Task 0 Execution

```
Block chain menu
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 recomputing the hashes.
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: 840336
Expected total hashes required for the whole chain: 256.0
Nonce for most recent block: 163
Chain hash: 007F0727E7EEBCB91E558DC5EA4302223632F5EB27B1BB34DC584D4886E1BD67
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
3
Enter the transaction
Sanjana pays Bob 100 dscoin
Time taken to add this block::596 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
3
Enter the transaction
Bob pays Sanjana 20 dscoin
Time taken to add this block::15 ms
```

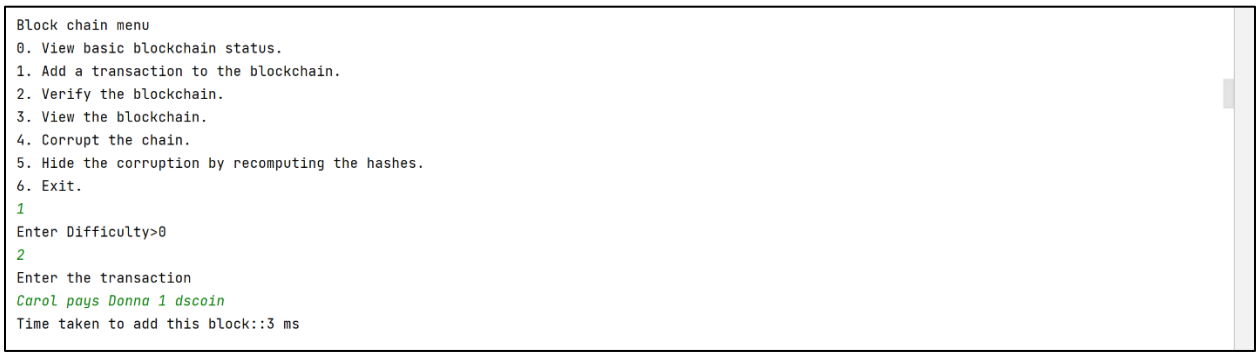
```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
4
Enter the transaction
Sanjana pays Carol 23 dscoin
Time taken to add this block::1308 ms
```

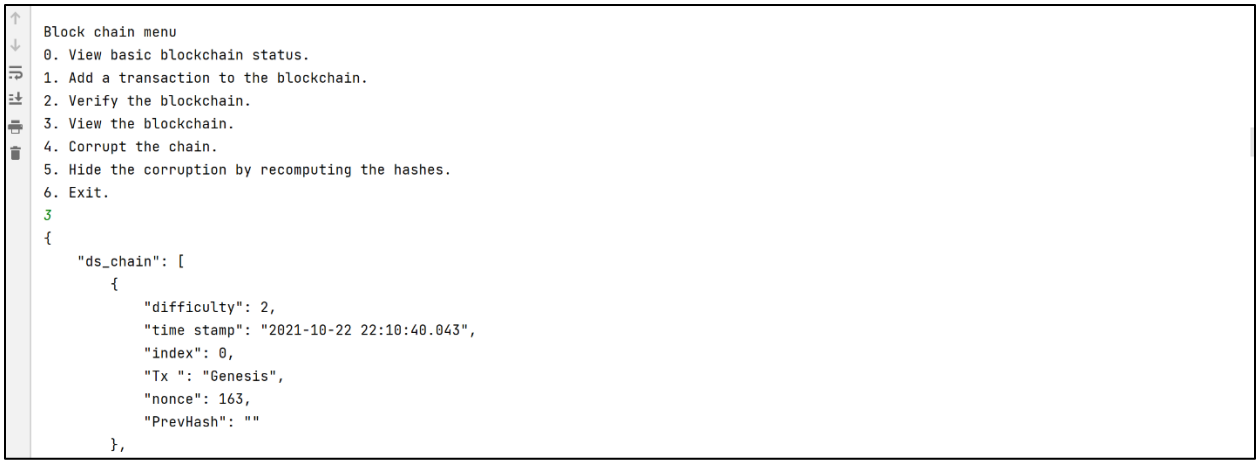
```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
5
Enter the transaction
Donna pays Sanjana 34 dscoin
Time taken to add this block::2612 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
3
{
    "ds_chain": [
        {
            "difficulty": 2,
            "time stamp": "2021-10-22 22:10:40.043",
            "index": 0,
            "Tx ": "Genesis",
            "nonce": 163,
            "PrevHash": ""
```


{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:10:40.043",
      "index": 0,
      "Tx ": "Genesis",
      "nonce": 163,
      "PrevHash": ""
    },
    {

```
        "difficulty": 3,
        "time stamp": "2021-10-22 22:10:56.808",
        "index": 1,
        "Tx ": "Sanjana pays Bob 100 dscoin",
        "nonce": 5842,
        "PrevHash": "007F0727E7EEBCB91E558DC5EA4302223632F5EB27B1BB34DC584D4886E1BD67"
      },
      {
        "difficulty": 3,
        "time stamp": "2021-10-22 22:11:08.673",
        "index": 2,
        "Tx ": "Bob pays Sanjana 20 dscoin",
        "nonce": 331,
        "PrevHash": "000EC588F02B2E17FE2DFCC2C139C6E74D6F4E7D1BDA5E4A93DB82212A1CC5B3"
      },
      {
        "difficulty": 4,
        "time stamp": "2021-10-22 22:11:17.888",
        "index": 3,
        "Tx ": "Sanjana pays Carol 23 dscoin",
        "nonce": 37118,
        "PrevHash": "0002069643C807ACD9F76F4AEDE285E21FCE845DBBCC6A7E5FD8EEE707D3F262"
      },
      {
        "difficulty": 5,
        "time stamp": "2021-10-22 22:11:30.001",
        "index": 4,
        "Tx ": "Donna pays Sanjana 34 dscoin",
        "nonce": 203713,
        "PrevHash": "00007617987F97BE08031168DD26253EB930744909D7E92DF553476501CC57AF"
      }
    ],
    "chainHash": "000004BF0776C245060453F2B49DAB5FEC7173B886F3C4480F59B0AD76C38481"
}
```

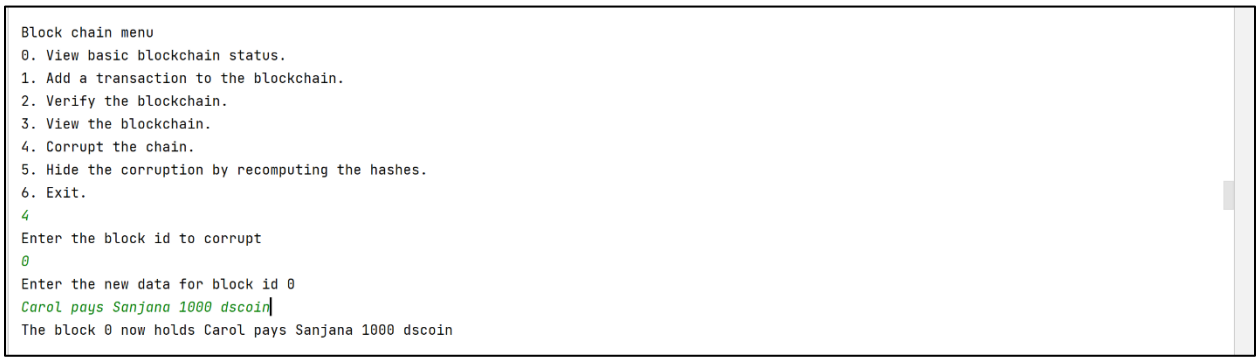



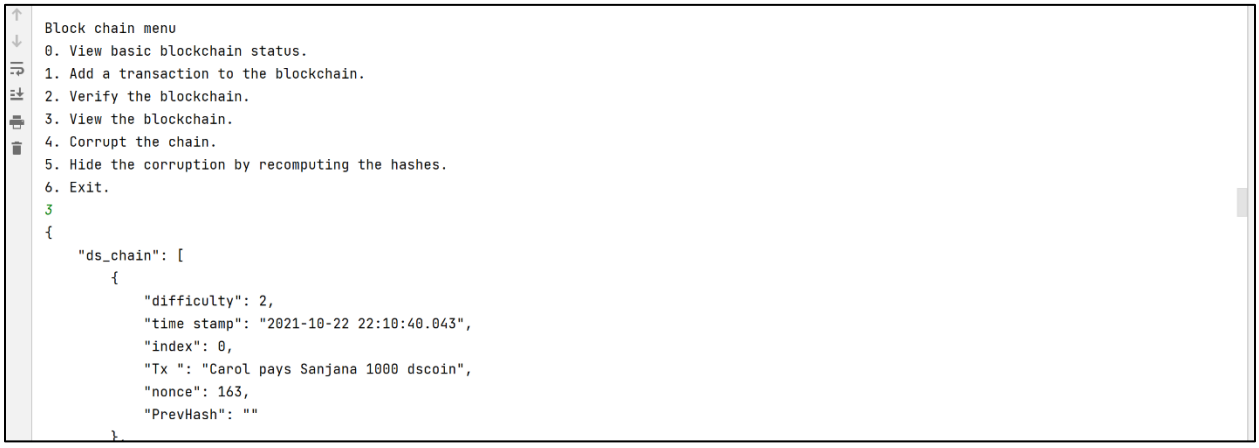


```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
3
{
    "ds_chain": [
        {
            "difficulty": 2,
            "time stamp": "2021-10-22 22:10:40.043",
            "index": 0,
            "Tx ": "Genesis",
            "nonce": 163,
            "PrevHash": ""
        },
```


```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:10:40.043",
      "index": 0,
      "Tx ": "Genesis",
      "nonce": 163,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:10:56.808",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 5842,
      "PrevHash": "007F0727E7EEBCB91E558DC5EA4302223632F5EB27B1BB34DC584D4886E1BD67"
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:11:08.673",
      "index": 2,
      "Tx ": "Bob pays Sanjana 20 dscoin",
      "nonce": 331,
      "PrevHash": "000EC588F02B2E17FE2DFCC2C139C6E74D6F4E7D1BDA5E4A93DB82212A1CC5B3"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:11:17.888",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 37118,
      "PrevHash": "0002069643C807ACD9F76F4AEDE285E21FCE845DBBCC6A7E5FD8EEE707D3F262"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:11:30.001",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 203713,
      "PrevHash": "00007617987F97BE08031168DD26253EB930744909D7E92DF553476501CC57AF"
    },
```

