(md.digest());  
 }  
  
 public String getData(){return Tx;}  
  
 */\*\*  
 \* This method returns the nonce for this block.  
 \** ***@return*** *a BigInteger representing the nonce for this block.  
 \*/* public BigInteger getNonce(){return nonce;}  
  
 public int getDifficulty(){return difficulty;}  
  
 public int getIndex(){return index;}  
  
 public String getPreviousHash(){return previousHash;}  
  
 public Timestamp getTimestamp(){return timestamp;}  
  
 */\*\*  
 \* The proof of work methods finds a good hash. It increments the nonce until it produces a good hash.  
 \** ***@return*** *a String with a hash that has the appropriate number of leading hex zeroes.  
 \** ***@throws*** *NoSuchAlgorithmException  
 \*/* public String proofOfWork() throws NoSuchAlgorithmException {  
 String hash = calculateHash();  
 while (difficulty>hash.length() - hash.replaceAll("^0+", "").length()){  
 // update the nonce  
 nonce = nonce.add(BigInteger.*ONE*);  
 hash = calculateHash();  
 }  
 return hash;  
 }  
  
 public void setData(String Tx) {  
 this.Tx = Tx;  
 }  
  
 public void setDifficulty(int difficulty) {  
 this.difficulty = difficulty;  
 }  
  
 public void setIndex(int index) {  
 this.index = index;  
 }  
  
 public void setTimestamp(Timestamp timestamp) {  
 this.timestamp = timestamp;  
 }  
  
 public void setPreviousHash(String previousHash) {  
 this.previousHash = previousHash;  
 }  
  
 */\*\*  
 \* Override Java's toString method  
 \** ***@return*** *A JSON representation of all of this block's data is returned.  
 \*/* @Override  
 public String toString() {  
 Gson gson = new Gson();  
 return gson.toJson(this);  
 }  
  
}

### Task 0 BlockChain.java

package edu.cmu.ruidic;  
  
import com.google.gson.Gson;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
import java.sql.Timestamp;  
import java.util.ArrayList;  
*/\*\*  
 \* BlockChain class  
 \** ***@author*** *Ruidi Chang  
 \** ***@email*** *ruidic@andrew.cmu.edu  
 \*/*public class BlockChain {  
 private ArrayList<Block> ds\_chain;  
 private String chainHash;  
 private int hashesPerSecond;  
  
 public BlockChain() throws NoSuchAlgorithmException {  
 this.ds\_chain = new ArrayList<Block>();  
 Block genesis = new Block(0, new Timestamp(System.*currentTimeMillis*()),"Genesis", 2);  
 genesis.proofOfWork();  
 addBlock(genesis);  
 computeHashesPerSecond();  
 };  
  
