Task 0 Execution

```
C:\Users\kanis\.jdks\openjdk-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=53130:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\kanis\IdeaProjects\DistributedSystes\Project3\Project3Task0\out\production\Project3Task0 blockchaintask.BlockChain

0. View basic blockchain status
1. Add a transaction to 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 the chain: 1
Difficulty of the most recent block: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 2873563
Expected total hashes required for the whole chain: 256.0
Nonce for the most recent block: 4
Chain hash: 009AFFED3499DDA88F07358C965E637B435E2C8577B38057DB0ADB5AC205114E

0. View basic blockchain status
1. Add a transaction to 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
3
Enter transaction
Kanishka pays Bob 100 dscoin
Total execution time to add this block was 34
```

```
31 milliseconds
32
33 0. View basic blockchain status
34 1. Add a transaction to blockchain.
35 2. Verify the blockchain.
36 3. View the blockchain.
37 4. Corrupt the chain.
38 5. Hide the corruption by repairing the chain
39 6. Exit
40 1
41 Enter difficulty > 0
42 3
43 Enter transaction
44 Bob pays Kanishka 20 dscoin
45 Total execution time to add this block was 27
   milliseconds
46
47 0. View basic blockchain status
48 1. Add a transaction to blockchain.
49 2. Verify the blockchain.
50 3. View the blockchain.
51 4. Corrupt the chain.
52 5. Hide the corruption by repairing the chain
53 6. Exit
54 1
55 Enter difficulty > 0
56 4
57 Enter transaction
58 Kanishka pays Carol 23 dscoin
59 Total execution time to add this block was 15
   milliseconds
60
61 0. View basic blockchain status
62 1. Add a transaction to blockchain.
63 2. Verify the blockchain.
64 3. View the blockchain.
65 4. Corrupt the chain.
66 5. Hide the corruption by repairing the chain
67 6. Exit
68 1
69 Enter difficulty > 0
```

```
70 5
71 Enter transaction
72 Donna pays Kanishka 34 dscoin
73 Total execution time to add this block was 1772
   milliseconds
74
75 0. View basic blockchain status
76 1. Add a transaction to blockchain.
77 2. Verify the blockchain.
78 3. View the blockchain.
79 4. Corrupt the chain.
80 5. Hide the corruption by repairing the chain
81 6. Exit
82 3
83 View the blockchain
84 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
   -10-23 15:54:25.55", "Tx" : "Genesis", "PrevHash
   " : "", "nonce" : "4", "difficulty" : "2"}
85 {"index" : "1","time stamp" : "2021-10-23 15:54:41.
   006", "Tx" : "Kanishka pays Bob 100 dscoin", "
   PrevHash" : "
   009AFFED3499DDA88F07358C965E637B435E2C8577B38057DB0A
   DB5AC205114E", "nonce" : "4003", "difficulty" : "3"}
86 {"index" : "2","time stamp" : "2021-10-23 15:54:54.
   298", "Tx" : "Bob pays Kanishka 20 dscoin", "
   PrevHash" : "
   0004FB16C09171CE87065D657677197BC5BCC7A6EC7284670530
   0851717884E1", "nonce" : "9103", "difficulty" : "3"}
87 {"index" : "3","time stamp" : "2021-10-23 15:55:06.
   672", "Tx" : "Kanishka pays Carol 23 dscoin", "
   PrevHash" : "
   000F45B806DD6B1345DB1E6EED5B004A4B965D089474969FF404
   F8DA4BFAE946", "nonce" : "25539", "difficulty" : "4
   "}
88 {"index" : "4","time stamp" : "2021-10-23 15:55:20.
   222", "Tx" : "Donna pays Kanishka 34 dscoin", "
   PrevHash" : "
   00009985EA97F29F52D3AEC52BD18E1F9D77FB749CA6A688B5B4
   CDDD2D4CE87B", "nonce" : "1286077", "difficulty" : "
   5"}
89 ],"chainHash":"
```

```
 89
    000009F7AC3789F4297A5DDCEFB3AA629770EF6689AF55BE65A7
    B786EC90567D"}
 90
 91 0. View basic blockchain status
 92 1. Add a transaction to blockchain.
 93 2. Verify the blockchain.
 94 3. View the blockchain.
 95 4. Corrupt the chain.
 96 5. Hide the corruption by repairing the chain
 97 6. Exit
 98 1
 99 Enter difficulty > 0
100 2
101 Enter transaction
102 Carol pays Donna 1 dscoin
103 Total execution time to add this block was 0
    milliseconds
104
105 0. View basic blockchain status
106 1. Add a transaction to blockchain.
107 2. Verify the blockchain.
108 3. View the blockchain.
109 4. Corrupt the chain.
110 5. Hide the corruption by repairing the chain
111 6. Exit
112 3
113 View the blockchain
114 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:54:25.55", "Tx" : "Genesis", "PrevHash
    " : "", "nonce" : "4", "difficulty" : "2"}
115 {"index" : "1","time stamp" : "2021-10-23 15:54:41.
    006", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    009AFFED3499DDA88F07358C965E637B435E2C8577B38057DB0A
    DB5AC205114E", "nonce" : "4003", "difficulty" : "3"}
116 {"index" : "2","time stamp" : "2021-10-23 15:54:54.
    298", "Tx" : "Bob pays Kanishka 20 dscoin", "
    PrevHash" : "
    0004FB16C09171CE87065D657677197BC5BCC7A6EC7284670530
    0851717884E1", "nonce" : "9103", "difficulty" : "3"}
```

```
{"index" : "3","time stamp" : "2021-10-23 15:55:06.672", "Tx" : "Kanishka pays Carol 23 dscoin", "PrevHash" : "000F45B806DD6B1345DB1E6EED5B004A4B965D089474969FF404F8DA4BFAE946", "nonce" : "25539", "difficulty" : "4"}
{"index" : "4","time stamp" : "2021-10-23 15:55:20.222", "Tx" : "Donna pays Kanishka 34 dscoin", "PrevHash" : "00009985EA97F29F52D3AEC52BD18E1F9D77FB749CA6A688B5B4CDDD2D4CE87B", "nonce" : "1286077", "difficulty" : "5"}
{"index" : "5","time stamp" : "2021-10-23 15:55:35.833", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash" : "000009F7AC3789F4297A5DDCEFB3AA629770EF6689AF55BE65A7B786EC90567D", "nonce" : "247", "difficulty" : "2"}
],"chainHash":"00CF2F32C34BF4B719FCA29291ABBD9F0C85A00187C95DB335AA2F365E093492"}

0. View basic blockchain status
1. Add a transaction to 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 the block ID of block to corrupt
0
Enter new data for block 0
Carol pays Kanishka 1000 dscoin
Block 0 now holds Carol pays Kanishka 1000 dscoin

0. View basic blockchain status
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
```

```
142 5. Hide the corruption by repairing the chain
143 6. Exit
144 3
145 View the blockchain
146 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:54:25.55", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "4", "
    difficulty" : "2"}
147 {"index" : "1","time stamp" : "2021-10-23 15:54:41.
    006", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    009AFFED3499DDA88F07358C965E637B435E2C8577B38057DB0A
    DB5AC205114E", "nonce" : "4003", "difficulty" : "3"}
148 {"index" : "2","time stamp" : "2021-10-23 15:54:54.
    298", "Tx" : "Bob pays Kanishka 20 dscoin", "
    PrevHash" : "
    0004FB16C09171CE87065D657677197BC5BCC7A6EC7284670530
    0851717884E1", "nonce" : "9103", "difficulty" : "3"}
149 {"index" : "3","time stamp" : "2021-10-23 15:55:06.
    672", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    000F45B806DD6B1345DB1E6EED5B004A4B965D089474969FF404
    F8DA4BFAE946", "nonce" : "25539", "difficulty" : "4
    "}
150 {"index" : "4","time stamp" : "2021-10-23 15:55:20.
    222", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    00009985EA97F29F52D3AEC52BD18E1F9D77FB749CA6A688B5B4
    CDDD2D4CE87B", "nonce" : "1286077", "difficulty" : "
    5"}
151 {"index" : "5","time stamp" : "2021-10-23 15:55:35.
    833", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    000009F7AC3789F4297A5DDCEFB3AA629770EF6689AF55BE65A7
    B786EC90567D", "nonce" : "247", "difficulty" : "2"}
152 ],"chainHash":"
    00CF2F32C34BF4B719FCA29291ABBD9F0C85A00187C95DB335AA
    2F365E093492"}
153
154 0. View basic blockchain status
155 1. Add a transaction to blockchain.
```

```
156 2. Verify the blockchain.
157 3. View the blockchain.
158 4. Corrupt the chain.
159 5. Hide the corruption by repairing the chain
160 6. Exit
161 5
162 Repairing the entire chain
163 Total execution time required to repair the chain
    was 1086 milliseconds
164
165 0. View basic blockchain status
166 1. Add a transaction to blockchain.
167 2. Verify the blockchain.
168 3. View the blockchain.
169 4. Corrupt the chain.
170 5. Hide the corruption by repairing the chain
171 6. Exit
172 3
173 View the blockchain
174 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:54:25.55", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "554", "
    difficulty" : "2"}
175 {"index" : "1","time stamp" : "2021-10-23 15:54:41.
    006", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    003E0630E28D707C04E1DDAC9F44DB3D004001AF11DC2F7D2DF0
    73E5A4F46C79", "nonce" : "11066", "difficulty" : "3
    "}