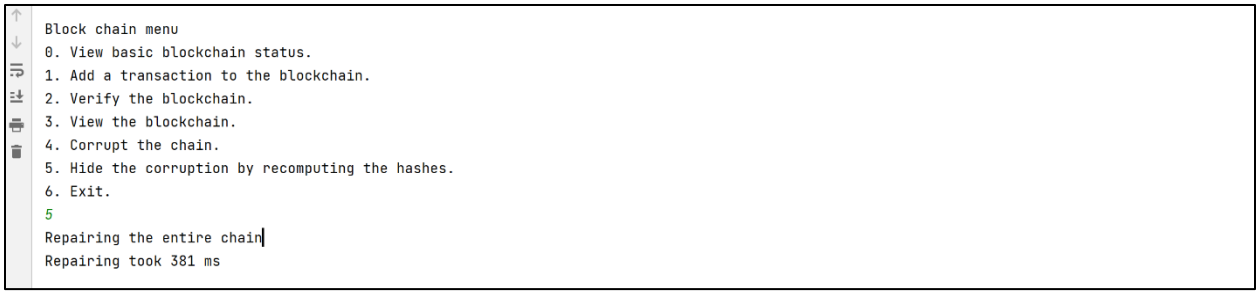
```
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:11:58.826",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 164,
      "PrevHash": "000004BF0776C245060453F2B49DAB5FEC7173B886F3C4480F59B0AD76C38481"
    }
  ],
  "chainHash": "00374400E349890EF84C5CBF5AC3C64780C234388A29B47568AEB78E6AAA6C8C"
}
```

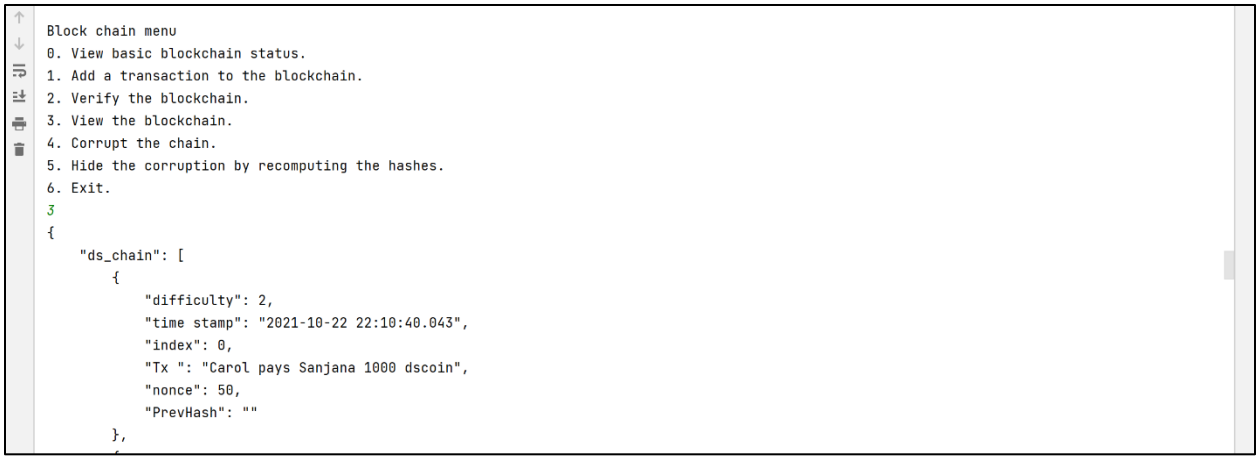






```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:10:40.043",
      "index": 0,
      "Tx ": "Carol pays Sanjana 1000 dscoin",
      "nonce": 163,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:10:56.808",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 5842,
      "PrevHash": "007F0727E7EEBCB91E558DC5EA4302223632F5EB27B1BB34DC584D4886E1BD67"
```

```
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:11:08.673",
      "index": 2,
      "Tx ": "Bob pays Sanjana 20 dscoin",
      "nonce": 331,
      "PrevHash": "000EC588F02B2E17FE2DFCC2C139C6E74D6F4E7D1BDA5E4A93DB82212A1CC5B3"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:11:17.888",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 37118,
      "PrevHash": "0002069643C807ACD9F76F4AEDE285E21FCE845DBBCC6A7E5FD8EEE707D3F262"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:11:30.001",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 203713,
      "PrevHash": "00007617987F97BE08031168DD26253EB930744909D7E92DF553476501CC57AF"
    },
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:11:58.826",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 164,
      "PrevHash": "000004BF0776C245060453F2B49DAB5FEC7173B886F3C4480F59B0AD76C38481"
    }
  ],
  "chainHash": "00374400E349890EF84C5CBF5AC3C64780C234388A29B47568AEB78E6AAA6C8C"
}
```

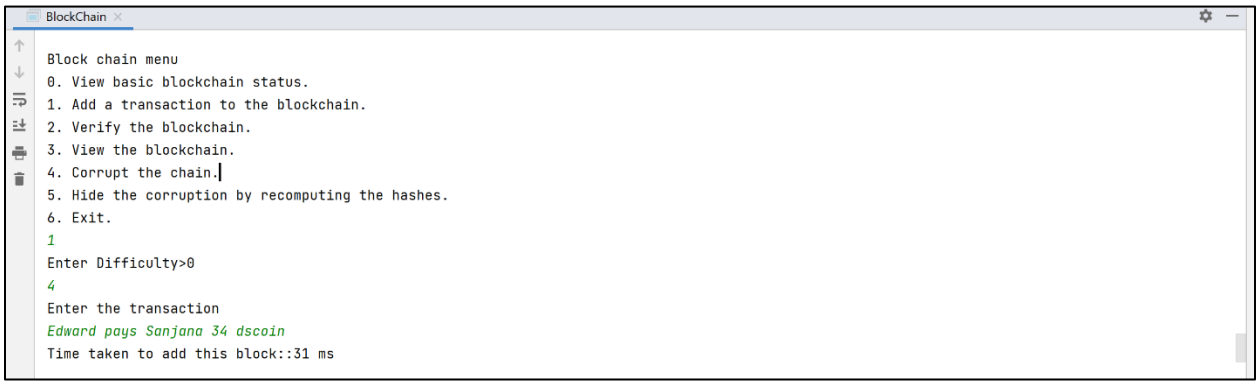



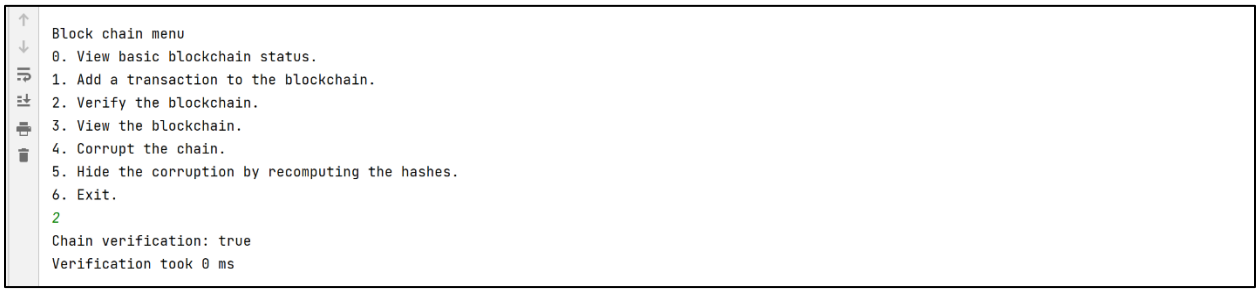


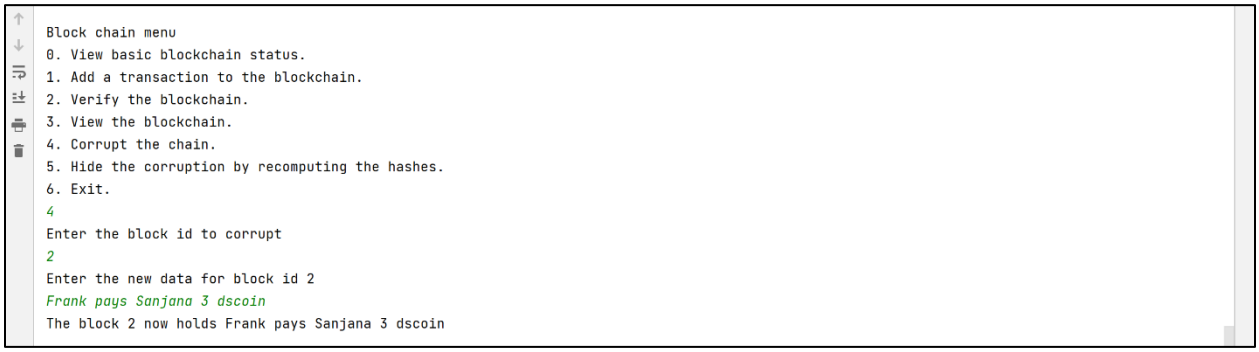
```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
3
{
    "ds_chain": [
        {
            "difficulty": 2,
            "time stamp": "2021-10-22 22:10:40.043",
            "index": 0,
            "Tx ": "Carol pays Sanjana 1000 dscoin",
            "nonce": 50,
            "PrevHash": ""
        },
```


```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:10:40.043",
      "index": 0,
      "Tx ": "Carol pays Sanjana 1000 dscoin",
      "nonce": 50,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:10:56.808",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 6968,
      "PrevHash": "0081DAA3E02B74BA8F3FF71BA9341013080DEC340770A000E6C8825DAA73AABA"
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:11:08.673",
      "index": 2,
      "Tx ": "Bob pays Sanjana 20 dscoin",
      "nonce": 1461,
      "PrevHash": "00077851B9144512AA0A89992BDA2AEE592C0E5D24F122764E65FC261842F921"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:11:17.888",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 28967,
      "PrevHash": "0007EC423DB793513168C5A8C28BE2C36FDC45199D26C8644A234E1C33E40203"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:11:30.001",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 405421,
      "PrevHash": "00009A6CABC347B076F75C86092A70B807B4E0AA0A7393A8F1E413D5BE92AB8C"
    },
```

```
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:11:58.826",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 120,
      "PrevHash": "000007E7AA548BD2B2A91D982186F9C56A4D04E4E18D6BEF6A68AE0856B754A0"
    }
  ],
  "chainHash": "00FFF16C2A4DD6C8E523402C5DEF61F5B5874947F3D5299CC94452D60CEF4C6F"
}
```