 public static void main(String[] args) throws IOException, NoSuchAlgorithmException {  
 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");  
 BlockChain blockChain = new BlockChain();  
 BufferedReader input = new BufferedReader(new InputStreamReader(System.*in*));  
 String nextLine;  
 long start, end;  
 while ((Integer.*parseInt*(nextLine = input.readLine()))!=6) {  
 switch (Integer.*parseInt*(nextLine)){  
 case 0:  
 System.*out*.println("Current size of chain: " + blockChain.getChainSize());  
 System.*out*.println("Difficulty of most recent block: " + blockChain.getLatestBlock().getDifficulty());  
 System.*out*.println("Total difficulty for all blocks: " + blockChain.getTotalDifficulty());  
 System.*out*.println("Approximate hashes per second on this machine: " + blockChain.getHashesPerSecond());  
 System.*out*.println("Expected total hashes required for the whole chain: " + blockChain.getTotalExpectedHashes());  
 System.*out*.println("Nonce for the most recent block: " + blockChain.getLatestBlock().getNonce());  
 System.*out*.println("Chain hash: " + blockChain.getChainHash());  
 break;  
 case 1:  
 // The execution time increases exponentially  
 // it takes less than 10 milliseconds in difficulty 1 and 2  
 // it takes about 10-99 milliseconds in difficulty 3 and 4  
 // it takes 100-999 digits milliseconds in difficulty 5  
 // it takes a long time in difficulty 6  
 // For difficulty 7, it cannot complete  
 System.*out*.println("Enter difficulty > 0");  
 int diff = Integer.*parseInt*(input.readLine());  
 System.*out*.println("Enter transaction");  
 String transaction = input.readLine();  
 start = System.*currentTimeMillis*();  
 Block newBlock = new Block(blockChain.getChainSize(), new Timestamp(start), transaction, diff);  
 newBlock.setPreviousHash(blockChain.getChainHash());  
 newBlock.proofOfWork();  
 blockChain.addBlock(newBlock);  
 end = System.*currentTimeMillis*();  
 System.*out*.println("Total execution time to add this block was " + (end - start) + " milliseconds");  
 break;  
 case 2:  
 // it is O(1), about several milliseconds  
 start = System.*currentTimeMillis*();  
 String valid = blockChain.isChainValid();  
 end = System.*currentTimeMillis*();  
 System.*out*.println("Chain verification: " + valid);  
 System.*out*.println("Total execution time to verify the chain was " + (end - start) + " milliseconds");  
 break;  
 case 3:  
 System.*out*.println("View the block chain");  
 System.*out*.println(blockChain);  
 break;  
 case 4:  
 System.*out*.println("Corrupt the Blockchain");  
 System.*out*.println("Enter block ID of block to corrupt");  
 int index = Integer.*parseInt*(input.readLine());  
 System.*out*.println("Enter new data for block "+ index);  
 String data = input.readLine();  
 blockChain.getBlock(index).setData(data);  
 System.*out*.println("Block "+ index +" now holds " + data);  
 break;  
 case 5:  
 // It takes super super super long time when high difficulty  
 start = System.*currentTimeMillis*();  
 blockChain.repairChain();  
 end = System.*currentTimeMillis*();  
 System.*out*.println("Total execution time required to repair the chain was "+ (end - start) +" milliseconds");  
 break;  
 default:  
 break;  
 }  
 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");  
 }  
 }  
  
 public void addBlock(Block block) throws NoSuchAlgorithmException {  
 ds\_chain.add(block);  
 chainHash = block.calculateHash();  
 }  
  
 */\*\*  
 \* This method computes exactly 2 million hashes and times how long that process takes.  
 \** ***@throws*** *NoSuchAlgorithmException  
 \*/* public void computeHashesPerSecond() throws NoSuchAlgorithmException {  
 String message = "00000000";  
 int number = 2000000;  
 long start = System.*currentTimeMillis*();  
 MessageDigest md = MessageDigest.*getInstance*("SHA-256");  
 for (int i = 0; i < number; i++) {  
 md.digest(message.getBytes());  
 }  
 long end = System.*currentTimeMillis*();  
 hashesPerSecond = (int) (number/(end-start));  
 }  
  
 public Block getBlock(int i){return ds\_chain.get(i);}  
  
 public String getChainHash() {  
 return chainHash;  
 }  
  
 public int getChainSize(){  
 return ds\_chain.size();  
 }  
  
 public int getHashesPerSecond(){  
 return hashesPerSecond;  
 }  
  
 public Block getLatestBlock(){  
 return getBlock(getChainSize()-1);  
 }  
  
 public Timestamp getTime(){  
 return new Timestamp(System.*currentTimeMillis*());  
 }  
  
 */\*\*  
 \* Compute and return the total difficulty of all blocks on the chain. Each block knows its own difficulty.  
 \** ***@return*** *totalDifficulty  
 \*/* public int getTotalDifficulty(){  
 int total\_difficulty = 0;  
 for(Block block : ds\_chain){  
 total\_difficulty += block.getDifficulty();  
 }  
 return total\_difficulty;  
 }  
  
 */\*\*  
 \* Compute and return the expected number of hashes required for the entire chain.  
 \** ***@return*** *totalExpectedHashes  
 \*/* public double getTotalExpectedHashes(){  
 double total\_ExpectedHashes = 0;  
 for(Block block : ds\_chain){  
 total\_ExpectedHashes += Math.*pow*(16, block.getDifficulty());  
 }  
 return total\_ExpectedHashes;  
 }  
  