176 {"index" : "2","time stamp" : "2021-10-23 15:54:54.
    298", "Tx" : "Bob pays Kanishka 20 dscoin", "
    PrevHash" : "
    0007C2DBF37BFC91D555564A8643ECC08861AD243B1EEE607DB5
    71E975361868", "nonce" : "5574", "difficulty" : "3"}
177 {"index" : "3","time stamp" : "2021-10-23 15:55:06.
    672", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    0003730A69DAB69A76E5241DB8697C6C39E10379AE6A276B8657
    5643FADB0B6F", "nonce" : "10217", "difficulty" : "4
    "}
178 {"index" : "4","time stamp" : "2021-10-23 15:55:20.
```

```
178 222", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    0000D64411944902FA0163EC0CB6A302429AE746421791FA28E3
    CF50BEE94409", "nonce" : "865806", "difficulty" : "5
    "}
179 {"index" : "5","time stamp" : "2021-10-23 15:55:35.
    833", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    00000A46BC499D04BCD083B9E3422EAFE0C32C6F5C8970CABD02
    88696C6A5FFB", "nonce" : "204", "difficulty" : "2"}
180 ],"chainHash":"
    009D6FA8F5D7003F3F7DBC76C9E3AC407FBC366BCF4E2C9D866F
    242EC8DD1551"}
181
182 0. View basic blockchain status
183 1. Add a transaction to blockchain.
184 2. Verify the blockchain.
185 3. View the blockchain.
186 4. Corrupt the chain.
187 5. Hide the corruption by repairing the chain
188 6. Exit
189 1
190 Enter difficulty > 0
191 4
192 Enter transaction
193 Edward pays Kanishka 34 dscoin
194 Total execution time to add this block was 0
    milliseconds
195
196 0. View basic blockchain status
197 1. Add a transaction to blockchain.
198 2. Verify the blockchain.
199 3. View the blockchain.
200 4. Corrupt the chain.
201 5. Hide the corruption by repairing the chain
202 6. Exit
203 2
204 Verifying the entire chain
205 Chain verification: true
206 Total execution time to verify the chain was 0
    milliseconds
```

```
207
208 0. View basic blockchain status
209 1. Add a transaction to blockchain.
210 2. Verify the blockchain.
211 3. View the blockchain.
212 4. Corrupt the chain.
213 5. Hide the corruption by repairing the chain
214 6. Exit
215 4
216 Corrupt the Blockchain
217 Enter the block ID of block to corrupt
218 2
219 Enter new data for block 2
220 Franks pays Kanishka 3 dscoin
221 Block 2 now holds Franks pays Kanishka 3 dscoin
222
223 0. View basic blockchain status
224 1. Add a transaction to blockchain.
225 2. Verify the blockchain.
226 3. View the blockchain.
227 4. Corrupt the chain.
228 5. Hide the corruption by repairing the chain
229 6. Exit
230 2
231 Verifying the entire chain
232 ..Improper hash at node2.Does not begin with 000
233 Chain verification: false
234 Total execution time to verify the chain was 1
    milliseconds
235
236 0. View basic blockchain status
237 1. Add a transaction to blockchain.
238 2. Verify the blockchain.
239 3. View the blockchain.
240 4. Corrupt the chain.
241 5. Hide the corruption by repairing the chain
242 6. Exit
243 5
244 Repairing the entire chain
245 Total execution time required to repair the chain
    was 653 milliseconds
```

```
246
247 0. View basic blockchain status
248 1. Add a transaction to blockchain.
249 2. Verify the blockchain.
250 3. View the blockchain.
251 4. Corrupt the chain.
252 5. Hide the corruption by repairing the chain
253 6. Exit
254 3
255 View the blockchain
256 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:54:25.55", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "554", "
    difficulty" : "2"}
257 {"index" : "1","time stamp" : "2021-10-23 15:54:41.
    006", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    003E0630E28D707C04E1DDAC9F44DB3D004001AF11DC2F7D2DF0
    73E5A4F46C79", "nonce" : "11066", "difficulty" : "3
    "}
258 {"index" : "2","time stamp" : "2021-10-23 15:54:54.
    298", "Tx" : "Franks pays Kanishka 3 dscoin", "
    PrevHash" : "
    0007C2DBF37BFC91D555564A8643ECC08861AD243B1EEE607DB5
    71E975361868", "nonce" : "12158", "difficulty" : "3
    "}
259 {"index" : "3","time stamp" : "2021-10-23 15:55:06.
    672", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    000F2DA646C1B0825AB7AED8FAE36C62DD81BF9A90D10F14AB1A
    A329DD58D02C", "nonce" : "31984", "difficulty" : "4
    "}
260 {"index" : "4","time stamp" : "2021-10-23 15:55:20.
    222", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    000050ED58DD347D57E9D016C4A78140A61B4CDD4C86AC72F22F
    0E962CC9A43C", "nonce" : "407401", "difficulty" : "5
    "}
261 {"index" : "5","time stamp" : "2021-10-23 15:55:35.
    833", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
```

```
261
    000002A9FBED51C8DBD5078FC62B0A50701A82F78A38EB83E534
    610B98865E8B", "nonce" : "31", "difficulty" : "2"}
262 {"index" : "6","time stamp" : "2021-10-23 15:56:20.
    402", "Tx" : "Edward pays Kanishka 34 dscoin", "
    PrevHash" : "
    0039C4D66D5D04D90AFC5ED1BCAFE3E9E64A62AA84D6CAA2FDAF
    26AA8303433A", "nonce" : "96267", "difficulty" : "4
    "}
263 ],"chainHash":"
    0000846CBE00A4AACF457C5D51C77BE74CEC9F5A160C3B90E00F
    02ABF58C7308"}
264
265 0. View basic blockchain status
266 1. Add a transaction to blockchain.
267 2. Verify the blockchain.
268 3. View the blockchain.
269 4. Corrupt the chain.
270 5. Hide the corruption by repairing the chain
271 6. Exit
272 2
273 Verifying the entire chain
274 Chain verification: true
275 Total execution time to verify the chain was 0
    milliseconds
276
277 0. View basic blockchain status
278 1. Add a transaction to blockchain.
279 2. Verify the blockchain.
280 3. View the blockchain.
281 4. Corrupt the chain.
282 5. Hide the corruption by repairing the chain
283 6. Exit
284 0
285 Current size of the chain: 7
286 Difficulty of the most recent block: 4
287 Total difficulty for all blocks: 23
288 Approximate hashes per second on this machine:
    2873563
289 Expected total hashes required for the whole chain:
    1188352.0
```

```
290 Nonce for the most recent block: 96267
291 Chain hash:
    0000846CBE00A4AACF457C5D51C77BE74CEC9F5A160C3B90E00F
    02ABF58C7308
292
293 0. View basic blockchain status
294 1. Add a transaction to blockchain.
295 2. Verify the blockchain.
296 3. View the blockchain.
297 4. Corrupt the chain.
298 5. Hide the corruption by repairing the chain
299 6. Exit
300 6
301
302 Process finished with exit code 0
303
```

```java
package blockchaintask;

import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;

/**
 * class Block:
 * The cass block is used to create a single block in the blockchain
 * It is also used to compute the proof of work and calculate the hash of a particular block
 * The block created stores the hash of the previous block
 */
public class Block {
    // Private variable of the block that are required to identify a block.
    // Like its index, the timestamp is which it was created
    // The nonce of the block, the transaction data and the previous hash
    private int index, difficulty;
    private BigInteger nonce;
    private Timestamp timestamp;
    private String data, previousHash;

    /**
     * Block Constructor
     * The constructor is used to create a block and initialize it with the values provided from the chain
     * @param index the number of the block in the blockchain
     * @param timestamp the timestamp at which the block was created
     * @param data the transaction data at the block
     * @param difficulty the difficulty to calculate the hash of the block
     */
```

```
32      Block(int index, Timestamp timestamp, String data
   , int difficulty)
33      {
34          this.index = index;
35          this.timestamp = timestamp;
36          this.data = data;
37          this.difficulty = difficulty;
38      }
39
40      /**
41       * function setIndex
42       * Sets the index of the block
43       * @param index
44       */
45      public void setIndex(int index) {
46          this.index = index;
47      }
48
49      /**
50       * function setDifficulty
51       * Sets the difficulty of the block
52       * @param difficulty
53       */
54      public void setDifficulty(int difficulty) {
55          this.difficulty = difficulty;
56      }
57
58      /**
59       * function setTimestamp
60       * Sets the timestamp of the block
61       * @param timestamp
62       */
63      public void setTimestamp(Timestamp timestamp) {
64          this.timestamp = timestamp;
65      }
66
67      /**
68       * function setData
69       * Sets the data/transaction of the block
70       * @param data
71       */
```