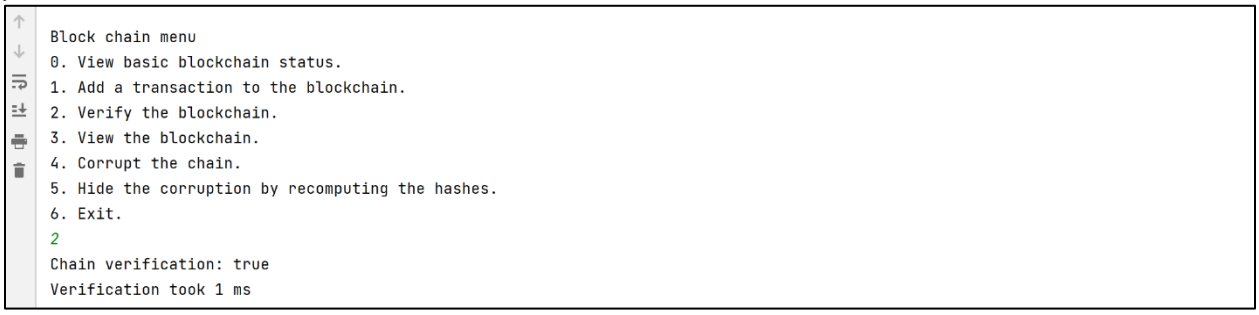
```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
2
Chain verification: false
..Improper hash on node 2. Does not begin with 000
Verification took 0 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
5
Repairing the entire chain
Repairing took 5674 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
3
{
    "ds_chain": [
        {
            "difficulty": 2,
            "time stamp": "2021-10-22 22:10:40.043",
            "index": 0,
            "Tx ": "Carol pays Sanjana 1000 dscoin",
            "nonce": 50,
            "PrevHash": ""
```


```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:10:40.043",
      "index": 0,
      "Tx ": "Carol pays Sanjana 1000 dscoin",
      "nonce": 50,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:10:56.808",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 6968,
      "PrevHash": "0081DAA3E02B74BA8F3FF71BA9341013080DEC340770A000E6C8825DAA73AABA"
    },
```

```
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:11:08.673",
      "index": 2,
      "Tx ": "Frank pays Sanjana 3 dscoin",
      "nonce": 1074,
      "PrevHash": "00077851B9144512AA0A89992BDA2AEE592C0E5D24F122764E65FC261842F921"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:11:17.888",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 3834,
      "PrevHash": "000674D1D4ED4CC6A5570594FE2D3F2A0130E495D8A4B71E52D98E08E8A90285"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:11:30.001",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 3443953,
      "PrevHash": "0000C446B5CCA0A7CF04A1B752143F622D3FC78B66C6C3F2549C24FD6E15C252"
    },
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:11:58.826",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 484,
      "PrevHash": "0000030A0A9F6088E68761E661219E40F0F364236EFFF089322BF7A4E8A668E5"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:13:11.892",
      "index": 6,
      "Tx ": "Edward pays Sanjana 34 dscoin",
      "nonce": 24482,
      "PrevHash": "00A92332D97C8D5054B9ADADE2449E1237C6897399E0EC64CFFEFAF0A799046E"
    }
  ],
  "chainHash": "0000CAF47CB51949A774DF89C576C38B19E51DCC43CC122F0B9D92DF91FF34B0"
}
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
0
Current size of chain: 7
Difficulty of most recent block: 4
Total difficulty for all blocks: 23
Approximate hashes per second on this machine: 840336
Expected total hashes required for the whole chain: 1188352.0
Nonce for most recent block: 24482
Chain hash: 0000CAF47CB51949A774DF89C576C38B19E51DCC43CC122F0B9D92DF91FF34B0
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
6
Exiting!

Process finished with exit code 0
```

Task 0 Block.java

```java
//         Name: Sanjana Rinke
//         Andrew ID: srinke
//         Email: srinke@andrew.cmu.edu
//         Project 3-Task 0
//         This is a Block class which contains all the instance variables for
the block and its getters/setters

package edu.cmu.andrew.srinke;

import org.json.JSONObject;

import java.io.UnsupportedEncodingException;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;

public class Block {
    MessageDigest md = null;
    int index;
    Timestamp timestamp;
    String data;
    String previousHash;
    BigInteger nonce;
    int difficulty;
    String hash;

    public Block(int index, Timestamp t, String data, int difficulty) {
        this.index = index;
```

```
        this.timestamp = t;
        this.data = data;
        this.previousHash = "";
        this.nonce = new BigInteger("0");
        this.difficulty = difficulty;
    }

    public String getHash() {
        return hash;
    }

    public void setHash(String hash) {
        this.hash = hash;
    }

    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;
    }

    public BigInteger getNonce() {
        return nonce;
    }

    public void setNonce(BigInteger nonce) {
        this.nonce = nonce;
    }

    public int getDifficulty() {
        return difficulty;
    }

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

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

```
    }

    public String getPreviousHash() {
        return previousHash;
    }

    //calculate SHA-256 hash for block
    public String calculateHash() {
        String hexaDecimalStr = null;
        try {
            md = MessageDigest.getInstance("SHA-256");
            String concatStr = index + "," + timestamp + "," + data + "," +
previousHash + "," + nonce + "," + difficulty;
            md.update(concatStr.getBytes("UTF-8"));
            byte[] digest = md.digest();
            hexaDecimalStr = bytesToHex(digest);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return hexaDecimalStr;
    }

    //calculate proof of work for a particular block
    // pseudocode reference taken from Lecture-1
    public void proofOfWork() {
        setNonce(new BigInteger("0"));
        setHash(calculateHash());
        while (!checkLeading0(getHash())) {
            setNonce(getNonce().add(new BigInteger("1")));
            setHash(calculateHash());
        }
    }

    //check if the no of '0' in the hex string is same as or more than the
difficulty
    private boolean checkLeading0(String hexStr) {
        int count = 0;
        for (char c : hexStr.toCharArray()) {
            if (c == '0')
                count++;
            else
                break;
        }
        if (count >= difficulty)
            return true;
        return false;
    }

    @Override
    public String toString() {
        JSONObject obj = new JSONObject();
        obj.put("index", index);
        obj.put("time stamp", timestamp.toString());
        obj.put("Tx ", data);
        obj.put("PrevHash", previousHash);
```

```
        obj.put("nonce", nonce);
        obj.put("difficulty", difficulty);
        return obj.toString();
    }

    //Lab 1-https://github.com/CMU-Heinz-95702/Lab1-InstallationAndRaft
    //converts the bytes to hexadecimal string
    public static String bytesToHex(byte[] bytes) {
        final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }

    public static void main(String[] args) {
        // write your code here
    }

}
```

Task 0 BlockChain.java

```
//        Name: Sanjana Rinke
//        Andrew ID: srinke
//        Email: srinke@andrew.cmu.edu
//        Project 3-Task 0
//        This is a Blockchain class which takes input from user to perform
various operations

package edu.cmu.andrew.srinke;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;


public class BlockChain {
    List<Block> blocks = null;
    String chainHash;
    int hashPerSec;
    //created a field to store the corrupt nodeID
    int corruptNodeId;
```

```
    MessageDigest md = null;

    //initialize blockchain
    public BlockChain() {
        corruptNodeId = -1;
        this.blocks = new ArrayList<>();
        this.chainHash = "";
        this.hashPerSec = 0;
        try {
            md = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
    }

    public int getHashesPerSecond() {
        return hashPerSec;
    }

    public Block getLatestBlock() {
        return blocks.get(getChainSize() - 1);
    }

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

    public int getCorruptNodeId() {
        return corruptNodeId;
    }

    public void setCorruptNodeId(int corruptNodeId) {
        this.corruptNodeId = corruptNodeId;
    }

    //adds a block at the end of the list
    public void addBlock(Block newBlock) {
        //if chain has many blocks, sets previous hash as the hash of the
previous block
        if ((getChainSize() + 1) > 1) {
            newBlock.setPreviousHash(blocks.get(newBlock.getIndex() -
1).getHash());
        } else
            //if the chain has 1 block, sets previous hash as "".
            newBlock.setPreviousHash("");
        newBlock.proofOfWork();
        //sets chainHash has the hash of the new block
        chainHash = newBlock.getHash();
        blocks.add(newBlock);
    }

    //computes hashes per second by hashing a constant string '00000000'
    public void computeHashesPerSecond() {
        String input = "00000000";
        Long startTime = getTime().getTime();
        for (int i = 0; i < 100000; i++) {
            //calculate hash for the input
```

```
            calculateHash(input);
        }
        Long endTime = getTime().getTime();
        Long timeDiff = endTime - startTime;
        //find the time diff in seconds
        float diffInSecs = (float) timeDiff / 1000;
        //divide million by time taken in secs
        hashPerSec = (int) (1000000 / diffInSecs);
    }

    //calculate SHA-256 hash for the passed input
    public String calculateHash(String input) {
        byte[] digest = null;
        try {
            md.update(input.getBytes("UTF-8"));
            digest = md.digest();

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return digest.toString();
    }

    //get the block at ith position
    public Block getBlock(int i) {
        return blocks.get(i);
    }

