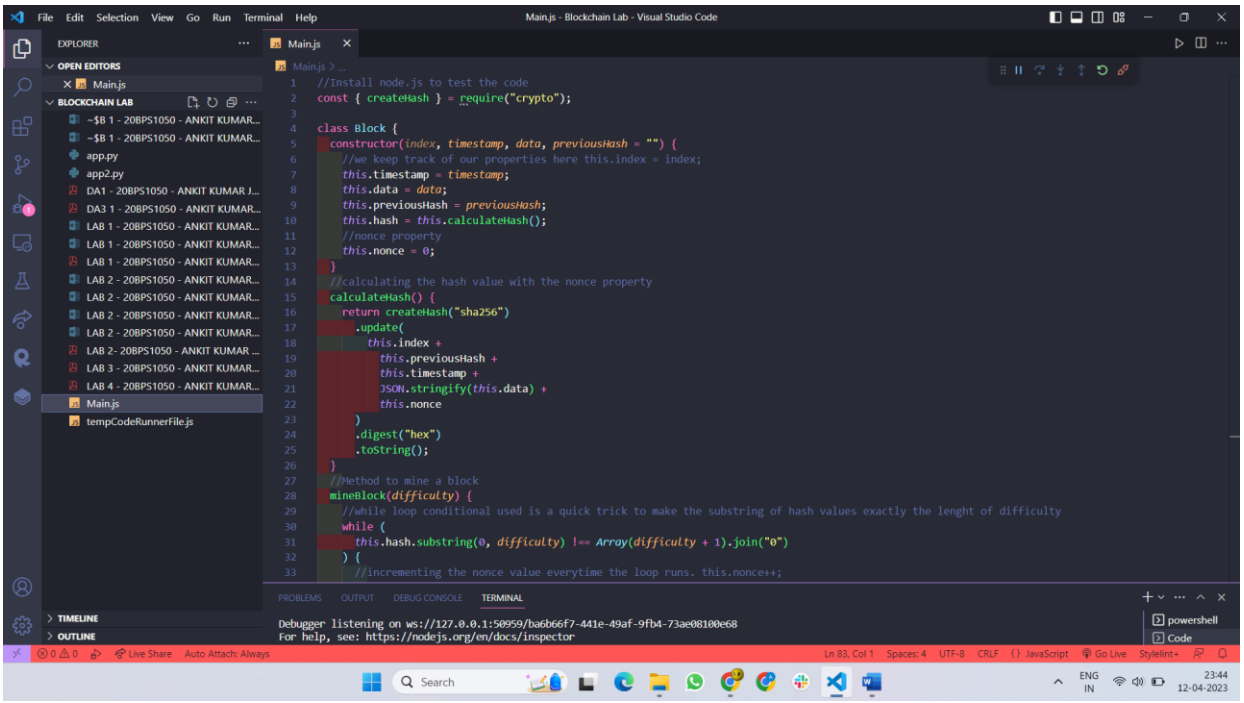
NAME: ANKIT KUMAR JHA
REG.NO.20BPS1050

EXERCISE 6
IMPLEMENT HASH PUZZLE



main.js

```
//Install node.js to test the code
const { createHash } = require("crypto");

class Block {
  constructor(index, timestamp, data, previousHash = "") {
    //we keep track of our properties here this.index = index;
    this.timestamp = timestamp;
    this.data = data;
    this.previousHash = previousHash;
    this.hash = this.calculateHash();
    //nonce property
    this.nonce = 0;
  }
  //calculating the hash value with the nonce property
  calculateHash() {
```

```
return createHash("sha256")
      .update(
        this.index +
          this.previousHash +
          this.timestamp +
          JSON.stringify(this.data) +
          this.nonce
      )
      .digest("hex")
      .toString();
  }
  //Method to mine a block
  mineBlock(difficulty) {
    //while loop conditional used is a quick trick to make the substring of hash values exactly the
lenght of difficulty
    while (
      this.hash.substring(0, difficulty) !== Array(difficulty + 1).join("0")
    ) {
      //incrementing the nonce value everytime the loop runs. this.nonce++;
      //recalculating the hash value
      this.hash = this.calculateHash();
    }
    //logging when a block is created
    console.log("Block mined: " + this.hash);
  }
}
class Blockchain {
  constructor() {
    this.chain = [this.createGenesisBlock()];
    //adding a difficulty property to the Blockchain class
    this.difficulty = 4;
  }
  createGenesisBlock() {
    return new Block(0, "02/01/2018", "Genesis Block", "0");
  }
  getlatestBlock() {
    return this.chain[this.chain.length - 1];
  }
  addBlock(newBlock) {
    newBlock.previousHash = this.getlatestBlock().hash;
    //We commented the earlier method that adds a block directly //newBlock.hash =
newBlock.calculateHash();
```

```
//New method to mine the block //Customizable difficulty value

    newBlock.mineBlock(this.difficulty);
    this.chain.push(newBlock);
  }
  isChainValid() {
    for (let i = 1; i < this.chain.length; i++) {
      const currentBlock = this.chain[i];
      const previousBlock = this.chain[i - 1];
      if (currentBlock.hash !== currentBlock.calculateHash()) {
        return false;
      } //check for hash calculations
      if (currentBlock.previousHash !== previousBlock.hash) {
        return false;
      } //check whether current block points to the correct previous block
    }
    return true;
  }
}
let koreCoin = new Blockchain();
console.log("Mining block 1...");
koreCoin.addBlock(new Block(1, "01/01/2018", { amount: 20 }));
console.log("Mining block 2...");

koreCoin.addBlock(new Block(2, "02/01/2018", { amount: 40 }));
console.log("Mining block 3...");
koreCoin.addBlock(new Block(3, "02/01/2018", { amount: 40 }));
```

Output-

```
Mining block 1...