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

    /**
     * function setPreviousHash
     * set the hash of the block's parent
     * @param previousHash
     */
    public void setPreviousHash(String previousHash
) {
        this.previousHash = previousHash;
    }

    /**
     * function getIndex
     * gets the index of the block
     * @return index
     */
    public int getIndex() {
        return index;
    }

    /**
     * function getDifficuty
     * gets the difficulty of the block
     * @return difficulty
     */
    public int getDifficulty() {
        return difficulty;
    }

    /**
     * function getTimestamp
     * gets the timestamp of the block
     * @return timestamp
     */
    public Timestamp getTimestamp() {
        return timestamp;
    }

```

```
112     /**
113      * function getData
114      * gets the transaction of the block
115      * @return data
116      */
117     public String getData() {
118         return data;
119     }
120 
121     /**
122      * function getPreviousHash
123      * gets the parent's hash of the block
124      * @return previous hash
125      */
126     public String getPreviousHash() {
127         return previousHash;
128     }
129 
130     /**
131      * function getNonce
132      * gets the nonce of the block
133      * @return nonce
134      */
135     public BigInteger getNonce()
136     {
137         return nonce;
138     }
139 
140     /**
141      * function calculate hash
142      * calculates the hash of of a string
143      * @return hashed String
144      * @throws NoSuchAlgorithmException for the SHA
    calculation
145      */
146     public String calculateHash() throws
    NoSuchAlgorithmException {
147         //Our string contains the index, timestamp,
    data, previous hash, nonce and difficulty
148         String hash = index + timestamp.toString
    () + data + previousHash + nonce + difficulty;
```

```
149         MessageDigest digest;
150         //Getting the instance of SHA-256
151         digest = MessageDigest.getInstance("SHA-256"
    );
152         //encoding with SHA-256
153         byte[] encodedhash = digest.digest(
154                 hash.getBytes(StandardCharsets.UTF_8
    ));
155         return bytesToHex(encodedhash);
156     }
157 
158     /**
159      * function byteToHex
160      * The function is used to convert byte to hex
    of the encoded string
161      * @pargm hash takes the hash
162      * @return
163      */
164     private static String bytesToHex(byte[] hash) {
165         //The hexString is double the length of the
    hash
166         StringBuilder hexString = new StringBuilder(
    2 * hash.length);
167         //Convert each value of string to a hex
168         for (int i = 0; i < hash.length; i++) {
169             String hex = Integer.toHexString(0xff &
    hash[i]);
170             if(hex.length() == 1) {
171                 hexString.append('0');
172             }
173             //append the values together
174             hexString.append(hex);
175         }
176         //return the string
177         return hexString.toString().toUpperCase();
178     }
179 
180     /**
181      * function proofOfWork
182      * The function does the proof of work of a
    block
```

```
183      * It calculates the hash based on the
    difficulty and increases the  nonce till
184      * the initial 0s are not equal to the
    difficulty
185      * @return the hash value after the proof of
    work
186      * @throws NoSuchAlgorithmException
187      */
188     public String proofOfWork() throws
    NoSuchAlgorithmException {
189         //The nonce is initialized to 0
190         this.nonce = BigInteger.ZERO;
191         //calculate hash with 0 nonce
192         String h = calculateHash();
193         //string builder of number of 0s equal to
    the difficulty number
194         StringBuilder sb = new StringBuilder();
195         for(int i = 0; i < difficulty; i++)
196         {
197             sb.append("0");
198         }
199         //if the number of 0s in the stringbuilder
    match the start of the hash then we stop calculating
    , else we continue calculating
200         while(!(h.substring(0,difficulty).equals(sb.
    toString()))) {
201             this.nonce = nonce.add(BigInteger.ONE);
    //adding one to nonce everytime the string does not
    match
202             h = calculateHash(); //calculating hash
    again
203         }
204         //returning the hash
205         return h;
206     }
207
208     /**
209      * function toString
210      * The function is used to display the data of
    the blocks
211      * @return String value
```

```
212      */
213     @Override
214     public String toString() {
215         return "{" +
216                 "\"index\" : \"" + index + "\"" +
217                 ",\"time stamp\" : \"" + timestamp
    + "\"" +
218                 ", \"Tx\" : \"" + data + "\"" +
219                 ", \"PrevHash\" : \"" + previousHash
    + "\"" +
220                 ", \"nonce\" : \"" + nonce + "\"" +
221                 ", \"difficulty\" : \"" + difficulty
    + "\"" +
222                 "}";
223     }
224 }
225
```

```java
package blockchaintask;

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

/**
 * class BlockChain
 * The class is used to create a chain of blocks
 * To see how chain gets corrupted, validate a chain and to repair a chain
 * We also check the time a particular function takes on the chain
 */
public class BlockChain {
    ArrayList<Block> blocks;
    private String chainHash;
    int hashesPerSecond;
    private String errorMessage = "";

    /**
     * Constructor BlockChain()
     * The constructor is used to initialize the list of blocks that will be connected to one another
     * The hash of the latest block and the hashes per second when there are 1 million hashes being computed
     */
    BlockChain()
    {
        this.blocks = new ArrayList<>();
        this.chainHash = "";
        this.hashesPerSecond = 0;
    }

    /**
     * function getTime
     * The get time function is used to get the
```

```
36 current time in milliseconds
37      * @return timestamp (time)
38      */
39     public Timestamp getTime()
40     {
41         return new Timestamp(System.currentTimeMillis
   ());
42     }
43 
44     /**
45      * function getLatestBlock
46      * the function is used to get the latest block
   of the chain. Recently added
47      * @return
48      */
49     public Block getLatestBlock()
50     {
51         return blocks.get(blocks.size() - 1);
52     }
53 
54     /**
55      * function getChainSize
56      * The function gets the size of the blockchain
57      * @return
58      */
59     public int getChainSize()
60     {
61         return blocks.size();
62     }
63 
64     /**
65      * function computeHashesPerSecond
66      * The function is used the compute 1 million
   hashes and check how many hashes the system can
   compute in one second
67      * @throws NoSuchAlgorithmException
68      */
69     public void computeHashesPerSecond() throws
   NoSuchAlgorithmException {
70         Timestamp startTime = getTime();
71         String hash =  "00000000";
```

```
 72         for(int i = 0; i < 1000000; i++) {
 73             MessageDigest digest;
 74             digest = MessageDigest.getInstance("SHA-
    256");
 75             //encoding with SHA-256
 76             byte[] encodedhash = digest.digest(
 77                     hash.getBytes(StandardCharsets.
    UTF_8));
 78         }
 79         Timestamp endTime = getTime();
 80         long totalTime = endTime.getTime() -
    startTime.getTime();
 81         this.hashesPerSecond = (int) ((1000000*1000
    )/totalTime);
 82     }
 83 
 84     /**
 85      * function get hashesPerSecond
 86      * return the hashes per second to display to
    the user
 87      * @return hashes per second
 88      */
 89     public int getHashesPerSecond()
 90     {
 91         return hashesPerSecond;
 92     }
 93 
 94     /**
 95      * function addBlock
 96      * It adds the new block provided to the user,
    after performing certain checks on it
 97      * @param newBlock a new block to be added to
    the chain
 98      * @throws NoSuchAlgorithmException
 99      */
100     public void addBlock(Block newBlock) throws
    NoSuchAlgorithmException {
101         //To check whether the block is valid or not
     before adding the block
102         StringBuilder sb = new StringBuilder();
103         //String builder to get the number of 0s
```

```
103 compared to difficulty
104         for(int i = 0; i < newBlock.getDifficulty
    (); i++)
105         {
106             sb.append("0");
107         }
108         //Calculate the hash of the new block
    created
109         String hash = newBlock.calculateHash();
110         //If the substring till the difficulty of
    the hash if equal to the string builder's number of
    0s
111         //and the previous hash of the new block is
    equal to the lastest block's hash only then add the
    block
112         //and assign the chainHash the new block's
    hash
113         if(hash.substring(0,newBlock.getDifficulty
    ()).equals(sb.toString()) && newBlock.
    getPreviousHash().equals(getLatestBlock().
    calculateHash())) {
114             blocks.add(newBlock); //add the block
115             this.chainHash = hash; //chainHash's new
     hash is the latest block's hash
116         }
117         else
118         {
119             //If verification fails
120             System.out.println("Addition failed due
    to failed verification or wrong previous hash.");
121         }
122     }
123
124     /**
125      * function toString
126      * The function is used to print the blockchain
    in the string
127      * @return
128      */
129     public String toString()
130     {
```

```
131         //sb to append the entire blockchain in a
    String
132         StringBuilder sb = new StringBuilder();
133         sb.append("{" + "\"ds_chain{\" : [ ");
134         //appending the block's to string to the
    block chain
135         for(int i = 0; i < blocks.size(); i++)
136         {
137             sb.append(getBlock(i).toString()+"\n");
138         }
139         sb.append("],\"chainHash\":\"" + chainHash
     + "\"}");
140         //return the string
141         return sb.toString();
142     }
143 
144     /**
145      * function getBlock
146      * To get the block at ith position
147      * @param i the position of the block
148      * @return the block
149      */
150     public Block getBlock(int i)
151     {
152         return blocks.get(i);
153     }
154 
155     /**
156      * function getTotalDifficulty
157      * Get the total difficulty of the entire block
    chain
158      * @return the total difficulty
159      */
160     public int getTotalDifficulty()
161     {
162         int totalDifficulty = 0;
163         //get the difficulty of every block and add
    it
164         for(int i = 0; i < blocks.size(); i++)
165         {
166             totalDifficulty += getBlock(i).
```

```
166 getDifficulty();
167         }
168         return totalDifficulty;
169     }
170 
171     /**
172      * function getTotalExpectedHAshes
173      * Get the total expected hashes of the
    blockchain
174      * @return the expected hashes
175      */
176     public double getTotalExpectedHashes()
177     {
178         double totalExpectedHashes = 0.0;
179         //Count the hashes based on the difficulty
    of every block and add them
180         for(int i = 0; i < blocks.size(); i++)
181         {
182             totalExpectedHashes += Math.pow(16,
    getBlock(i).getDifficulty());
183         }
184         //return the total expected hashes
185         return totalExpectedHashes;
186     }
187 
188     /**
189      * function isChainValid
190      * Check if the chain is valid or not.
191      * If it is corrupted at some point
192      * @return the boolean value of true or false
193      * @throws NoSuchAlgorithmException
194      */
195     public boolean isChainValid() throws
    NoSuchAlgorithmException {
196         //If there is ony 1 block in the chain
197         if(blocks.size() == 1)
198         {
199             //Get the block and create a string
    builder of 0s till its difficulty value
200             Block block = blocks.get(0);
201             StringBuilder sb = new StringBuilder();
```