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

    //get the combined difficulty of all blocks
    public int getTotalDifficulty() {
        int totalDiff = 0;
        for (int i = 0; i < getChainSize(); i++) {
            totalDiff += getBlock(i).getDifficulty();
        }
        return totalDiff;
    }

    //get total expected hashes
    public double getTotalExpectedHashes() {
        double totHash = 0;
        for (int i = 0; i < getChainSize(); i++) {
            totHash += Math.pow(16, getBlock(i).difficulty);
        }
        return totHash;
    }

    //check if entire chain is valid or not
    public boolean isChainValid() {
        Block b = null;
        boolean flag = false;
        try {
//            md = MessageDigest.getInstance("SHA-256");
```

```
            //will execute if the chain has only 1 block
            if (getChainSize() == 1) {
                b = getBlock(0);
                //check for valid block and if the hash of the block is same
as chainhash
                if (b.getHash().equalsIgnoreCase(chainHash) &&
checkValidBlock(b))
                    flag = true;
                else {
                    flag = false;
                    //if the block is corrupt, set the corrupt node ID
                    setCorruptNodeId(b.getIndex());
                }
            }
            //will execute if the chain has more than 1 block
            else {
                for (int i = 0; i < getChainSize(); i++) {
                    if (i != getChainSize() - 1) {
                        b = getBlock(i);
                        if (b.getHash().equalsIgnoreCase(blocks.get(i +
1).getPreviousHash()) && checkValidBlock(b))
                            flag = true;
                        else {
                            flag = false;
                            setCorruptNodeId(b.getIndex());
                            break;
                        }
                    } else {
                        b = getBlock(i);
                        if (b.getHash().equalsIgnoreCase(chainHash) &&
checkValidBlock(b))
                            flag = true;
                        else {
                            flag = false;
                            setCorruptNodeId(b.getIndex());
                            break;
                        }
                    }

                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return flag;
    }
    //checking if the hash of the block as same or more no of of leading 0 as
difficulty
    // Eg: if difficulty=2 then valid hash='00...'
    private boolean checkProofOfWork(String hexaDecimalStr, int difficulty) {
        int count = 0;
        for (char c : hexaDecimalStr.toCharArray()) {
            if (c == '0')
                count++;
            else
                break;
        }
```

```
        if (count >= difficulty) {
            return true;
        }
        return false;
    }

    //repairs the invalid chain
    public void repairChain() {
        for (int i = 0; i < getChainSize(); i++) {
            if (!checkValidBlock(getBlock(i))) {
                getBlock(i).proofOfWork();
                reAssignPrevHash(getBlock(i));
            }
        }
    }

    //re-assigns updated previous hash for all the blocks after the corrupt
block
    private void reAssignPrevHash(Block block) {
        for (int i = block.getIndex(); i < getChainSize(); i++) {
            if (i + 1 < getChainSize())
                blocks.get(i + 1).setPreviousHash(getBlock(i).getHash());
        }
        chainHash = blocks.get(getChainSize() - 1).getHash();
    }

    //check if a block is valid by verifying its proof of work
    public boolean checkValidBlock(Block b) {
        String message = b.getIndex() + "," + b.getTimestamp() + "," +
b.getData() + "," + b.getPreviousHash() + "," + b.getNonce() + "," +
b.getDifficulty();
        try {
            md.update(message.getBytes("UTF-8"));
            byte[] digest = md.digest();
            String hexaDecimalStr = bytesToHex(digest);
            if (checkProofOfWork(hexaDecimalStr, b.getDifficulty())) {
                return true;
            }
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return false;
    }

    @Override
    public String toString() {
        JSONObject obj = new JSONObject();
        JSONArray jsonArray = new JSONArray(blocks.toString());
        obj.put("ds_chain", jsonArray);
        obj.put("chainHash", chainHash);
        return obj.toString();
    }

    //Lab 1-https://github.com/CMU-Heinz-95702/Lab1-InstallationAndRaft
    //converts the bytes to hexadecimal string
    public static String bytesToHex(byte[] bytes) {
        final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
```

```
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }

    public static void main(String[] args) {
        //While adding blocks of increasing difficulty, the time taken to add
increases.
        // For example- time taken to add block of difficulty:2 is 6ms and
time taken for difficulty: 5 is 211ms
        //While the chain becomes longer the the block-chain verification
takes similar amount of time
        //for chain size- 2: 0ms ; for chain size-8 1ms
        //While the chain becomes longer with increasing difficulty, the
repair takes longer times.
        //Repair time for 8 blocks with difficulty- 23: 1342ms
        // Repair time for 2 blocks with difficulty-5: 1ms
        BlockChain blockChain = new BlockChain();
        //compute hashPerSecond on startup
        blockChain.computeHashesPerSecond();
        Timestamp startTime;
        Timestamp endTime;
        //adding the genesis block
        Block genesisBlock = new Block(0, blockChain.getTime(), "Genesis",
2);
        blockChain.addBlock(genesisBlock);
        BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));
        try {
            int ch = -1;
            do {
                System.out.println("\nBlock chain menu");
                System.out.println("0. View basic blockchain status.");
                System.out.println("1. Add a transaction to the
blockchain.");
                System.out.println("2. Verify the blockchain.");
                System.out.println("3. View the blockchain.");
                System.out.println("4. Corrupt the chain.");
                System.out.println("5. Hide the corruption by recomputing the
hashes.");
                System.out.println("6. Exit.");
                ch = Integer.parseInt(typed.readLine());
                switch (ch) {
                    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 most recent block: " +
blockChain.getLatestBlock().getNonce());
                        System.out.println("Chain hash: " +
blockChain.chainHash);
                        break;
                    case 1:
                        System.out.println("Enter Difficulty>0");
                        int diff = Integer.parseInt(typed.readLine());
                        System.out.println("Enter the transaction");
                        String transaction = typed.readLine();
                        Block b = new Block(blockChain.getChainSize(),
blockChain.getTime(), transaction, diff);
                        startTime = blockChain.getTime();
                        blockChain.addBlock(b);
                        endTime = blockChain.getTime();
                        System.out.println("Time taken to add this block::" +
(endTime.getTime() - startTime.getTime()) + " ms");
                        break;
                    case 2:
                        startTime = blockChain.getTime();
                        if (blockChain.isChainValid() == true) {
                            System.out.print("Chain verification: true");
                        } else {
                            System.out.println("Chain verification: false");
                            System.out.print("..Improper hash on node " +
blockChain.getCorruptNodeId() + ". Does not begin with ");
                            for (int i = 0; i <
blockChain.getBlock(blockChain.getCorruptNodeId()).getDifficulty(); i++) {
                                System.out.print("0");
                            }
                        }
                        endTime = blockChain.getTime();
                        System.out.println("\nVerification took " +
(endTime.getTime() - startTime.getTime()) + " ms");
                        break;
                    case 3:
                        JSONObject json = new
JSONObject(blockChain.toString());
                        System.out.println(json.toString(4));
                        break;
                    case 4:
                        System.out.println("Enter the block id to corrupt");
                        int id = Integer.parseInt(typed.readLine());
                        System.out.println("Enter the new data for block id "
+ id);
                        String data = typed.readLine();
                        blockChain.blocks.get(id).setData(data);
                        System.out.println("The block "+id+" now holds " +
blockChain.blocks.get(id).getData());
                        break;
                    case 5:
                        startTime = blockChain.getTime();
                        System.out.println("Repairing the entire chain");
                        blockChain.repairChain();
                        endTime = blockChain.getTime();
                        System.out.println("Repairing took " +
```

```
(endTime.getTime() - startTime.getTime()) + " ms");
                        break;
                    case 6:
                        System.out.println("Exiting!");
                        System.exit(0);
                        break;
                }


            } while (ch != 6);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Task 1 Execution

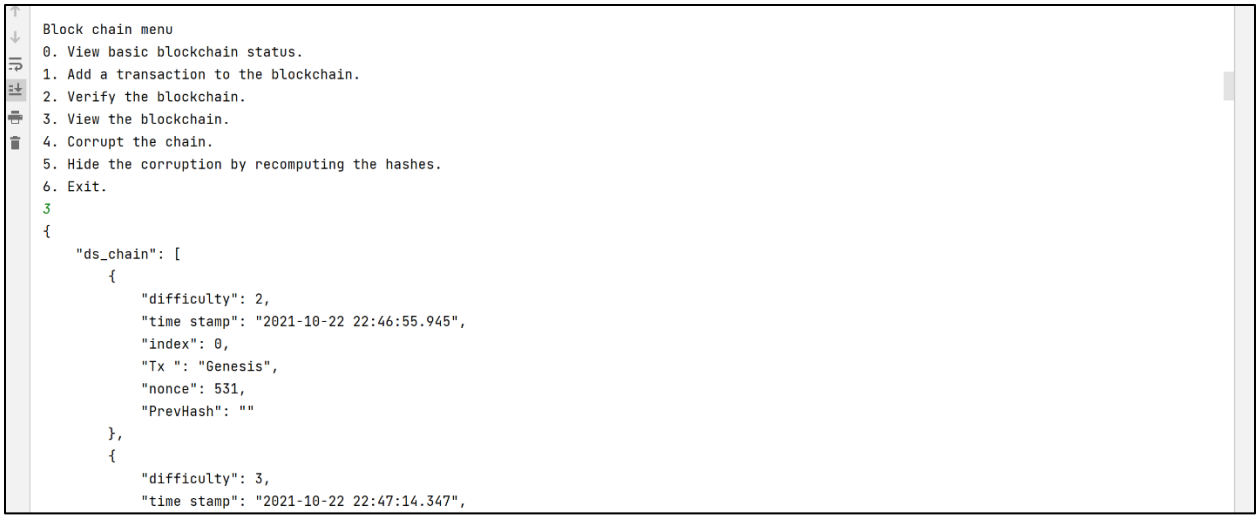
```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
0
Current size of chain: 1
Difficulty of most recent block: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 13888889
Expected total hashes required for the whole chain: 256
Nonce for most recent block: 531
Chain hash: 008E79B15F3F76421FEEDC851EDFDC0A639DF846E21F9D8E49F4F577557BBFFB
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
3
Enter the transaction
Sanjana pays Bob 100 dscoin
Time taken to add this block::27 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
3
Enter the transaction
Bob pays Sanjana 20 dscoin
Time taken to add this block::14 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
4
Enter the transaction
Sanjana pays Carol 23 dscoin
Time taken to add this block::158 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
5
Enter the transaction
Donna pays Sanjana 34 dscoin
Time taken to add this block::1333 ms
```



```
Block chain menu
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 recomputing the hashes.
6. Exit.
3
{
    "ds_chain": [
        {
            "difficulty": 2,
            "time stamp": "2021-10-22 22:46:55.945",
            "index": 0,
            "Tx ": "Genesis",
            "nonce": 531,
            "PrevHash": ""
        },
        {
            "difficulty": 3,
            "time stamp": "2021-10-22 22:47:14.347",
```


```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:46:55.945",
      "index": 0,
      "Tx ": "Genesis",
      "nonce": 531,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:14.347",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 5584,
      "PrevHash": "008E79B15F3F76421FEEDC851EDFDC0A639DF846E21F9D8E49F4F577557BBFFB"
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:20.791",
      "index": 2,
      "Tx ": "Bob pays Sanjana 20 dscoin",
      "nonce": 3509,
      "PrevHash": "0007A1F9A3D551B7E7B9147862E438FB2FFB69E1FCEB2675D5B71B3A6C81C764"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:48:04.858",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 104878,
      "PrevHash": "00043D5F7051F9A6A27EFF9B67428D0D9A2A370A77DD2049A5B5D97BA5C8DE46"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:48:14.398",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 2554216,
```

```
        "PrevHash": "0000DC3793292F75076A36A99906963F7F62898398297B524B4A589397534E12"
      }
    ],
    "chainHash": "00000C4D743353BF9723CD9206C39437861AC471D6B7D1EBA36C9676D2DEC55B"
}
```