 */\*\*  
 \* Check the chain is valid or not  
 \** ***@return*** *"TRUE" if the chain is valid, otherwise return a string with an appropriate error message  
 \** ***@throws*** *NoSuchAlgorithmException  
 \*/* public String isChainValid() throws NoSuchAlgorithmException {  
 // If the chain only contains one block, the genesis block at position 0,  
 if (getChainSize()==1){  
 // this routine computes the hash of the block  
 // and checks that the hash has the requisite number of leftmost 0's (proof of work)  
 // as specified in the difficulty field.  
 if (getLatestBlock().calculateHash().equals(getLatestBlock().proofOfWork())){  
 // It also checks that the chain hash is equal to this computed hash.  
 if (getLatestBlock().calculateHash().equals(getChainHash())){  
 // Otherwise, return the string "TRUE".  
 return "TRUE";  
 }  
 else{  
 // If either check fails, return an error message.  
 return "Chain Hash is NOT Equal to Computed Hash";  
 }  
 }else{  
 // If either check fails, return an error message.  
 return "Proof of Work Error";  
 }  
 // If the chain has more blocks than one,  
 }else{  
 // begin checking from block one.  
 Block genesis = getBlock(0);  
 String previous\_hash = genesis.calculateHash();  
 if (!getLatestBlock().calculateHash().equals(getChainHash())){  
 return "Chain Hash is NOT Equal to Computed Hash";  
 }  
 if (!genesis.calculateHash().equals(genesis.proofOfWork())){  
 return "Proof of Work Error";  
 }  
 // Continue checking until you have validated the entire chain.  
 for (int i = 1; i < getChainSize(); i++) {  
 Block b = ds\_chain.get(i);  
 if (!b.getPreviousHash().equals(previous\_hash)) {  
 return "Block "+i+" Previous Hash Mismatch!";  
 } else if (!b.calculateHash().equals(b.proofOfWork())) {  
 String out="";  
 for (int j = 0; j < b.getDifficulty(); j++) {  
 out=out+"0";  
 }  
 return "Improper hash on node " + i + " Does not begin with " + out;  
 }else{  
 previous\_hash = b.calculateHash();  
 }  
 }  
 }  
 // At the end, check that the chain hash is also correct.  
 return "TRUE";  
 }  
  
 */\*\*  
 \* This routine repairs the chain.  
 \** ***@throws*** *NoSuchAlgorithmException  
 \*/* public void repairChain() throws NoSuchAlgorithmException {  
 Block genesis = getBlock(0);  
 if(getChainSize()==1){  
 // Recompute illegal hashes  
 if(!genesis.calculateHash().equals(getLatestBlock().proofOfWork())){  
 genesis.proofOfWork();  
 }else if(!genesis.calculateHash().equals(getChainHash())){  
 chainHash = genesis.calculateHash();  
 }  
 }else{  
 String previous\_hash = genesis.calculateHash();  
 if(!genesis.calculateHash().equals(genesis.proofOfWork())){  
 genesis.proofOfWork();  
 }  
 Block b;  
 // Recompute illegal hashes  
 for(int i = 1; i < getChainSize(); i++){  
 b = ds\_chain.get(i);  
 if(!b.getPreviousHash().equals(previous\_hash)){  
 b.setPreviousHash(previous\_hash);  
 }  
 if(!b.calculateHash().equals(b.proofOfWork())){  
 b.proofOfWork();  
 }  
 previous\_hash = b.calculateHash();  
 }  
 if (!previous\_hash.equals(getChainHash())) {  
 chainHash = previous\_hash;  
 }  
 }  
 }  
  
 */\*\*  
 \* Override Java's toString method  
 \** ***@return*** *A JSON representation of all of this block's data is returned.  
 \*/* @Override  
 public String toString() {  
 Gson gson = new Gson();  
 return gson.toJson(this);  
 }  
  
}

## Task 1

### Task 1 Client Side 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: 0

Approximate hashes per second on this machine: 2020

Expected total hashes required for the whole chain: 256.0

Nonce for the most recent block: 17

Chain hash: 0075D0CCA07F2F4863AE0ADF03B24AE50195B5E22B89D6F16AA0542565F012F3

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 4 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 27 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 Donna 10 DSCoin

Total execution time to add this block was 36 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

Chain verification: TRUE

Total execution time to verify the chain was 14 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