```
202             for(int i = 0; i < block.getDifficulty
    (); i++)
203             {
204                 sb.append("0");
205             }
206             //calculate the hash of the block
207             String hash = block.calculateHash();
208             //If the substring of hash matchs the sb
    's zeros and the chainHash is equal to it's hash
    then it is valid
209             //Else the chain is not valid and return
     false with the 1st node being invalid
210             if(!(hash.substring(0,block.
    getDifficulty()).equals(sb.toString()) && chainHash.
    equals(hash))) {
211                 errorMessage = "..Improper hash at
    node 1. Does not begin with " + sb;
212                 return false;
213             }
214         }
215         //If the size of blockchain is more than 1
216         else
217         {
218             String hash = "";
219             //For every block in the chain
220             for(int i = 1; i < blocks.size(); i++)
221             {
222                 //Get the block and create a string
    builder of 0s till its difficulty value
223                 StringBuilder sb = new StringBuilder
    ();
224                 for(int j = 0; j < getBlock(i).
    getDifficulty(); j++)
225                 {
226                     sb.append("0");
227                 }
228                 //calculate the hash of the block
229                 hash = getBlock(i).calculateHash();
230                 //If the substring of hash matchs
    the sb's zeros and the previous hash of the block is
     equal to hash of the previous block
```

```
231                 //Else the chain is not valid and
    return false with the node that is invalid
232                 if(!(hash.substring(0,getBlock(i).
    getDifficulty()).equals(sb.toString()) && getBlock(i
    ).getPreviousHash().equals(blocks.get(i-1).
    calculateHash()))) {
233                     errorMessage = "..Improper hash
    at node" + i + ".Does not begin with " + sb;
234                     return false;
235                 }
236             }
237             //Also check if the hash that we get at
    the end of the loop is valid to the chain hash. If
    not, return false
238             if(!(hash.equals(chainHash)))
239             {
240                 errorMessage = "..Improper value of
    chainHash";
241                 return false;
242             }
243         }
244         return true; //return true if all cases pass
245     }
246 
247     /**
248      * function repairChain
249      * The function is to repair the chain when it
    is corrupted.
250      * A different message is set in place of the
    previous one for the block
251      * @throws NoSuchAlgorithmException
252      */
253     public void repairChain() throws
    NoSuchAlgorithmException {
254         //For every block
255         for(int i= 0; i< blocks.size(); i++)
256         {
257             //Get the block and create a string
    builder of 0s till its difficulty value
258             StringBuilder sb = new StringBuilder();
259             for(int j = 0; j < getBlock(i).
```

```
259 getDifficulty(); j++)
260             {
261                 sb.append("0");
262             }
263             //calculate the hash of the block
264             String hash = getBlock(i).calculateHash
    ();
265             //If the substring of hash matches the
    sb's zeros then the block needs no repairing,
266             // if not then calculate the proof of
    work of the block all over again
267             if(!(hash.substring(0,getBlock(i).
    getDifficulty()).equals(sb.toString()))) {
268                 getBlock(i).proofOfWork(); //
    Calculating the proof of work
269                 if(i+1 != blocks.size()) { // If it
    is not the last block then set the next block's
    previous hash as the hash just calculated
270                     blocks.get(i + 1).
    setPreviousHash(getBlock(i).calculateHash());
271                 }
272                 else
273                 {
274                     //If it is the last block then
    chainHash should be the block's hash
275                     chainHash = getBlock(i).
    calculateHash();
276                 }
277             }
278         }
279     }
280 
281     /**
282      * function setChainHash
283      * The function sets the chain hash of the block
     chain
284      * @param chainHash String
285      */
286     public void setChainHash(String chainHash)
287     {
288         this.chainHash = chainHash;
```

```
289     }
290 
291     /**
292      * getChainHash
293      * The function reti=urns the chain hash of the
    block chain
294      * @return String chain hash
295      */
296     public String getChainHash()
297     {
298         return chainHash;
299     }
300 
301     /**
302      * function getErrorMessage()
303      * sets the error message to the point of
    failure if the chain is valid or not
304      * @return the error message of the chain being
    valid or not
305      */
306     public String getErrorMessage()
307     {
308         return errorMessage;
309     }
310 
311     /**
312      * function setErrorMessage
313      * Setting the error message as back to empty
    once the chain verification failure point is known
314      * @param errorMessage the message if the chain
    is valid or not
315      */
316     public void setErrorMessage(String errorMessage)
317     {
318         this.errorMessage = errorMessage;
319     }
320 
321     /**
322      * function main
323      * The main function gives user the choices in
    order to manipulate the blockchain
```

```
324      * A genisis block is created and the user is
    then given options to add a transaction
325      * corrupt a chain or repair a chain
326      * If the chain is valid and display the chain
    along with the time taken by each function
327      * @param args
328      */
329     public static void main(String[] args) {
330         //Taking the user input
331         Scanner input = new Scanner(System.in);
332         //Creating a blockchain object
333         BlockChain blockChain = new BlockChain();
334         //Creating a genesis block wth difficulty of
     2
335         Block block = new Block(0, blockChain.
    getTime(), "Genesis", 2);
336         //Setting the previous hash of 1t block to 0
337         block.setPreviousHash("");
338         try {
339             //Computing the proof of work of the 1st
     block
340             block.proofOfWork();
341             //Adding the block to the chain
342             blockChain.blocks.add(block);
343             //Computing a million hashes per second
    and displaying the number of hashes done by system
    in a second
344             blockChain.computeHashesPerSecond();
345             //Setting the value of chainHash
    variable to the 1st block's hash
346             blockChain.setChainHash(block.
    calculateHash());
347         } catch (NoSuchAlgorithmException e) {
348             e.printStackTrace();
349         }
350
351         while(true)
352         {
353             //Menu choices for the user to choose
    from
354             System.out.println("\n0. View basic
```

```
354 blockchain status");
355             System.out.println("1. Add a transaction
     to blockchain.");
356             System.out.println("2. Verify the
    blockchain.");
357             System.out.println("3. View the
    blockchain.");
358             System.out.println("4. Corrupt the chain
    .");
359             System.out.println("5. Hide the
    corruption by repairing the chain");
360             System.out.println("6. Exit");
361 
362             int choice = Integer.parseInt(input.
    nextLine());
363 
364             switch(choice)
365             {           //Displaying the data of the
     blockchain. The chain size, difficulty of the
    latest block, total difficulty
366                         //the hashes per second
    computed above, total expected hashes by computing
    expected hashes of each block
367                         //Nonce of the latest block
    and the chainHash i.e the hash of the latest block
368                 case 0: System.out.println("Current
    size of the chain: "+ blockChain.getChainSize());
369                         System.out.println("
    Difficulty of the most recent block: "+ blockChain.
    getLatestBlock().getDifficulty());
370                         System.out.println("Total
    difficulty for all blocks: "+ blockChain.
    getTotalDifficulty());
371                         System.out.println("
    Approximate hashes per second on this machine: "+
    blockChain.getHashesPerSecond());
372                         System.out.println("Expected
     total hashes required for the whole chain: "+
    blockChain.getTotalExpectedHashes());
373                         System.out.println("Nonce
    for the most recent block: "+ blockChain.
```

```
getLatestBlock().getNonce());
                        System.out.println("Chain hash: "+ blockChain.getChainHash());
                        break;
                        //Adding a transaction to the block, asking the user for the difficulty and the transaction data
                        //For adding a block with 2 difficulty, the computation is fast, and it takes 0 milliseconds.
                        //For adding a block with difficulty of 3 it takes about 5 milliseconds
                        //The block with difficulty of 4 takes about 173 milliseconds
                        //The block with difficulty of 5 took 232 milliseconds
                        //The block with difficulty of 6 took 68888 milliseconds, we can see an increase in computation of time
                        // And for difficulty 7 it took 758503 milliseconds, indicating that as we increase the difficulty, the computation time increases exponentially
                case 1: System.out.println("Enter difficulty > 0");
                        int difficulty = Integer.parseInt(input.nextLine());
                        System.out.println("Enter transaction");
                        String data = input.nextLine();
                        Timestamp startTime = blockChain.getTime();
                        Block block1 = new Block(blockChain.getChainSize(), blockChain.getTime(), data, difficulty);
                        block1.setPreviousHash(blockChain.getChainHash());
                        try {
                            block1.proofOfWork();
                            blockChain.addBlock(
```

```
392 block1);
393                         } catch (
    NoSuchAlgorithmException e) {
394                             e.printStackTrace();
395                         }
396                         Timestamp endTime =
    blockChain.getTime();
397                         System.out.println("Total
    execution time to add this block was " + (endTime.
    getTime() - startTime.getTime()) + " milliseconds");
398                         break;
399                         //If the chain is valid then
     the time taken by the isChainValid function to
    verify is 0 milliseconds
400                         //The chain verification if
    the chain is invalid also takes 0 milliseconds,
    because as soon as the
401                         //chain finds a failure
    point it returns false and let's us know the point
    of failure.
402                         //The computation time is
    the same
403                 case 2: System.out.println("
    Verifying the entire chain");
404                         Timestamp startTimeValid =
    blockChain.getTime();
405                         try {
406                             boolean isValid =
    blockChain.isChainValid();
407                             if(blockChain.
    getErrorMessage().equals("")) {
408                                 System.out.println("
    Chain verification: " + isValid);
409                             }
410                             else
411                             {
412                                 System.out.println(
    blockChain.getErrorMessage());
413                                 System.out.println("
    Chain verification: " + isValid);
414                                 blockChain.
```