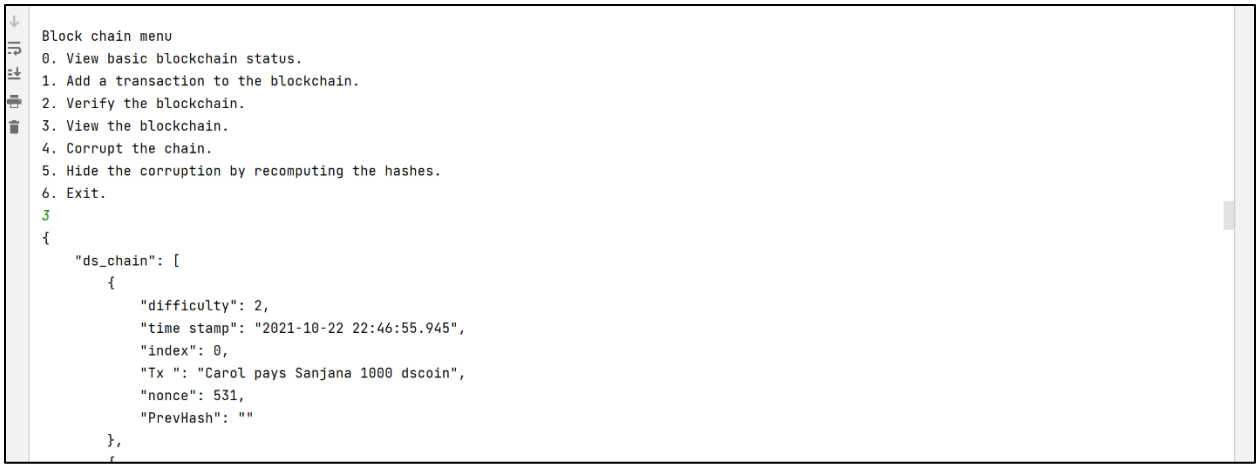

```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
2
Enter the transaction
Carol pays Donna 1 dscoin
Time taken to add this block::0 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
3
{
    "ds_chain": [
        {
            "difficulty": 2,
            "time stamp": "2021-10-22 22:46:55.945",
            "index": 0,
            "Tx ": "Genesis",
            "nonce": 531,
            "PrevHash": ""
        },
        {
```


```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:46:55.945",
      "index": 0,
      "Tx ": "Genesis",
      "nonce": 531,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:14.347",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 5584,
      "PrevHash": "008E79B15F3F76421FEEDC851EDFDC0A639DF846E21F9D8E49F4F577557BBFFB"
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:20.791",
      "index": 2,
```

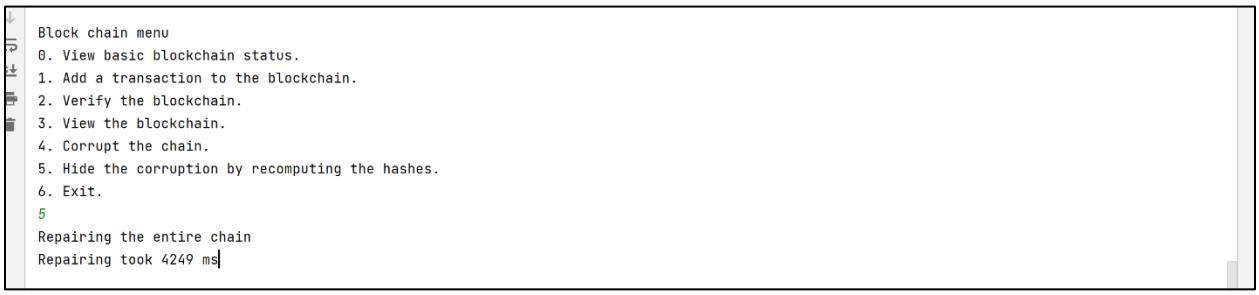
```
      "Tx ": "Bob pays Sanjana 20 dscoin",
      "nonce": 3509,
      "PrevHash": "0007A1F9A3D551B7E7B9147862E438FB2FFB69E1FCEB2675D5B71B3A6C81C764"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:48:04.858",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 104878,
      "PrevHash": "00043D5F7051F9A6A27EFF9B67428D0D9A2A370A77DD2049A5B5D97BA5C8DE46"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:48:14.398",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 2554216,
      "PrevHash": "0000DC3793292F75076A36A99906963F7F62898398297B524B4A589397534E12"
    },
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:48:44.435",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 219,
      "PrevHash": "00000C4D743353BF9723CD9206C39437861AC471D6B7D1EBA36C9676D2DEC55B"
    }
  ],
  "chainHash": "00A31D7D4D70FAB9C2E52D7A4E3C82BA64DDB88B89CF0985C15A98AD3FB34E1F"
}
```


```
Block chain menu
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 recomputing the hashes.
6. Exit.
4
Enter the id to corrupt
0
Enter the data
Carol pays Sanjana 1000 dscoin
The block 0 now holds Carol pays Sanjana 1000 dscoin
```

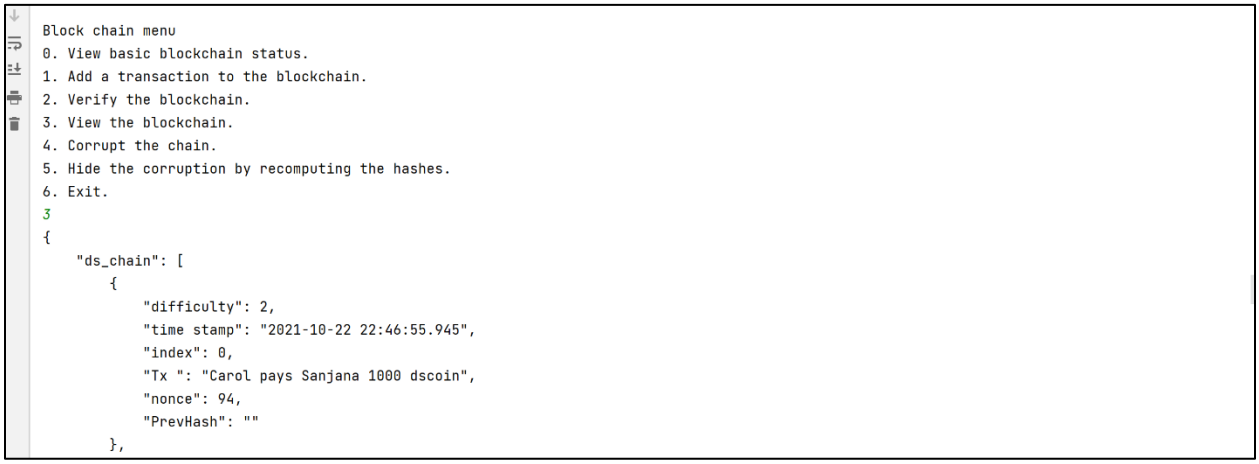



```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:46:55.945",
      "index": 0,
      "Tx ": "Carol pays Sanjana 1000 dscoin",
      "nonce": 531,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:14.347",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 5584,
      "PrevHash": "008E79B15F3F76421FEEDC851EDFDC0A639DF846E21F9D8E49F4F577557BBFFB"
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:20.791",
      "index": 2,
      "Tx ": "Bob pays Sanjana 20 dscoin",
      "nonce": 3509,
      "PrevHash": "0007A1F9A3D551B7E7B9147862E438FB2FFB69E1FCEB2675D5B71B3A6C81C764"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:48:04.858",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 104878,
      "PrevHash": "00043D5F7051F9A6A27EFF9B67428D0D9A2A370A77DD2049A5B5D97BA5C8DE46"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:48:14.398",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 2554216,
      "PrevHash": "0000DC3793292F75076A36A99906963F7F62898398297B524B4A589397534E12"
    },
```

```
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:48:44.435",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 219,
      "PrevHash": "00000C4D743353BF9723CD9206C39437861AC471D6B7D1EBA36C9676D2DEC55B"
    }
  ],
  "chainHash": "00A31D7D4D70FAB9C2E52D7A4E3C82BA64DDB88B89CF0985C15A98AD3FB34E1F"
}
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
5
Repairing the entire chain
Repairing took 4249 ms
```



```
Block chain menu
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 recomputing the hashes.
6. Exit.
3
{
    "ds_chain": [
        {
            "difficulty": 2,
            "time stamp": "2021-10-22 22:46:55.945",
            "index": 0,
            "Tx ": "Carol pays Sanjana 1000 dscoin",
            "nonce": 94,
            "PrevHash": ""
        },
```


```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:46:55.945",
      "index": 0,
      "Tx ": "Carol pays Sanjana 1000 dscoin",
      "nonce": 94,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:14.347",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 4925,
      "PrevHash": "003DC44850C1070017D95492C4B74421AA34AA6B760453C0199BFCA7421F4C2E"
    },
    {
```

```
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:20.791",
      "index": 2,
      "Tx ": "Bob pays Sanjana 20 dscoin",
      "nonce": 9329,
      "PrevHash": "000565953F6393AC534474D3D80A2C7458F076D32F6707C917230AA994988CE2"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:48:04.858",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 161295,
      "PrevHash": "00003C0764B83CBDDE846DD19F1CE3100C92177E64D422AB624C6EAB0D5A96F0"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:48:14.398",
      "index": 4,
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 977599,
      "PrevHash": "0000FA4E4A242899736322E54161272EF5E1FE9D5CE0B48B9B73948C5EDD1A19"
    },
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:48:44.435",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 53,
      "PrevHash": "0000067DE8F95E2E1524E0B6A5C1ADB35570CEFC35FF6722DD82BC9801548C46"
    }
  ],
  "chainHash": "00C51192BE452ACB9F418CF6EC982583D5BE1FE6DA2FE374CF309983D31E5863"
}
```