View the block chain

{"ds\_chain":[{"index":0,"timestamp":"Oct 31, 2022, 10:10:30 PM","Tx":"Genesis","nonce":17,"difficulty":2},{"index":1,"timestamp":"Oct 31, 2022, 10:10:47 PM","Tx":"Alice pays Bob 100 DSCoin","previousHash":"0075D0CCA07F2F4863AE0ADF03B24AE50195B5E22B89D6F16AA0542565F012F3","nonce":8,"difficulty":2},{"index":2,"timestamp":"Oct 31, 2022, 10:10:57 PM","Tx":"Bob pays Carol 50 DSCoin","previousHash":"004A51A70906095EAF9DD23D33D84DFC0A03D03F2DD4FBC597DB560A51E5D23E","nonce":334,"difficulty":2},{"index":3,"timestamp":"Oct 31, 2022, 10:11:05 PM","Tx":"Carol pays Donna 10 DSCoin","previousHash":"00C4FB37ABC53DA6DC51C80A6AA5EA5326C71EB7CC16ECB222C451DB94DDCA19","nonce":568,"difficulty":2}],"chainHash":"0067A28228A4F027298D8F814942EA5C554B5A70AB2675E8AADA423CC051E5B8","hashesPerSecond":2020}

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

{"ds\_chain":[{"index":0,"timestamp":"Oct 31, 2022, 10:10:30 PM","Tx":"Genesis","nonce":17,"difficulty":2},{"index":1,"timestamp":"Oct 31, 2022, 10:10:47 PM","Tx":"Alice pays Bob 76 DSCoin","previousHash":"0075D0CCA07F2F4863AE0ADF03B24AE50195B5E22B89D6F16AA0542565F012F3","nonce":8,"difficulty":2},{"index":2,"timestamp":"Oct 31, 2022, 10:10:57 PM","Tx":"Bob pays Carol 50 DSCoin","previousHash":"004A51A70906095EAF9DD23D33D84DFC0A03D03F2DD4FBC597DB560A51E5D23E","nonce":334,"difficulty":2},{"index":3,"timestamp":"Oct 31, 2022, 10:11:05 PM","Tx":"Carol pays Donna 10 DSCoin","previousHash":"00C4FB37ABC53DA6DC51C80A6AA5EA5326C71EB7CC16ECB222C451DB94DDCA19","nonce":568,"difficulty":2}],"chainHash":"0067A28228A4F027298D8F814942EA5C554B5A70AB2675E8AADA423CC051E5B8","hashesPerSecond":2020}

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

Chain verification: Improper hash on node 1 Does not begin with 00

Total execution time to verify the chain was 6 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

Total execution time required to repair the chain was 15 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

Chain verification: TRUE

Total execution time to verify the chain was 1 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

Donna pays Sean 25 DSCoin

Total execution time to add this block was 58 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: 0

Approximate hashes per second on this machine: 2020

Expected total hashes required for the whole chain: 66560.0

Nonce for the most recent block: 3614

Chain hash: 000053449FB4A4B29E6E704CF8D55A361DD2375C8BBC67D31DC696FFECF9D2F5

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 1 Server Side Execution

Blockchain server running

We have a visitor

Response : {"selection":0,"size":1,"chainHash":"0075D0CCA07F2F4863AE0ADF03B24AE50195B5E22B89D6F16AA0542565F012F3","totalHashes":256.0,"totalDiff":0,"recentNonce":17,"diff":2,"hps":2020}

Adding a block

...{"selection":1,"response":"Total execution time to add this block was 4 milliseconds"}

Adding a block

...{"selection":1,"response":"Total execution time to add this block was 27 milliseconds"}

Adding a block

...{"selection":1,"response":"Total execution time to add this block was 36 milliseconds"}