```
414 setErrorMessage("");
415                             }
416                         } catch (
    NoSuchAlgorithmException e) {
417                             e.printStackTrace();
418                         }
419                         Timestamp endTimeValid =
    blockChain.getTime();
420                         System.out.println("Total
    execution time to verify the chain was " + (
    endTimeValid.getTime() - startTimeValid.getTime
    ()) + " milliseconds");
421                         break;
422                         //This case is used to print
     the entire blockchain in the json format
423                 case 3: System.out.println("View the
     blockchain");
424                         System.out.println(
    blockChain);
425                         break;
426                         //The case helps in
    corrupting the data by asking the user the block
    that it wants to corrupt and
427                         //the new data for that
    block
428                 case 4: System.out.println("Corrupt
    the Blockchain");
429                         System.out.println("Enter
    the block ID of block to corrupt");
430                         int id = Integer.parseInt(
    input.nextLine());
431                         System.out.println("Enter
    new data for block " + id);
432                         String corruptData = input.
    nextLine();
433                         blockChain.getBlock(id).
    setData(corruptData);
434                         System.out.println("Block "
    + id +" now holds " + corruptData);
435                         break;
436                         //Repairing the block of the
```

```
436  chain with the difficulty of 3 when the data is corrupt takes about 248 milliseconds
437                          //It takes more time to repair the chain if the difficulty is greater, for difficulty 5 it takes about 2142 milliseconds
438                          //This indicates that the repair time depends on the block's difficulty and time taken to calculate the proof of work again.
439                          //The time taken also depends on the number of blocks that are corrupt and  need repairing
440                  case 5: System.out.println("Repairing the entire chain");
441                          Timestamp startTimeRepair = blockChain.getTime();
442                          try {
443                              blockChain.repairChain();
444                          } catch (NoSuchAlgorithmException e) {
445                              e.printStackTrace();
446                          }
447                          Timestamp endTimeRepair = blockChain.getTime();
448                          System.out.println("Total execution time required to repair the chain was " + (endTimeRepair.getTime() - startTimeRepair.getTime()) + " milliseconds");
449                          break;
450                          //Exiting the system if it is case 6
451                  case 6: System.exit(0);
452                          break;
453              }
454          }
455      }
456
457
458 }
459
```

```
C:\Users\kanis\.jdks\openjdk-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=64068:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\kanis\IdeaProjects\DistributedSystes\Project3\Project3Task1\target\classes;C:\Users\kanis\.m2\repository\org\json\json\20190722\json-20190722.jar blockchaintask.BlockChainClient

0. View basic blockchain status
1. Add a transaction to 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 the chain: 1
Difficulty of the most recent block: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 2958579
Expected total hashes required for the whole chain: 256.0
Nonce for the most recent block: 649
Chain hash: 007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53DFC62A1C8AFB

0. View basic blockchain status
1. Add a transaction to 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
3
Enter transaction
```

```
Kanishka pays Bob 100 dscoin
Total execution time to add this block was 0
   milliseconds

0. View basic blockchain status
1. Add a transaction to 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
3
Enter transaction
Bob pays Kanishka 20 dscoin
Total execution time to add this block was 48
   milliseconds

0. View basic blockchain status
1. Add a transaction to 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
Kanishka pays Carol 23 dscoin
Total execution time to add this block was 16
   milliseconds

0. View basic blockchain status
1. Add a transaction to 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
5
Enter transaction
Donna pays Kanishka 34 dscoin
Total execution time to add this block was 1267
   milliseconds

0. View basic blockchain status
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
3
{"ds_chain{" : [ {"index" : "0","time stamp" : "2021
   -10-23 15:38:58.645", "Tx" : "Genesis", "PrevHash
   " : "", "nonce" : "649", "difficulty" : "2"}
{"index" : "1","time stamp" : "2021-10-23 15:39:20.
   962", "Tx" : "Kanishka pays Bob 100 dscoin", "
   PrevHash" : "
   007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53
   DFC62A1C8AFB", "nonce" : "26", "difficulty" : "3"}
{"index" : "2","time stamp" : "2021-10-23 15:39:33.
   61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
   " : "
   0008088409F83872BE1DC83EDDB158D61567C016EE2EBD820D9C
   377359E8DB35", "nonce" : "11845", "difficulty" : "3
   "}
{"index" : "3","time stamp" : "2021-10-23 15:39:46.
   412", "Tx" : "Kanishka pays Carol 23 dscoin", "
   PrevHash" : "
   0005869905F8BB8305CF90B7EC686EAF520C826514702B52200C
   318AE3DDE667", "nonce" : "13011", "difficulty" : "4
   "}
{"index" : "4","time stamp" : "2021-10-23 15:40:02.
   041", "Tx" : "Donna pays Kanishka 34 dscoin", "
   PrevHash" : "
   00004111A2F975C575636BD1E25EFA0E73C357CDA5806E6B7F76
   E79D2CC4A628", "nonce" : "962195", "difficulty" : "5
```

```
 87 "}
 88 ],"chainHash":"
    000002C72FE12FC4BF8BFD4A5A6D356E17D9888C375608B0F886
    C8A494F50F76"}
 89
 90 0. View basic blockchain status
 91 1. Add a transaction to blockchain.
 92 2. Verify the blockchain.
 93 3. View the blockchain.
 94 4. Corrupt the chain.
 95 5. Hide the corruption by repairing the chain
 96 6. Exit
 97 1
 98 Enter difficulty > 0
 99 2
100 Enter transaction
101 Carol pays Donna 1 dscoin
102 Total execution time to add this block was 0
    milliseconds
103
104 0. View basic blockchain status
105 1. Add a transaction to blockchain.
106 2. Verify the blockchain.
107 3. View the blockchain.
108 4. Corrupt the chain.
109 5. Hide the corruption by repairing the chain
110 6. Exit
111 3
112 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Genesis", "PrevHash
    " : "", "nonce" : "649", "difficulty" : "2"}
113 {"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53
    DFC62A1C8AFB", "nonce" : "26", "difficulty" : "3"}
114 {"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
    " : "
    0008088409F83872BE1DC83EDDB158D61567C016EE2EBD820D9C
    377359E8DB35", "nonce" : "11845", "difficulty" : "3
```

```
114 "}
115 {"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    0005869905F8BB8305CF90B7EC686EAF520C826514702B52200C
    318AE3DDE667", "nonce" : "13011", "difficulty" : "4
    "}
116 {"index" : "4","time stamp" : "2021-10-23 15:40:02.
    041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    00004111A2F975C575636BD1E25EFA0E73C357CDA5806E6B7F76
    E79D2CC4A628", "nonce" : "962195", "difficulty" : "5
    "}
117 {"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    000002C72FE12FC4BF8BFD4A5A6D356E17D9888C375608B0F886
    C8A494F50F76", "nonce" : "40", "difficulty" : "2"}
118 ],"chainHash":"
    00918B22252F665C4D5DE79611FE9254A021F170CF7E7FA5B2A0
    B29B42B55C69"}
119
120 0. View basic blockchain status
121 1. Add a transaction to blockchain.
122 2. Verify the blockchain.
123 3. View the blockchain.
124 4. Corrupt the chain.
125 5. Hide the corruption by repairing the chain
126 6. Exit
127 4
128 Corrupt the BlockChain
129 Enter the block ID of block to corrupt
130 0
131 Enter new data for block 2
132 Carol pays Kanishka 1000 dscoin
133 Block 0 now holds Carol pays Kanishka 1000 dscoin
134
135 0. View basic blockchain status
136 1. Add a transaction to blockchain.
137 2. Verify the blockchain.
138 3. View the blockchain.
```

```
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
3
{"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "649", "
    difficulty" : "2"}
{"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53
    DFC62A1C8AFB", "nonce" : "26", "difficulty" : "3"}
{"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
    " : "
    0008088409F83872BE1DC83EDDB158D61567C016EE2EBD820D9C
    377359E8DB35", "nonce" : "11845", "difficulty" : "3
    "}
{"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    0005869905F8BB8305CF90B7EC686EAF520C826514702B52200C
    318AE3DDE667", "nonce" : "13011", "difficulty" : "4
    "}
{"index" : "4","time stamp" : "2021-10-23 15:40:02.
    041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    00004111A2F975C575636BD1E25EFA0E73C357CDA5806E6B7F76
    E79D2CC4A628", "nonce" : "962195", "difficulty" : "5
    "}
{"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    000002C72FE12FC4BF8BFD4A5A6D356E17D9888C375608B0F886
    C8A494F50F76", "nonce" : "40", "difficulty" : "2"}
],"chainHash":"
    00918B22252F665C4D5DE79611FE9254A021F170CF7E7FA5B2A0
    B29B42B55C69"}

0. View basic blockchain status
```

```
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
5
Repairing the entire chain
Total execution time required to repair the chain
    was 168 milliseconds