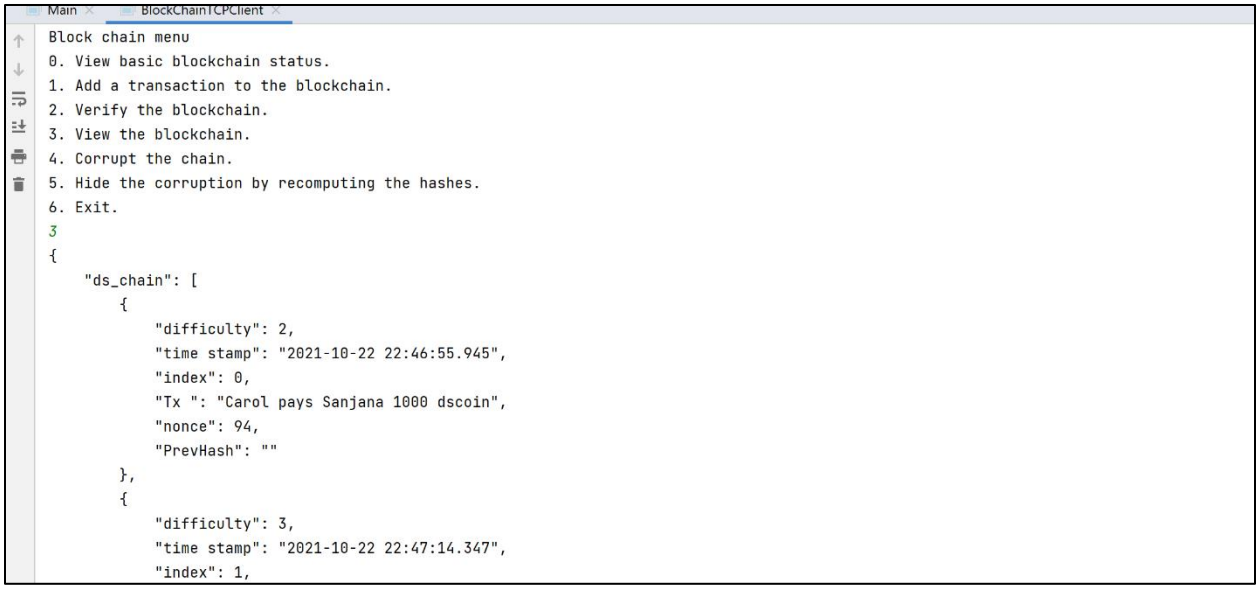

```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
1
Enter Difficulty>0
4
Enter the transaction
Edward pays Sanjana 34 dscoin
Time taken to add this block::20 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
2
Chain verification: true
Verification took 1 ms
```

```
Block chain menu
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 recomputing the hashes.
6. Exit.
4
Enter the id to corrupt
2
Enter the data
Frank pays Sanjana 3 dscoin
The block 2 now holds Frank pays Sanjana 3 dscoin
```

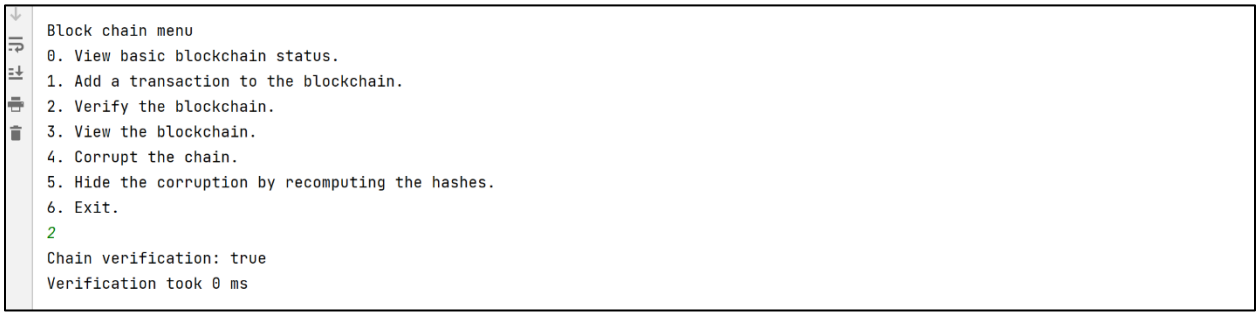
```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
2
Chain verification: false
..Improper hash on node 2. Does not begin with 000
Verification took 0 ms
```

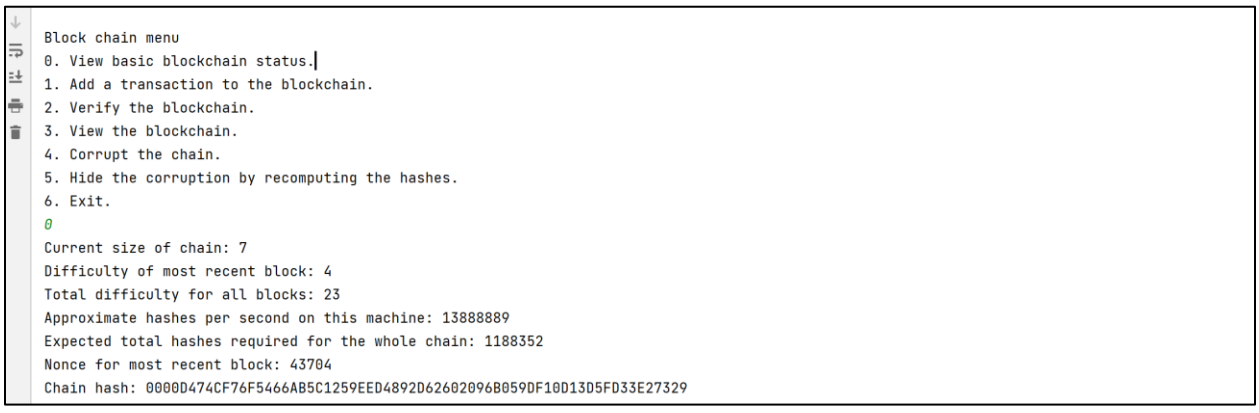
```
Block chain menu
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by recomputing the hashes.
6. Exit.
5
Repairing the entire chain
Repairing took 259 ms
```




```
{
  "ds_chain": [
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:46:55.945",
      "index": 0,
      "Tx ": "Carol pays Sanjana 1000 dscoin",
      "nonce": 94,
      "PrevHash": ""
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:14.347",
      "index": 1,
      "Tx ": "Sanjana pays Bob 100 dscoin",
      "nonce": 4925,
      "PrevHash": "003DC44850C1070017D95492C4B74421AA34AA6B760453C0199BFCA7421F4C2E"
    },
    {
      "difficulty": 3,
      "time stamp": "2021-10-22 22:47:20.791",
      "index": 2,
      "Tx ": "Frank pays Sanjana 3 dscoin",
      "nonce": 731,
      "PrevHash": "000565953F6393AC534474D3D80A2C7458F076D32F6707C917230AA994988CE2"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:48:04.858",
      "index": 3,
      "Tx ": "Sanjana pays Carol 23 dscoin",
      "nonce": 33101,
      "PrevHash": "000D43B838164EA71EC8B9FBBA13B6CD3F70EE06463AD3FCFA2636127CD9BF44"
    },
    {
      "difficulty": 5,
      "time stamp": "2021-10-22 22:48:14.398",
      "index": 4,
```

```
      "Tx ": "Donna pays Sanjana 34 dscoin",
      "nonce": 397687,
      "PrevHash": "0000107A5C7F19F2A0C5CCDBBF61EC0B57D2269D21C9510EBF3540D60BCA5427"
    },
    {
      "difficulty": 2,
      "time stamp": "2021-10-22 22:48:44.435",
      "index": 5,
      "Tx ": "Carol pays Donna 1 dscoin",
      "nonce": 133,
      "PrevHash": "00000C938A7442E7D1BD726E811C390962D082E63752DE91EE6AE3DE660297AA"
    },
    {
      "difficulty": 4,
      "time stamp": "2021-10-22 22:49:39.866",
      "index": 6,
      "Tx ": "Edward pays Sanjana 34 dscoin",
      "nonce": 43704,
      "PrevHash": "006AF99A86044E683EA4B857E5C17C09A4ED1F6C12B9E0E423647796B3DB6FC1"
    }
  ],
  "chainHash": "0000D474CF76F5466AB5C1259EED4892D62602096B059DF10D13D5FD33E27329"
}
```



```
Block chain menu
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 recomputing the hashes.
6. Exit.
6
Exiting!

Process finished with exit code 0
```