Verifying entire chain

Chain verification: TRUE

Total execution time to verify the chain was 14 milliseconds

Setting response to Total execution time to verify the chain was 14 milliseconds

View the Blockchain

Setting response to {"ds\_chain":[{"index":0,"timestamp":"Oct 31, 2022, 10:10:30 PM","Tx":"Genesis","nonce":17,"difficulty":2},{"index":1,"timestamp":"Oct 31, 2022, 10:10:47 PM","Tx":"Alice pays Bob 100 DSCoin","previousHash":"0075D0CCA07F2F4863AE0ADF03B24AE50195B5E22B89D6F16AA0542565F012F3","nonce":8,"difficulty":2},{"index":2,"timestamp":"Oct 31, 2022, 10:10:57 PM","Tx":"Bob pays Carol 50 DSCoin","previousHash":"004A51A70906095EAF9DD23D33D84DFC0A03D03F2DD4FBC597DB560A51E5D23E","nonce":334,"difficulty":2},{"index":3,"timestamp":"Oct 31, 2022, 10:11:05 PM","Tx":"Carol pays Donna 10 DSCoin","previousHash":"00C4FB37ABC53DA6DC51C80A6AA5EA5326C71EB7CC16ECB222C451DB94DDCA19","nonce":568,"difficulty":2}],"chainHash":"0067A28228A4F027298D8F814942EA5C554B5A70AB2675E8AADA423CC051E5B8","hashesPerSecond":2020}

Corrupt the Blockchain

Block 1 now holds Alice pays Bob 76 DSCoin

Setting response to Block 1 now holds Alice pays Bob 76 DSCoin

View the Blockchain

Setting response to {"ds\_chain":[{"index":0,"timestamp":"Oct 31, 2022, 10:10:30 PM","Tx":"Genesis","nonce":17,"difficulty":2},{"index":1,"timestamp":"Oct 31, 2022, 10:10:47 PM","Tx":"Alice pays Bob 76 DSCoin","previousHash":"0075D0CCA07F2F4863AE0ADF03B24AE50195B5E22B89D6F16AA0542565F012F3","nonce":8,"difficulty":2},{"index":2,"timestamp":"Oct 31, 2022, 10:10:57 PM","Tx":"Bob pays Carol 50 DSCoin","previousHash":"004A51A70906095EAF9DD23D33D84DFC0A03D03F2DD4FBC597DB560A51E5D23E","nonce":334,"difficulty":2},{"index":3,"timestamp":"Oct 31, 2022, 10:11:05 PM","Tx":"Carol pays Donna 10 DSCoin","previousHash":"00C4FB37ABC53DA6DC51C80A6AA5EA5326C71EB7CC16ECB222C451DB94DDCA19","nonce":568,"difficulty":2}],"chainHash":"0067A28228A4F027298D8F814942EA5C554B5A70AB2675E8AADA423CC051E5B8","hashesPerSecond":2020}

Verifying entire chain

Chain verification: Improper hash on node 1 Does not begin with 00

Total execution time to verify the chain was 6 milliseconds

Setting response to Total execution time to verify the chain was 6 milliseconds

Repairing the entire chain

Setting response to Total execution time required to repair the chain was 15 milliseconds

Verifying entire chain

Chain verification: TRUE

Total execution time to verify the chain was 1 milliseconds

Setting response to Total execution time to verify the chain was 1 milliseconds

Adding a block

...{"selection":1,"response":"Total execution time to add this block was 58 milliseconds"}

Response : {"selection":0,"size":5,"chainHash":"000053449FB4A4B29E6E704CF8D55A361DD2375C8BBC67D31DC696FFECF9D2F5","totalHashes":66560.0,"totalDiff":0,"recentNonce":3614,"diff":4,"hps":2020}

### Task 1 Client Source Code

package edu.cmu;  
  
import com.google.gson.Gson;  
  
import java.io.\*;  
import java.net.Socket;  
*/\*\*  
 \* BlockChainClientTCP class  
 \** ***@author*** *Ruidi Chang  
 \** ***@email*** *ruidic@andrew.cmu.edu  
 \*/*public class BlockChainClientTCP {  
 static BufferedReader *in*;  
 static PrintWriter *out*;  
 static ResponseMessage *response*;  
 static Gson *gson* = new Gson();  
 static BufferedReader *typed*;  
 static Socket *clientSocket* = null;  
  