Block mined: 0000023168f87d968813b22c4dc92f60c127ff5084af8487d913d497ea7a7900
Mining block 2...
Block mined: 00008e0c291aaf728e015855328b14e651231cce209e6413503fd299e0df6c5e
Mining block 3...
Block mined: 0000fb4a126ef4c1c3c93bf2ed25e8db4c7da2ec89a46aad4f7bf092afd8b6b4
```

Changing difficulty to 7

```javascript
//Observe the zeros above-5 zeros and hence met the target of 4 //increase difficulty level to 7
const { createHash } = require("crypto");
class Block {
  constructor(index, timestamp, data, previousHash = "") {
    //we keep track of our properties here
    this.index = index;
    this.timestamp = timestamp;
    this.data = data;
    this.previousHash = previousHash;
    this.hash = this.calculateHash();
    //nonce property
    this.nonce = 0;
  }
  //calculating the hash value with the nonce property
  calculateHash() {
    return createHash("sha256")
      .update(
        this.index +
          this.previousHash +
          this.timestamp +
          JSON.stringify(this.data) +
          this.nonce
      )
      .digest("hex")
      .toString();
  }
  //Method to mine a block
  mineBlock(difficulty) {
    //while loop conditional used is a quick trick to make the substring of hash values exactly the lenght of difficulty
    while (
      this.hash.substring(0, difficulty) !== Array(difficulty + 1).join("0")
    ) {
      //incrementing the nonce value everytime the loop runs. this.nonce++;
      //recalculating the hash value
      this.hash = this.calculateHash();
    }
    //logging when a block is created
```

```javascript
  createGenesisBlock() {
    return new Block(0, "02/01/2018", "Genesis Block", "0");
  }

  getlatestBlock() {
    return this.chain[this.chain.length - 1];
  }
  addBlock(newBlock) {
    newBlock.previousHash = this.getlatestBlock().hash;
    //We commented the earlier method that adds a block directly //newBlock.hash = newBlock.calculateHash();
    //New method to mine the block //Customizable difficulty value newBlock.mineBlock( this.difficulty );
    this.chain.push(newBlock);
  }
  isChainValid() {
    for (let i = 1; i < this.chain.length; i++) {
      const currentBlock = this.chain[i];
      const previousBlock = this.chain[i - 1];
      if (currentBlock.hash !== currentBlock.calculateHash()) {
        return false;
      } //check for hash calculations
      if (currentBlock.previousHash !== previousBlock.hash) {
        return false;
      } //check whether current block points to the correct previous block
    }

    return true;
  }
}
let koreCoin = new Blockchain();
console.log("Mining block 1...");
koreCoin.addBlock(new Block(1, "01/01/2018", { amount: 20 }));
console.log("Mining block 2...");
koreCoin.addBlock(new Block(2, "02/01/2018", { amount: 40 }));
console.log("Mining block 3...");
koreCoin.addBlock(new Block(3, "02/01/2018", { amount: 40 }));
```