Task 1 Client Source Code

```
//         Name: Sanjana Rinke
//         Andrew ID: srinke
//         Email: srinke@andrew.cmu.edu
//         Project 3-Task 1
//         This is a TCP Client which takes input from user and sends data to
server for block-chain operations
//         All the execution times are calculated on server and sent back to
client
//         External jar file is used for JSON.
//         It was added in Intellij as File->Project Structure->Libraries->
Add-> jar_files.jar

package edu.cmu.andrew.srinke;


import org.json.JSONObject;

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

public class BlockChainTCPClient {

    //socket
    Socket clientSocket = null;
```

```
    public static void main(String[] args) {
        BlockChainTCPClient blockChainTCPClient = new BlockChainTCPClient();
        //initialize sockets
        blockChainTCPClient.init();
        try {
            BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));
            JSONObject request = new JSONObject();
            JSONObject response;
            int ch = -1;
            do {
                System.out.println("\nBlock chain menu");
                System.out.println("0. View basic blockchain status.");
                System.out.println("1. Add a transaction to the
blockchain.");
                System.out.println("2. Verify the blockchain.");
                System.out.println("3. View the blockchain.");
                System.out.println("4. Corrupt the chain.");
                System.out.println("5. Hide the corruption by recomputing the
hashes.");
                System.out.println("6. Exit.");
                ch = Integer.parseInt(typed.readLine());
                switch (ch) {
                    case 0:
                        //create request object
                        request.put("choice", 0);
                        //send data to server
                        blockChainTCPClient.send(request.toString());
                        //receive data from server and parse it
                        response = blockChainTCPClient.receiveServerData();
                        System.out.println("Current size of chain: " +
response.get("chainSize"));
                        System.out.println("Difficulty of most recent block:
" + response.get("recentDiff"));
                        System.out.println("Total difficulty for all blocks:
" + response.get("totDiff"));
                        System.out.println("Approximate hashes per second on
this machine: " + response.get("approxHash"));
                        System.out.println("Expected total hashes required
for the whole chain: " + response.get("expectedHash"));
                        System.out.println("Nonce for most recent block: " +
response.get("nonce"));
                        System.out.println("Chain hash: " +
response.get("chainHash"));
                        break;
                    case 1:
                        //create request object
                        request.put("choice", 1);
                        System.out.println("Enter Difficulty>0");
                        int diff = Integer.parseInt(typed.readLine());
                        request.put("difficulty", diff);
                        System.out.println("Enter the transaction");
                        String transaction = typed.readLine();
                        request.put("data", transaction);
                        //send data to server
                        blockChainTCPClient.send(request.toString());
```

```
                        //receive data from server and parse it
                        response = blockChainTCPClient.receiveServerData();
                        System.out.println("Time taken to add this block::" +
response.get("addTime") + " ms");
                        break;
                    case 2:
                        //create request object
                        request.put("choice", 2);
                        //send data to server
                        blockChainTCPClient.send(request.toString());
                        //receive data from server and parse it
                        response = blockChainTCPClient.receiveServerData();
                        if
(response.get("result").toString().equalsIgnoreCase("true")) {
                            System.out.print("Chain verification: true");
                        } else {
                            System.out.println("Chain verification: false");
                            System.out.print("..Improper hash on node " +
response.get("corruptNode") + ". Does not begin with ");
                            for (int i = 0; i <
Integer.parseInt(response.get("corruptNodeDiff").toString()); i++) {
                                System.out.print("0");
                            }
                        }
                        System.out.println("\nVerification took " +
response.get("verificationTime") + " ms");
                        break;
                    case 3:
                        //create request object
                        request.put("choice", 3);
                        //send data to server
                        blockChainTCPClient.send(request.toString());
                        //receive data from server and parse it
                        response = blockChainTCPClient.receiveServerData();
                        JSONObject json=new
JSONObject(response.get("blockChain").toString());
                        System.out.println(json.toString(4));
                        break;
                    case 4:
                        //create request object
                        request.put("choice", 4);
                        System.out.println("Enter the id to corrupt");
                        int id = Integer.parseInt(typed.readLine());
                        request.put("corruptID", id);
                        System.out.println("Enter the data");
                        String newData = typed.readLine();
                        request.put("newData", newData);
                        //send data to server
                        blockChainTCPClient.send(request.toString());
                        //receive data from server and parse it
                        response = blockChainTCPClient.receiveServerData();
                        System.out.println("The block "+id+" now holds " +
response.get("newData"));
                        break;
                    case 5:
                        //create request object
                        request.put("choice", 5);
```

```
                        System.out.println("Repairing the entire chain");
                        //send data to server
                        blockChainTCPClient.send(request.toString());
                        //receive data from server and parse it
                        response = blockChainTCPClient.receiveServerData();
                        System.out.println("Repairing took " +
(response.get("repairTime")) + " ms");
                        break;
                    case 6:
                        System.out.println("Exiting!");
                        System.exit(0);
                        break;
                }


            } while (ch != 6);

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            blockChainTCPClient.close();
        }
    }

    //receive JSON data from server
    public JSONObject receiveServerData() {
        BufferedReader in = null;
        JSONObject response = null;
        try {
            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            String data = in.readLine();
            response = new JSONObject(data);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return response;
    }

    //initialize socket
    private void init() {
        int serverPort = 7777;
        try {
            clientSocket = new Socket("localhost", serverPort);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //send data to server
    private void send(String message) {

        try {
            PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            out.println(message);
            out.flush();
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //close socket
    private void close() {
        try {
            if (clientSocket != null) {
                clientSocket.close();
            }
        } catch (IOException e) {
            // ignore exception on close
        }
    }
}
```

Task 1 Server Source Code

```
//        Name: Sanjana Rinke
//        Andrew ID: srinke
//        Email: srinke@andrew.cmu.edu
//        Project 3-Task 1
//        This is a TCP Server which takes data from client and performs
block-chain operations
//        All the execution times are calculated on server and sent back to
client
//        External jar file is used for JSON.
//        It was added in Intellij as File->Project Structure->Libraries->
Add-> jar_files.jar

package edu.cmu.andrew.srinke;


import org.json.JSONObject;

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;

public class BlockChainTCPServer {

    Socket socket = null;
    ServerSocket listenSocket = null;
    BlockChain blockChain = new BlockChain();

    public static void main(String[] args) {
        // write your code here
        System.out.println("Server started!");
        BlockChainTCPServer blockChainTCPServer = new BlockChainTCPServer();
        //initialize sockets
```

```
        blockChainTCPServer.init();
        Scanner in;
        //add genesis block
        Block genesisBlock = new Block(0,
blockChainTCPServer.blockChain.getTime(), "Genesis", 2);
        blockChainTCPServer.blockChain.addBlock(genesisBlock);
        blockChainTCPServer.blockChain.computeHashesPerSecond();
        try {
            while (true) {
                //accept connection
                blockChainTCPServer.socket =
blockChainTCPServer.listenSocket.accept();
                //get input from client
                in = new
Scanner(blockChainTCPServer.socket.getInputStream());
                while (in.hasNextLine()) {
                    String data = in.nextLine();
                    //send data to getClientData to process
                    blockChainTCPServer.getClientData(data);
                }
                //close socket
                blockChainTCPServer.socket.close();
            }
            // Handle exceptions
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());
//            // If quitting (typically by you sending quit signal) clean up
sockets
        } finally {
            try {
                if (blockChainTCPServer.socket != null) {
                    blockChainTCPServer.socket.close();
                }
            } catch (IOException e) {
                // ignore exception on close
            }
        }
    }

    //process the JSON data coming from client
    private void getClientData(String incomingMessage) {
        try {
            //parse incoming json request
            JSONObject request = new JSONObject(incomingMessage);
            int operation =
Integer.parseInt(request.get("choice").toString());
            Timestamp startTime;
            Timestamp endTime;
            JSONObject response = new JSONObject();
            switch (operation) {
                case 0:
                    //create a response JSON object
                    System.out.println("Sending basic blockchain status");
                    response.put("chainSize", blockChain.getChainSize());
                    response.put("recentDiff",
blockChain.getLatestBlock().getDifficulty());
                    response.put("totDiff", blockChain.getTotalDifficulty());
```

```
                    response.put("approxHash",
blockChain.getHashesPerSecond());
                    response.put("expectedHash",
blockChain.getTotalExpectedHashes());
                    response.put("nonce",
blockChain.getLatestBlock().getNonce());
                    response.put("chainHash", blockChain.chainHash);
                    break;
                case 1:
                    //parse incoming request
                    System.out.println("Adding block");
                    int diff =
Integer.parseInt(request.get("difficulty").toString());
                    String transaction = request.get("data").toString();
                    Block b = new Block(blockChain.getChainSize(),
blockChain.getTime(), transaction, diff);
                    startTime = blockChain.getTime();
                    blockChain.addBlock(b);
                    endTime = blockChain.getTime();
                    //create a response JSON object
                    response.put("addTime", (endTime.getTime() -
startTime.getTime()));
                    break;
                case 2:
                    System.out.println("Verifying blockchain");
                    startTime = blockChain.getTime();
                    if (blockChain.isChainValid() == true) {
                        //create a response JSON object
                        response.put("result", true);
                    } else {
                        //create a response JSON object
                        response.put("result", false);
                        response.put("corruptNode",
blockChain.getCorruptNodeId());
                        response.put("corruptNodeDiff",
blockChain.getBlock(blockChain.getCorruptNodeId()).getDifficulty());
                    }
                    endTime = blockChain.getTime();
                    response.put("verificationTime", (endTime.getTime() -
startTime.getTime()));
                    break;
                case 3:
                    System.out.println("Viewing blockchain");
                    response.put("blockChain", blockChain);
                    break;
                case 4:
                    System.out.println("Corrupting blockchain");
                    //parse incoming request
                    int id =
Integer.parseInt(request.get("corruptID").toString());
                    String data = request.get("newData").toString();
                    blockChain.blocks.get(id).setData(data);
                    //create a response JSON object
                    response.put("newData",
blockChain.blocks.get(id).getData());
                    break;
                case 5:
```

```
                    System.out.println("Repairing blockchain");
                    startTime = blockChain.getTime();
                    blockChain.repairChain();
                    endTime = blockChain.getTime();
                    //create a response JSON object
                    response.put("repairTime", (endTime.getTime() -
startTime.getTime()));
                    break;
            }
            //send data to client
            send(response.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //send data back to client
    private void send(String message) {
        try {
            PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())));
            out.println(message);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //initialize socket
    private void init() {
        int serverPort = 7777;
        try {
            listenSocket = new ServerSocket(serverPort);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Task 1 Block Source Code