 public static void main(String args[]) {  
 try {  
 int serverPort = 7777;  
 *clientSocket* = new Socket("localhost", serverPort);  
 *in* = new BufferedReader(new InputStreamReader(*clientSocket*.getInputStream()));  
 *out* = new PrintWriter(new BufferedWriter(new OutputStreamWriter(*clientSocket*.getOutputStream())));  
 *typed* = new BufferedReader(new InputStreamReader(System.*in*));  
 String nextLine;  
 String id;  
 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");  
 while ((nextLine = *typed*.readLine()) != null) {  
 int option = Integer.*parseInt*(nextLine);  
 *Operation*(option);  
 }  
 } catch (IOException e) {  
 System.*out*.println("IO Exception:" + e.getMessage());  
 } finally {  
 try {  
 if (*clientSocket* != null) {  
 *clientSocket*.close();  
 }  
 } catch (IOException e) {  
 // ignore exception on close  
 }  
 }  
 }  
  
 */\*\*  
 \* Handle request  
 \** ***@param*** *option  
 \** ***@throws*** *IOException  
 \*/* public static void Operation(int option) throws IOException {  
 String transaction;  
 switch (option) {  
 case 0:  
 *out*.println(new RequestMessage(0));  
 *out*.flush();  
 *response* = *gson*.fromJson(*in*.readLine(), ResponseMessage.class);  
 System.*out*.println("Current size of chain: " + *response*.size);  
 System.*out*.println("Difficulty of most recent block: " + *response*.diff);  
 System.*out*.println("Total difficulty for all blocks: " + *response*.totalDiff);  
 System.*out*.println("Approximate hashes per second on this machine: " + *response*.hps);  
 System.*out*.println("Expected total hashes required for the whole chain: " + *response*.totalHashes);  
 System.*out*.println("Nonce for the most recent block: " + *response*.recentNonce);  
 System.*out*.println("Chain hash: " + *response*.chainHash);  
 break;  
 case 1:  
 System.*out*.println("Enter difficulty > 0 ");  
 int difficulty = Integer.*parseInt*(*typed*.readLine());  
 System.*out*.println("Enter transaction");  
 transaction = *typed*.readLine();  
 *out*.println(new RequestMessage(1, transaction, difficulty));  
 *out*.flush();  
 *response* = *gson*.fromJson(*in*.readLine(), ResponseMessage.class);  
 System.*out*.println(*response*.responses);  
 break;  
 case 2:  
 *out*.println(new RequestMessage(2));  
 *out*.flush();  
 *response* = *gson*.fromJson(*in*.readLine(), ResponseMessage.class);  
 System.*out*.println(*response*.verification);  
 System.*out*.println(*response*.responses);  
 break;  
 case 3:  
 System.*out*.println("View the block chain");  
 *out*.println(new RequestMessage(3));  
 *out*.flush();  
 *response* = *gson*.fromJson(*in*.readLine(), ResponseMessage.class);  
 System.*out*.println(*response*.responses);  
 break;  
 case 4:  
 System.*out*.println("Corrupt the Blockchain");  
 System.*out*.println("Enter block ID of block to corrupt");  
 int id = Integer.*parseInt*(*typed*.readLine());  
 System.*out*.println("Enter new data for block " + id);  
 transaction = *typed*.readLine();  
 *out*.println(new RequestMessage(4, id, transaction));  
 *out*.flush();  
 *response* = *gson*.fromJson(*in*.readLine(), ResponseMessage.class);  
 System.*out*.println(*response*.responses);  
 break;  
 case 5:  
 *out*.println(new RequestMessage(5));  
 *out*.flush();  
 *response* = *gson*.fromJson(*in*.readLine(), ResponseMessage.class);  
 System.*out*.println(*response*.responses);  
 break;  
 case 6:  
 // handle request for exit  
 *out*.println(new RequestMessage(6));  
 *out*.flush();  
 *out*.close();  
 *clientSocket*.close();  
 System.*exit*(0);  
 }  
 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");  
 }  
}

### Task 1 Server Source Code

package edu.cmu;  
  