0. View basic blockchain status
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
3
{"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "71", "
    difficulty" : "2"}
{"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    008E55AC62F4A9441622104EACC5EFAD04468C3FCE28A952F586
    80094632AFD3", "nonce" : "11344", "difficulty" : "3
    "}
{"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
    " : "
    00049B7B15B6857EAA263704DF98628AE775F693FDFC753AC28D
    8364D2B20AB6", "nonce" : "5678", "difficulty" : "3"}
{"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    000D70DB72009755C06596525A52588C6A9C634BA4FF10E9D5D1
    47061CC13AA7", "nonce" : "76284", "difficulty" : "4
    "}
{"index" : "4","time stamp" : "2021-10-23 15:40:02.
```

```
041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    00007698FC8C9B2AD374A41035A8DFCE4E23479ACB4000F33DC7
    2F6193346BE7", "nonce" : "45039", "difficulty" : "5
    "}
{"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    00000634C741C47A369DD4A9F0E32A5109D0395B5AA64ECEF146
    982E74B57563", "nonce" : "159", "difficulty" : "2"}
],"chainHash":"
    0055ECB9F581BB4AF7648C4E7CA8E842D88E8905DE9B826D14B0
    933607D6F994"}

0. View basic blockchain status
1. Add a transaction to 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
Edward pays Kanishka 34 dscoin
Total execution time to add this block was 279
    milliseconds

0. View basic blockchain status
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
2
Verifying the entire chain
Chain verification: true
Total execution time to verify the chain was 0
    milliseconds
```

```
203
204 0. View basic blockchain status
205 1. Add a transaction to blockchain.
206 2. Verify the blockchain.
207 3. View the blockchain.
208 4. Corrupt the chain.
209 5. Hide the corruption by repairing the chain
210 6. Exit
211 4
212 Corrupt the BlockChain
213 Enter the block ID of block to corrupt
214 2
215 Enter new data for block 2
216 Frank pays Kanishka 3 dscoin
217 Block 2 now holds Frank pays Kanishka 3 dscoin
218
219 0. View basic blockchain status
220 1. Add a transaction to blockchain.
221 2. Verify the blockchain.
222 3. View the blockchain.
223 4. Corrupt the chain.
224 5. Hide the corruption by repairing the chain
225 6. Exit
226 2
227 Verifying the entire chain
228 Chain verification: false
229 ..Improper hash at node2.Does not begin with 000
230 Total execution time to verify the chain was 0
    milliseconds
231
232 0. View basic blockchain status
233 1. Add a transaction to blockchain.
234 2. Verify the blockchain.
235 3. View the blockchain.
236 4. Corrupt the chain.
237 5. Hide the corruption by repairing the chain
238 6. Exit
239 5
240 Repairing the entire chain
241 Total execution time required to repair the chain
    was 2712 milliseconds
```

```
242
243 0. View basic blockchain status
244 1. Add a transaction to blockchain.
245 2. Verify the blockchain.
246 3. View the blockchain.
247 4. Corrupt the chain.
248 5. Hide the corruption by repairing the chain
249 6. Exit
250 3
251 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "71", "
    difficulty" : "2"}
252 {"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    008E55AC62F4A9441622104EACC5EFAD04468C3FCE28A952F586
    80094632AFD3", "nonce" : "11344", "difficulty" : "3
    "}
253 {"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Frank pays Kanishka 3 dscoin", "
    PrevHash" : "
    00049B7B15B6857EAA263704DF98628AE775F693FDFC753AC28D
    8364D2B20AB6", "nonce" : "7285", "difficulty" : "3"}
254 {"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    0000EB4557E9BAB0BC569DA7A31D1DB97B2664A3D94FE448D62A
    5CFF410BBEC5", "nonce" : "2456", "difficulty" : "4"}
255 {"index" : "4","time stamp" : "2021-10-23 15:40:02.
    041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    0000813AF2FF526ACFB395CE5748E69FD2C5050D6DABA4FC4557
    A0B80BCAAFBE", "nonce" : "2123455", "difficulty" : "
    5"}
256 {"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    00000E08912AE327BDF2D674AF0279BCAC9A65DACBA3F59DB787
    DE70DAEB9443", "nonce" : "394", "difficulty" : "2"}
257 {"index" : "6","time stamp" : "2021-10-23 15:41:19.
```

```
034", "Tx" : "Edward pays Kanishka 34 dscoin", "
    PrevHash" : "
    000847C9B2F3285BF3DC6EC98023930C6FD11C80F3EA78BE3ED2
    98FEF78FFE23", "nonce" : "47600", "difficulty" : "4
    "}
],"chainHash":"
    0000E540167C91E8FDE3150CF99E9A39D8797230F42135922702
    818557507753"}

0. View basic blockchain status
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
2
Verifying the entire chain
Chain verification: true
Total execution time to verify the chain was 0
    milliseconds

0. View basic blockchain status
1. Add a transaction to 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 the chain: 7
Difficulty of the most recent block: 4
Total difficulty for all blocks: 23
Approximate hashes per second on this machine:
    2958579
Expected total hashes required for the whole chain:
    1188352.0
Nonce for the most recent block: 47600
Chain hash:
    0000E540167C91E8FDE3150CF99E9A39D8797230F42135922702
    818557507753
```

Task 1 Execution

```
287
288 0. View basic blockchain status
289 1. Add a transaction to blockchain.
290 2. Verify the blockchain.
291 3. View the blockchain.
292 4. Corrupt the chain.
293 5. Hide the corruption by repairing the chain
294 6. Exit
295 6
296
297 Process finished with exit code 0
298
```

```
C:\Users\kanis\.jdks\openjdk-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=64068:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\kanis\IdeaProjects\DistributedSystes\Project3\Project3Task1\target\classes;C:\Users\kanis\.m2\repository\org\json\json\20190722\json-20190722.jar blockchaintask.BlockChainClient

0. View basic blockchain status
1. Add a transaction to 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 the chain: 1
Difficulty of the most recent block: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 2958579
Expected total hashes required for the whole chain: 256.0
Nonce for the most recent block: 649
Chain hash: 007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53DFC62A1C8AFB

0. View basic blockchain status
1. Add a transaction to 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
3
Enter transaction
```

```
30 Kanishka pays Bob 100 dscoin
31 Total execution time to add this block was 0
   milliseconds
32
33 0. View basic blockchain status
34 1. Add a transaction to blockchain.
35 2. Verify the blockchain.
36 3. View the blockchain.
37 4. Corrupt the chain.
38 5. Hide the corruption by repairing the chain
39 6. Exit
40 1
41 Enter difficulty > 0
42 3
43 Enter transaction
44 Bob pays Kanishka 20 dscoin
45 Total execution time to add this block was 48
   milliseconds
46
47 0. View basic blockchain status
48 1. Add a transaction to blockchain.
49 2. Verify the blockchain.
50 3. View the blockchain.
51 4. Corrupt the chain.
52 5. Hide the corruption by repairing the chain
53 6. Exit
54 1
55 Enter difficulty > 0
56 4
57 Enter transaction
58 Kanishka pays Carol 23 dscoin
59 Total execution time to add this block was 16
   milliseconds
60
61 0. View basic blockchain status
62 1. Add a transaction to blockchain.
63 2. Verify the blockchain.
64 3. View the blockchain.
65 4. Corrupt the chain.
66 5. Hide the corruption by repairing the chain
67 6. Exit
```

```
68 1
69 Enter difficulty > 0
70 5
71 Enter transaction
72 Donna pays Kanishka 34 dscoin
73 Total execution time to add this block was 1267
   milliseconds
74
75 0. View basic blockchain status
76 1. Add a transaction to blockchain.
77 2. Verify the blockchain.
78 3. View the blockchain.
79 4. Corrupt the chain.
80 5. Hide the corruption by repairing the chain
81 6. Exit
82 3
83 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
   -10-23 15:38:58.645", "Tx" : "Genesis", "PrevHash
   " : "", "nonce" : "649", "difficulty" : "2"}
84 {"index" : "1","time stamp" : "2021-10-23 15:39:20.
   962", "Tx" : "Kanishka pays Bob 100 dscoin", "
   PrevHash" : "
   007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53
   DFC62A1C8AFB", "nonce" : "26", "difficulty" : "3"}
85 {"index" : "2","time stamp" : "2021-10-23 15:39:33.
   61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
   " : "
   0008088409F83872BE1DC83EDDB158D61567C016EE2EBD820D9C
   377359E8DB35", "nonce" : "11845", "difficulty" : "3
   "}
86 {"index" : "3","time stamp" : "2021-10-23 15:39:46.
   412", "Tx" : "Kanishka pays Carol 23 dscoin", "
   PrevHash" : "
   0005869905F8BB8305CF90B7EC686EAF520C826514702B52200C
   318AE3DDE667", "nonce" : "13011", "difficulty" : "4
   "}
87 {"index" : "4","time stamp" : "2021-10-23 15:40:02.
   041", "Tx" : "Donna pays Kanishka 34 dscoin", "
   PrevHash" : "
   00004111A2F975C575636BD1E25EFA0E73C357CDA5806E6B7F76
   E79D2CC4A628", "nonce" : "962195", "difficulty" : "5
```