```
//        Name: Sanjana Rinke
//        Andrew ID: srinke
//        Email: srinke@andrew.cmu.edu
//        Project 3-Task 1
//        This is a Block class which contains all the instance variables for
the block and its getters/setters

package edu.cmu.andrew.srinke;

import org.json.JSONObject;

import java.io.UnsupportedEncodingException;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

```java
import java.sql.Timestamp;

public class Block {
    MessageDigest md = null;
    int index;
    Timestamp timestamp;
    String data;
    String previousHash;
    BigInteger nonce;
    int difficulty;
    String hash;

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

    public String getHash() {
        return hash;
    }

    public void setHash(String hash) {
        this.hash = hash;
    }

    public int getIndex() {
        return index;
    }

    public Timestamp getTimestamp() {
        return timestamp;
    }

    public String getData() {
        return data;
    }

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

    public BigInteger getNonce() {
        return nonce;
    }

    public void setNonce(BigInteger nonce) {
        this.nonce = nonce;
    }

    public int getDifficulty() {
        return difficulty;
    }
```

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

    public String getPreviousHash() {
        return previousHash;
    }

    //calculate SHA-256 hash for block
    public String calculateHash() {
        String hexaDecimalStr = null;
        try {
            md = MessageDigest.getInstance("SHA-256");
            String concatStr = index + "," + timestamp + "," + data + "," +
previousHash + "," + nonce + "," + difficulty;
            md.update(concatStr.getBytes("UTF-8"));
            byte[] digest = md.digest();
            hexaDecimalStr = bytesToHex(digest);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return hexaDecimalStr;
    }

    //calculate proof of work for a particular block
    // pseudocode reference taken from Lecture-1
    public void proofOfWork() {
        setNonce(new BigInteger("0"));
        setHash(calculateHash());
        while (!checkLeading0(getHash())) {
            setNonce(getNonce().add(new BigInteger("1")));
            setHash(calculateHash());
        }
    }

    //check if the no of '0' in the hex string is same as or more than
difficulty
    private boolean checkLeading0(String hexStr) {
        int count = 0;
        for (char c : hexStr.toCharArray()) {
            if (c == '0')
                count++;
            else
                break;
        }
        if (count >= difficulty)
            return true;
        return false;
    }

    @Override
    public String toString() {
        JSONObject obj = new JSONObject();
        obj.put("index", index);
        obj.put("time stamp", timestamp.toString());
```

```
        obj.put("Tx ", data);
        obj.put("PrevHash", previousHash);
        obj.put("nonce", nonce);
        obj.put("difficulty", difficulty);
        return obj.toString();
    }

    //Lab 1-https://github.com/CMU-Heinz-95702/Lab1-InstallationAndRaft
    //converts the bytes to hexadecimal string
    public static String bytesToHex(byte[] bytes) {
        final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }

}
```

Task 1 BlockChain Source Code

```
//        Name: Sanjana Rinke
//        Andrew ID: srinke
//        Email: srinke@andrew.cmu.edu
//        Project 3-Task 1
//        This is a Blockchain class which takes input from user to perform
various operations

package edu.cmu.andrew.srinke;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

public class BlockChain {
    List<Block> blocks = null;
    String chainHash;
    int hashPerSec;
    //created a field to store the corrupt nodeID
    int corruptNodeId;
    MessageDigest md = null;

    //initialize blockchain
    public BlockChain() {
        corruptNodeId = -1;
        this.blocks = new ArrayList<>();
```

```
        this.chainHash = "";
        this.hashPerSec = 0;
        try {
            md = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
    }

    public int getHashesPerSecond() {
        return hashPerSec;
    }

    public Block getLatestBlock() {
        return blocks.get(getChainSize() - 1);
    }

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

    public int getCorruptNodeId() {
        return corruptNodeId;
    }

    public void setCorruptNodeId(int corruptNodeId) {
        this.corruptNodeId = corruptNodeId;
    }

    //adds a block at the end of the list
    public void addBlock(Block newBlock) {
        //if chain has many blocks, sets previous hash as the hash of the
previous block
        if ((getChainSize() + 1) > 1) {
            newBlock.setPreviousHash(blocks.get(newBlock.getIndex() -
1).getHash());
        } else
            //if the chain has 1 block, sets previous hash as "".
            newBlock.setPreviousHash("");
        newBlock.proofOfWork();
        //sets chainHash has the hash of the new block
        chainHash = newBlock.getHash();
        blocks.add(newBlock);
    }

    //computes hashes per second by hashing a constant string '00000000'
    public void computeHashesPerSecond() {
        String input = "00000000";
        Long startTime = getTime().getTime();
        for (int i = 0; i < 100000; i++) {
            //calculate hash for the input
            calculateHash(input);
        }
        Long endTime = getTime().getTime();
        Long timeDiff = endTime - startTime;
        //find the time diff in seconds
        float diffInSecs=(float)timeDiff/1000;
```

```
        //divide million by time taken in secs
        hashPerSec = (int) (1000000 / diffInSecs);
    }

    //calculate SHA-256 hash for the passed input
    public String calculateHash(String input) {
        byte[] digest = null;
        try {
            md.update(input.getBytes("UTF-8"));
            digest = md.digest();

        }catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return digest.toString();
    }

    //get the block at ith position
    public Block getBlock(int i) {
        return blocks.get(i);
    }

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

    //get the combined difficulty of all blocks
    public int getTotalDifficulty() {
        int totalDiff = 0;
        for (int i = 0; i < getChainSize(); i++) {
            totalDiff += getBlock(i).getDifficulty();
        }
        return totalDiff;
    }

    //get total expected hashes
    public double getTotalExpectedHashes() {
        double totHash = 0;
        for (int i = 0; i < getChainSize(); i++) {
            totHash += Math.pow(16, getBlock(i).difficulty);
        }
        return totHash;
    }

    //check if entire chain is valid or not
    public boolean isChainValid() {
        Block b = null;
        boolean flag = false;
        try {
            //will execute if the chain has only 1 block
            if (getChainSize() == 1) {
                b = getBlock(0);
                //check for valid block and if the hash of the block is same
as chainhash
                if (b.getHash().equalsIgnoreCase(chainHash) &&
checkValidBlock(b))
```

```
                    flag = true;
                else {
                    flag = false;
                    //if the block is corrupt, set the corrupt node ID
                    setCorruptNodeId(b.getIndex());
                }
            }
            //will execute if the chain has more than 1 block
            else {
                for (int i = 0; i < getChainSize(); i++) {
                if (i != getChainSize() - 1) {
                    b = getBlock(i);
                    if (b.getHash().equalsIgnoreCase(blocks.get(i +
1).getPreviousHash()) && checkValidBlock(b))
                        flag = true;
                    else {
                        flag = false;
                        setCorruptNodeId(b.getIndex());
                        break;
                    }
                } else {
                    b = getBlock(i);
                    if (b.getHash().equalsIgnoreCase(chainHash) &&
checkValidBlock(b))
                        flag = true;
                    else {
                        flag = false;
                        setCorruptNodeId(b.getIndex());
                        break;
                    }
                }

                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return flag;
    }

    //checking if the hash of the block as same or more no of leading 0 as
difficulty
    // Eg: if difficulty=2 then valid hash='00...'
    private boolean checkProofOfWork(String hexaDecimalStr, int difficulty) {
        int count = 0;
        for (char c : hexaDecimalStr.toCharArray()) {
            if (c == '0')
                count++;
            else
                break;
        }
        if (count >= difficulty) {
            return true;
        }
        return false;
    }
```

```
    //repairs the invalid chain
    public void repairChain() {
        for (int i = 0; i < getChainSize(); i++) {
            if (!checkValidBlock(getBlock(i))) {
                getBlock(i).proofOfWork();
                reAssignPrevHash(getBlock(i));
            }
        }
    }

    //re-assigns updated previous hash for all the blocks after the corrupt
block
    private void reAssignPrevHash(Block block) {
        for (int i = block.getIndex(); i < getChainSize(); i++) {
            if (i + 1 < getChainSize())
                blocks.get(i + 1).setPreviousHash(getBlock(i).getHash());
        }
        chainHash = blocks.get(getChainSize() - 1).getHash();
    }

    //check if a block is valid by verifying its proof of work
    public boolean checkValidBlock(Block b) {
        String message = b.getIndex() + "," + b.getTimestamp() + "," +
b.getData() + "," + b.getPreviousHash() + "," + b.getNonce() + "," +
b.getDifficulty();
        try {
            md.update(message.getBytes("UTF-8"));
            byte[] digest = md.digest();
            String hexaDecimalStr = bytesToHex(digest);
            if (checkProofOfWork(hexaDecimalStr, b.getDifficulty())) {
                return true;
            }
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return false;
    }

    @Override
    public String toString() {
        JSONObject obj = new JSONObject();
        JSONArray jsonArray = new JSONArray(blocks.toString());
        obj.put("ds_chain", jsonArray);
        obj.put("chainHash", chainHash);
        return obj.toString();
    }

    //Lab 1-https://github.com/CMU-Heinz-95702/Lab1-InstallationAndRaft
    //converts the bytes to hexadecimal string
    public static String bytesToHex(byte[] bytes) {
        final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
```

```
        return new String(hexChars);
    }
}
```