main.js

```
//Observe the zeros above-5 zeros and hence met the target of 4 //increase difficulty level to 7
const { createHash } = require("crypto");
class Block {
  constructor(index, timestamp, data, previousHash = "") {
    //we keep track of our properties here
    this.index = index;
    this.timestamp = timestamp;
    this.data = data;
    this.previousHash = previousHash;
    this.hash = this.calculateHash();
    //nonce property
    this.nonce = 0;
  }
  //calculating the hash value with the nonce property
  calculateHash() {
    return createHash("sha256")
      .update(
        this.index +
          this.previousHash +
          this.timestamp +
          JSON.stringify(this.data) +
          this.nonce
      )
      .digest("hex")
      .toString();
  }
  //Method to mine a block
  mineBlock(difficulty) {
    //while loop conditional used is a quick trick to make the substring of hash values exactly the
lenght of difficulty
    while (
      this.hash.substring(0, difficulty) !== Array(difficulty + 1).join("0")
    ) {
      //incrementing the nonce value everytime the loop runs. this.nonce++;
      //recalculating the hash value
      this.hash = this.calculateHash();
    }
    //logging when a block is created
    console.log("Block mined: " + this.hash);
  }
```

```
}
class Blockchain {

constructor() {
   this.chain = [this.createGenesisBlock()];
   //adding a difficulty property to the Blockchain class
   this.difficulty = 7;
  }
  createGenesisBlock() {
   return new Block(0, "02/01/2018", "Genesis Block", "0");
  }

  getlatestBlock() {
   return this.chain[this.chain.length - 1];
  }
  addBlock(newBlock) {
   newBlock.previousHash = this.getlatestBlock().hash;
   //We commented the earlier method that adds a block directly //newBlock.hash =
newBlock.calculateHash();
   //New method to mine the block //Customizable difficulty value newBlock.mineBlock(
this.difficulty );
   this.chain.push(newBlock);
  }
  isChainValid() {
   for (let i = 1; i < this.chain.length; i++) {
     const currentBlock = this.chain[i];
     const previousBlock = this.chain[i - 1];
     if (currentBlock.hash !== currentBlock.calculateHash()) {
       return false;
     } //check for hash calculations
     if (currentBlock.previousHash !== previousBlock.hash) {
       return false;
     } //check whether current block points to the correct previous block
   }

   return true;
  }
}
let koreCoin = new Blockchain();
console.log("Mining block 1...");
koreCoin.addBlock(new Block(1, "01/01/2018", { amount: 20 }));
console.log("Mining block 2...");
```

```
koreCoin.addBlock(new Block(2, "02/01/2018", { amount: 40 }));


console.log("Mining block 3...");
koreCoin.addBlock(new Block(3, "02/01/2018", { amount: 40 }));
```

Output –

```
Mining block 1…
Block mined: 0000000099da284da2a3a454e75576632ab139495dfd9cb160080e534bbf90d5
Mining block 2…
Block mined: 00000007f6fb7d91f0a6ee2472aa6ab5d7fe73e04b0d2e5cfc6feca75c571cb2
Mining block 3…
Block mined: 000000086f9f621bd96e61faae30fb3f8f4da1e992f50a76d7c2d658d4d49ac6
```