```
 87 "}
 88 ],"chainHash":"
    000002C72FE12FC4BF8BFD4A5A6D356E17D9888C375608B0F886
    C8A494F50F76"}
 89
 90 0. View basic blockchain status
 91 1. Add a transaction to blockchain.
 92 2. Verify the blockchain.
 93 3. View the blockchain.
 94 4. Corrupt the chain.
 95 5. Hide the corruption by repairing the chain
 96 6. Exit
 97 1
 98 Enter difficulty > 0
 99 2
100 Enter transaction
101 Carol pays Donna 1 dscoin
102 Total execution time to add this block was 0
    milliseconds
103
104 0. View basic blockchain status
105 1. Add a transaction to blockchain.
106 2. Verify the blockchain.
107 3. View the blockchain.
108 4. Corrupt the chain.
109 5. Hide the corruption by repairing the chain
110 6. Exit
111 3
112 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Genesis", "PrevHash
    " : "", "nonce" : "649", "difficulty" : "2"}
113 {"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53
    DFC62A1C8AFB", "nonce" : "26", "difficulty" : "3"}
114 {"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
    " : "
    0008088409F83872BE1DC83EDDB158D61567C016EE2EBD820D9C
    377359E8DB35", "nonce" : "11845", "difficulty" : "3
```

```
114 "}
115 {"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    0005869905F8BB8305CF90B7EC686EAF520C826514702B52200C
    318AE3DDE667", "nonce" : "13011", "difficulty" : "4
    "}
116 {"index" : "4","time stamp" : "2021-10-23 15:40:02.
    041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    00004111A2F975C575636BD1E25EFA0E73C357CDA5806E6B7F76
    E79D2CC4A628", "nonce" : "962195", "difficulty" : "5
    "}
117 {"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    000002C72FE12FC4BF8BFD4A5A6D356E17D9888C375608B0F886
    C8A494F50F76", "nonce" : "40", "difficulty" : "2"}
118 ],"chainHash":"
    00918B22252F665C4D5DE79611FE9254A021F170CF7E7FA5B2A0
    B29B42B55C69"}
119
120 0. View basic blockchain status
121 1. Add a transaction to blockchain.
122 2. Verify the blockchain.
123 3. View the blockchain.
124 4. Corrupt the chain.
125 5. Hide the corruption by repairing the chain
126 6. Exit
127 4
128 Corrupt the BlockChain
129 Enter the block ID of block to corrupt
130 0
131 Enter new data for block 2
132 Carol pays Kanishka 1000 dscoin
133 Block 0 now holds Carol pays Kanishka 1000 dscoin
134
135 0. View basic blockchain status
136 1. Add a transaction to blockchain.
137 2. Verify the blockchain.
138 3. View the blockchain.
```

```
139 4. Corrupt the chain.
140 5. Hide the corruption by repairing the chain
141 6. Exit
142 3
143 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "649", "
    difficulty" : "2"}
144 {"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    007B359631B3DD848EF2D652B8B3E6FDFF18871367AB68270C53
    DFC62A1C8AFB", "nonce" : "26", "difficulty" : "3"}
145 {"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
    " : "
    0008088409F83872BE1DC83EDDB158D61567C016EE2EBD820D9C
    377359E8DB35", "nonce" : "11845", "difficulty" : "3
    "}
146 {"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    0005869905F8BB8305CF90B7EC686EAF520C826514702B52200C
    318AE3DDE667", "nonce" : "13011", "difficulty" : "4
    "}
147 {"index" : "4","time stamp" : "2021-10-23 15:40:02.
    041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    00004111A2F975C575636BD1E25EFA0E73C357CDA5806E6B7F76
    E79D2CC4A628", "nonce" : "962195", "difficulty" : "5
    "}
148 {"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    000002C72FE12FC4BF8BFD4A5A6D356E17D9888C375608B0F886
    C8A494F50F76", "nonce" : "40", "difficulty" : "2"}
149 ],"chainHash":"
    00918B22252F665C4D5DE79611FE9254A021F170CF7E7FA5B2A0
    B29B42B55C69"}
150
151 0. View basic blockchain status
```

```
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
5
Repairing the entire chain
Total execution time required to repair the chain
    was 168 milliseconds

0. View basic blockchain status
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
3
{"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "71", "
    difficulty" : "2"}
{"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    008E55AC62F4A9441622104EACC5EFAD04468C3FCE28A952F586
    80094632AFD3", "nonce" : "11344", "difficulty" : "3
    "}
{"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Bob pays Kanishka 20 dscoin", "PrevHash
    " : "
    00049B7B15B6857EAA263704DF98628AE775F693FDFC753AC28D
    8364D2B20AB6", "nonce" : "5678", "difficulty" : "3"}
{"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    000D70DB72009755C06596525A52588C6A9C634BA4FF10E9D5D1
    47061CC13AA7", "nonce" : "76284", "difficulty" : "4
    "}
{"index" : "4","time stamp" : "2021-10-23 15:40:02.
```

```
041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    00007698FC8C9B2AD374A41035A8DFCE4E23479ACB4000F33DC7
    2F6193346BE7", "nonce" : "45039", "difficulty" : "5
    "}
{"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    00000634C741C47A369DD4A9F0E32A5109D0395B5AA64ECEF146
    982E74B57563", "nonce" : "159", "difficulty" : "2"}
],"chainHash":"
    0055ECB9F581BB4AF7648C4E7CA8E842D88E8905DE9B826D14B0
    933607D6F994"}

0. View basic blockchain status
1. Add a transaction to 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
Edward pays Kanishka 34 dscoin
Total execution time to add this block was 279
    milliseconds

0. View basic blockchain status
1. Add a transaction to blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain
6. Exit
2
Verifying the entire chain
Chain verification: true
Total execution time to verify the chain was 0
    milliseconds
```

```
203
204 0. View basic blockchain status
205 1. Add a transaction to blockchain.
206 2. Verify the blockchain.
207 3. View the blockchain.
208 4. Corrupt the chain.
209 5. Hide the corruption by repairing the chain
210 6. Exit
211 4
212 Corrupt the BlockChain
213 Enter the block ID of block to corrupt
214 2
215 Enter new data for block 2
216 Frank pays Kanishka 3 dscoin
217 Block 2 now holds Frank pays Kanishka 3 dscoin
218
219 0. View basic blockchain status
220 1. Add a transaction to blockchain.
221 2. Verify the blockchain.
222 3. View the blockchain.
223 4. Corrupt the chain.
224 5. Hide the corruption by repairing the chain
225 6. Exit
226 2
227 Verifying the entire chain
228 Chain verification: false
229 ..Improper hash at node2.Does not begin with 000
230 Total execution time to verify the chain was 0
    milliseconds
231
232 0. View basic blockchain status
233 1. Add a transaction to blockchain.
234 2. Verify the blockchain.
235 3. View the blockchain.
236 4. Corrupt the chain.
237 5. Hide the corruption by repairing the chain
238 6. Exit
239 5
240 Repairing the entire chain
241 Total execution time required to repair the chain
    was 2712 milliseconds
```

```
242
243 0. View basic blockchain status
244 1. Add a transaction to blockchain.
245 2. Verify the blockchain.
246 3. View the blockchain.
247 4. Corrupt the chain.
248 5. Hide the corruption by repairing the chain
249 6. Exit
250 3
251 {"ds_chain{" : [ {"index" : "0","time stamp" : "2021
    -10-23 15:38:58.645", "Tx" : "Carol pays Kanishka
    1000 dscoin", "PrevHash" : "", "nonce" : "71", "
    difficulty" : "2"}
252 {"index" : "1","time stamp" : "2021-10-23 15:39:20.
    962", "Tx" : "Kanishka pays Bob 100 dscoin", "
    PrevHash" : "
    008E55AC62F4A9441622104EACC5EFAD04468C3FCE28A952F586
    80094632AFD3", "nonce" : "11344", "difficulty" : "3
    "}
253 {"index" : "2","time stamp" : "2021-10-23 15:39:33.
    61", "Tx" : "Frank pays Kanishka 3 dscoin", "
    PrevHash" : "
    00049B7B15B6857EAA263704DF98628AE775F693FDFC753AC28D
    8364D2B20AB6", "nonce" : "7285", "difficulty" : "3"}
254 {"index" : "3","time stamp" : "2021-10-23 15:39:46.
    412", "Tx" : "Kanishka pays Carol 23 dscoin", "
    PrevHash" : "
    0000EB4557E9BAB0BC569DA7A31D1DB97B2664A3D94FE448D62A
    5CFF410BBEC5", "nonce" : "2456", "difficulty" : "4"}
255 {"index" : "4","time stamp" : "2021-10-23 15:40:02.
    041", "Tx" : "Donna pays Kanishka 34 dscoin", "
    PrevHash" : "
    0000813AF2FF526ACFB395CE5748E69FD2C5050D6DABA4FC4557
    A0B80BCAAFBE", "nonce" : "2123455", "difficulty" : "
    5"}
256 {"index" : "5","time stamp" : "2021-10-23 15:40:22.
    322", "Tx" : "Carol pays Donna 1 dscoin", "PrevHash
    " : "
    00000E08912AE327BDF2D674AF0279BCAC9A65DACBA3F59DB787
    DE70DAEB9443", "nonce" : "394", "difficulty" : "2"}
257 {"index" : "6","time stamp" : "2021-10-23 15:41:19.
```