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.security.NoSuchAlgorithmException;  
import java.sql.Timestamp;  
import java.util.Scanner;  
  
import static edu.cmu.BlockChainClientTCP.*response*;  
*/\*\*  
 \* BlockChainServerTCP class  
 \** ***@author*** *Ruidi Chang  
 \** ***@email*** *ruidic@andrew.cmu.edu  
 \*/*public class BlockChainServerTCP {  
 static Socket *clientSocket* = null;  
 static Scanner *in*;  
 static PrintWriter *out*;  
 static RequestMessage *request*;  
 static Gson *gson* = new Gson();  
  
 public static void main(String args[]) throws IOException {  
 try {  
 int serverPort = 7777; // the server port we are using  
 // Create a new server socket  
 ServerSocket listenSocket = new ServerSocket(serverPort);  
 System.*out*.println("Blockchain server running");  
 BlockChain chain = new BlockChain();  
 while (true) {  
 try {  
 *clientSocket* = listenSocket.accept();  
 System.*out*.println("We have a visitor");  
 *in* = new Scanner(*clientSocket*.getInputStream());  
 *out* = new PrintWriter(new BufferedWriter(new OutputStreamWriter(*clientSocket*.getOutputStream())));  
 while (true) {  
 if (*in*.hasNext()) {  
 // Receive the request message from client  
 String operation = *in*.nextLine();  
 *request* = *gson*.fromJson(operation, RequestMessage.class);  
 // if user choose 6, break the loop  
 if (*request*.option==6) break;  
 *Operation*(*request*, chain);  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 } catch (NoSuchAlgorithmException e) {  
 throw new RuntimeException(e);  
 } finally {  
 try {  
 if (*clientSocket* != null) {  
 *clientSocket*.close();  
 }  
 } catch (IOException e) {  
 // ignore exception on close  
 }  
 }  
 }  
  
 private static void Operation(RequestMessage request, BlockChain chain) throws NoSuchAlgorithmException {  
 long start, end;  
 switch (request.option) {  
 case (0):  
 *response* = new ResponseMessage(0, chain);  
 System.*out*.println("Response : " + *response*);  
 break;  
 case (1):  
 System.*out*.println("Adding a block");  
 start = System.*currentTimeMillis*();  
 Block nb = new Block(chain.getChainSize(), new Timestamp(start), request.transaction, request.difficulty);  
 nb.setPreviousHash(chain.getChainHash());  
 nb.proofOfWork();  
 chain.addBlock(nb);  
 end = System.*currentTimeMillis*();  
 *response* = new ResponseMessage(1, "Total execution time to add this block was " + (end - start) + " milliseconds");  
 System.*out*.println("...{\"selection\":"+*response*.selection+",\"response\":\""+*response*.responses+"\"}");  
 break;  
 case (2):  
 System.*out*.println("Verifying entire chain");  
 start = System.*currentTimeMillis*();  
 String valid\_chain = chain.isChainValid();  
 end = System.*currentTimeMillis*();  
 System.*out*.println("Chain verification: " + valid\_chain);  
 System.*out*.println("Total execution time to verify the chain was " + (end - start) + " milliseconds");  
 System.*out*.println("Setting response to Total execution time to verify the chain was " + (end - start) + " milliseconds");  
 *response* = new ResponseMessage(2, "Total execution time to verify the chain was " + (end - start) + " milliseconds", "Chain verification: " + valid\_chain);  
 break;  
 case (3):  
 System.*out*.println("View the Blockchain");  
 *response* = new ResponseMessage(3, chain);  
 System.*out*.println("Setting response to "+*response*.responses);  
 break;  
 case (4):  
 System.*out*.println("Corrupt the Blockchain");  
 chain.getBlock(request.index).setData(request.transaction);  
 System.*out*.println("Block "+ request.index +" now holds " + request.transaction);  
 *response* = new ResponseMessage(4,"Block "+ request.index +" now holds " + request.transaction );  
 System.*out*.println("Setting response to "+*response*.responses);  
 break;  
 case (5):  
 System.*out*.println("Repairing the entire chain");  
 start = System.*currentTimeMillis*();  
 chain.repairChain();  
 end = System.*currentTimeMillis*();  
 *response* = new ResponseMessage(5, "Total execution time required to repair the chain was "+(end-start)+" milliseconds");  
 System.*out*.println("Setting response to Total execution time required to repair the chain was "+(end-start)+" milliseconds");  
 break;  
 case (6):  
 break;  
 }  
 *out*.println(*response*);  
 *out*.flush();  
 }  
  
}