```
257 034", "Tx" : "Edward pays Kanishka 34 dscoin", "
    PrevHash" : "
    000847C9B2F3285BF3DC6EC98023930C6FD11C80F3EA78BE3ED2
    98FEF78FFE23", "nonce" : "47600", "difficulty" : "4
    "}
258 ],"chainHash":"
    0000E540167C91E8FDE3150CF99E9A39D8797230F42135922702
    818557507753"}
259
260 0. View basic blockchain status
261 1. Add a transaction to blockchain.
262 2. Verify the blockchain.
263 3. View the blockchain.
264 4. Corrupt the chain.
265 5. Hide the corruption by repairing the chain
266 6. Exit
267 2
268 Verifying the entire chain
269 Chain verification: true
270 Total execution time to verify the chain was 0
    milliseconds
271
272 0. View basic blockchain status
273 1. Add a transaction to blockchain.
274 2. Verify the blockchain.
275 3. View the blockchain.
276 4. Corrupt the chain.
277 5. Hide the corruption by repairing the chain
278 6. Exit
279 0
280 Current size of the chain: 7
281 Difficulty of the most recent block: 4
282 Total difficulty for all blocks: 23
283 Approximate hashes per second on this machine:
    2958579
284 Expected total hashes required for the whole chain:
    1188352.0
285 Nonce for the most recent block: 47600
286 Chain hash:
    0000E540167C91E8FDE3150CF99E9A39D8797230F42135922702
    818557507753
```

```
287
288 0. View basic blockchain status
289 1. Add a transaction to blockchain.
290 2. Verify the blockchain.
291 3. View the blockchain.
292 4. Corrupt the chain.
293 5. Hide the corruption by repairing the chain
294 6. Exit
295 6
296
297 Process finished with exit code 0
298
```

```
 1 package blockchaintask;
 2
 3 import org.json.JSONObject;
 4
 5 import java.io.BufferedWriter;
 6 import java.io.IOException;
 7 import java.io.OutputStreamWriter;
 8 import java.io.PrintWriter;
 9 import java.net.ServerSocket;
10 import java.net.Socket;
11 import java.security.NoSuchAlgorithmException;
12 import java.sql.Timestamp;
13 import java.util.HashMap;
14 import java.util.Map;
15 import java.util.NoSuchElementException;
16 import java.util.Scanner;
17
18 /**
19  * class BlockChainServer
20  * The server class that interacts with the
   blockchain to perform functions on the blockchain
   provided by the client
21  */
22 public class BlockChainServer {
23
24     /**
25      * function chain
26      * The chain function is used to perform various
   operations on the blockchain
27      * @param in the client input
28      * @param out output to client
29      * @param blockChain the blockchain
30      */
31     public static void chain(Scanner in, PrintWriter
   out, BlockChain blockChain) {
32         //If the size of the blocks is 0 then add the
    genesis block
33         if(blockChain.blocks.size() == 0) {
34             //creating a block and its previous hash
35             Block block = new Block(0, blockChain.
   getTime(), "Genesis", 2);
```

```
36              block.setPreviousHash("");
37              try {
38                  //doing the proof of work of the
   block
39                  block.proofOfWork();
40                  //adding the block in the array list
41                  blockChain.blocks.add(block);
42                  blockChain.computeHashesPerSecond();
   //computing the hashes per second
43                  blockChain.setChainHash(block.
   calculateHash()); //setting the chain hash
44              } catch (NoSuchAlgorithmException e) {
45                  e.printStackTrace();
46              }
47          }
48          //Json object to get the data from the client
49          JSONObject json = new JSONObject(in.nextLine
   ());
50          //Checking the choice sent by the client and
   performing functions based on that
51          int choice = Integer.parseInt(json.getString(
   "choice"));
52          JSONObject jsonResponse;
53          Map<String, String> responseMap = new HashMap
   <>();
54          switch(choice)
55          {   //Sending the data of the blockchain. The
    chain size, difficulty of the latest block, total
   difficulty
56              //the hashes per second computed above,
   total expected hashes by computing expected hashes of
    each block
57              //Nonce of the latest block and the
   chainHash i.e the hash of the latest block
58              case 0: responseMap = new HashMap<>();
59                  responseMap.put("chainSize", String.
   valueOf(blockChain.getChainSize()));
60                  responseMap.put("difficulty", String.
   valueOf(blockChain.getLatestBlock().getDifficulty
   ()));
61                  responseMap.put("totalDifficulty",
```

```
61 String.valueOf(blockChain.getTotalDifficulty()));
62                 responseMap.put("hashesPerSecond",
   String.valueOf(blockChain.getHashesPerSecond()));
63                 responseMap.put("totalExpectedHashes
   ", String.valueOf(blockChain.getTotalExpectedHashes
   ()));
64                 responseMap.put("nonce", String.
   valueOf(blockChain.getLatestBlock().getNonce()));
65                 responseMap.put("chainHash",
   blockChain.getChainHash());
66                 break;
67             //Adding a transaction to the block,
   getting the difficulty and the transaction data from
    the client
68             case 1:
69                 int difficulty = Integer.parseInt(
   json.getString("difficulty"));
70                 String data = json.getString("data"
   );
71                 Timestamp startTime = blockChain.
   getTime();
72                 //creating the new block
73                 Block block1 = new Block(blockChain.
   getChainSize(), blockChain.getTime(), data,
   difficulty);
74                 block1.setPreviousHash(blockChain.
   getChainHash()); //setting the previous hash of the
   block
75                 try {
76                     block1.proofOfWork(); //proof of
    work for the block
77                     blockChain.addBlock(block1); //
   adding the block to the chain
78                 } catch (NoSuchAlgorithmException e
   ) {
79                     e.printStackTrace();
80                 }
81                 Timestamp endTime = blockChain.
   getTime();
82                 long totalTime = endTime.getTime
   () - startTime.getTime();
```

```
 83                 responseMap = new HashMap<>();
 84                 //Sending back the time taken to add
     the block
 85                 responseMap.put("time", String.
    valueOf(totalTime));
 86                 break;
 87                 //verifying the chain
 88             case 2: boolean isValid = false;
 89                 Timestamp startTimeValid =
    blockChain.getTime();
 90                 try {
 91                     isValid = blockChain.
    isChainValid();
 92                 } catch (NoSuchAlgorithmException e
    ) {
 93                     e.printStackTrace();
 94                 }
 95                 Timestamp endTimeValid = blockChain.
    getTime();
 96                 responseMap = new HashMap<>();
 97                 //sending true if the chain is valid
 98                 if(blockChain.getErrorMessage().
    equals("")) {
 99                     responseMap.put("isValid",
    String.valueOf(isValid));
100                     responseMap.put("time", String.
    valueOf(endTimeValid.getTime() - startTimeValid.
    getTime()));
101                 }
102                 else //sending an error message in
    json object if the chain is not valid
103                 {
104                     responseMap.put("isValid",
    String.valueOf(isValid));
105                     responseMap.put("errorMessage",
    blockChain.getErrorMessage());
106                     responseMap.put("time", String.
    valueOf(endTimeValid.getTime() - startTimeValid.
    getTime()));
107                     blockChain.setErrorMessage("");
108                 }
```

```
109                 break;
110                 //Sending the toString of the entire
     chain to the client
111             case 3: responseMap = new HashMap<>();
112                 responseMap.put("blockchain",
    blockChain.toString());
113                 break;
114                 //Corrupting the data of the block
    if given by the client
115             case 4: int id = Integer.parseInt(json.
    getString("id"));
116                 String corruptData = json.getString(
    "data");
117                 blockChain.getBlock(id).setData(
    corruptData); //adding the corrupt data
118                 responseMap = new HashMap<>();
119                 //sending a message that the data is
     corrupted for the id
120                 responseMap.put("id", String.valueOf
    (id));
121                 responseMap.put("corruptData",
    corruptData);
122                 break;
123                 //Repairing the blockchain
124             case 5: Timestamp startTimeRepair =
    blockChain.getTime();
125                 try {
126                     blockChain.repairChain();
127                 } catch (NoSuchAlgorithmException e
    ) {
128                     e.printStackTrace();
129                 }
130                 //sending the time taken to repair
    the blockchain
131                 Timestamp endTimeRepair = blockChain
    .getTime();
132                 responseMap = new HashMap<>();
133                 responseMap.put("time", String.
    valueOf(endTimeRepair.getTime() - startTimeRepair.
    getTime()));
134                 break;
```

```
135         }
136         //The maps created during the choice,
    sending them in the jsonResponse to the client
137         jsonResponse = new JSONObject(responseMap);
138         out.println(jsonResponse);
139         out.flush();
140     }
141 
142     /**
143      * function main
144      * The function is used to connect with the
    client. The server socket listens to the client.
145      * @param args
146      */
147     public static void main(String args[]) {
148         Socket clientSocket = null;
149         BlockChain blockChain = new BlockChain();
150         try {
151             int serverPort = 7777; // the server
    port we are using
152 
153             // Create a new server socket
154             ServerSocket listenSocket = new
    ServerSocket(serverPort);
155             while (true) {
156                 clientSocket = listenSocket.accept
    ();
157                 //"in" to read from the client
    socket
158                 Scanner in = new Scanner(
    clientSocket.getInputStream());
159                 //"out" to write to the client
    socket
160                 PrintWriter out;
161                 out = new PrintWriter(new
    BufferedWriter(new OutputStreamWriter(clientSocket.
    getOutputStream())));
162                 //verifying the message and
    returning the computation result
163                 chain(in, out, blockChain);
164             }
```

```
165
166             // Handle exceptions
167         } catch (IOException e) { //Checking for
    input output exceptions
168         } catch (NoSuchElementException e) {
169         } finally { //If socket is not null and
    request is done, close the socket
170             try {
171                 if (clientSocket != null) {
172                     clientSocket.close();
173                 }
174             } catch (IOException e) {
175                 // ignore exception on close
176             }
177         }
178     }
179 }
180
```