FINOLEX ACADEMY OF MANAGEMENT AND TECHNOLOGY, RATNAGIRI

DEPARTMENT OF MCA

PRACTICAL NO. 02

Cryptocurrency

1.Write a program to create the chain with Genesis block and adding block into blockchain and validating the chain for any alteration. (Part-I) Ans :

- **Code –**

blockchain1.js

```
//javascript program to create the blockchain with genesis block and
//adding block into the blockchain and valiting the chain for any declaration part-I
//Cammand to install the crypto-js libtrary for sha256 -> npm install --save crypto-js
const SHA256=require('crypto-js/sha256');
class Block
{
   constructor(index, timestamp, data, previousHash='')
   {
      this.index=index;
      this.timestamp=timestamp;
      this.data=data;
      this.previousHash=previousHash;
      this.hash=this.calculateHash();
   }
   calculateHash()
   {
      return
SHA256(this.index+this.previousHash+this.timestamp+JSON.stringify(this.data)).toS
tring();
   }
}
class blockchain
{
   constructor()
   {
      this.chain=[this.createGenesisBlock()];
   }
   createGenesisBlock()
   {
      return new Block(0,"04/09/2023","Genesis Block","0");
   }
   getLatestBlock()
   {
      return this.chain[this.chain.length-1];
   }
```

```
    addBlock(newBlock)
    {
       newBlock.previousHash=this.getLatestBlock().hash;
       newBlock.hash=newBlock.calculateHash();
       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;
          }
          if(currentBlock.previousHash !== previousBlock.hash)
          {
             return false;
          }
       }
       return true;
    }
}
let tanuCoin = new blockchain();
console.log("Adding block....");
tanuCoin.addBlock(new Block(1,"05/09/2023",{amount:4000}));
tanuCoin.addBlock(new Block(2,"06/09/2023",{amount:5000}));
console.log(JSON.stringify(tanuCoin,null,4));
console.log('Is blockchain valid ? '+tanuCoin.isChainValid());
tanuCoin.chain[2].data={amount: 2000};
console.log('Is blockchain valid ? '+tanuCoin.isChainValid());
```

Output:

```
PS D:\BlockChain> cd Practicle_2
PS D:\BlockChain\Practicle_2> cd node_modules
PS D:\BlockChain\Practicle_2\node_modules> node blockChain1.js
Adding block....
{
    "chain": [
        {
            "index": 0,
            "timestamp": "04/09/2023",
            "data": "Genesis Block",
            "previousHash": "0",
            "hash": "32e4871c84f59b3b726ba548dfd417045141586ec5be0c5e4d6995ced69a4801"
        },
        {
            "index": 1,
            "timestamp": "05/09/2023",
            "data": {
                "amount": 4000
            },
            "previousHash": "32e4871c84f59b3b726ba548dfd417045141586ec5be0c5e4d6995ced69a4801",
            "hash": "343831705b9ef245467c8b354119c619f4e718b668bb7fbb5a725dc8360bb426"
        },
        {
            "index": 2,
            "timestamp": "06/09/2023",
            "data": {
                "amount": 5000
            },
            "previousHash": "343831705b9ef245467c8b354119c619f4e718b668bb7fbb5a725dc8360bb426",
            "hash": "1d2ce7e3fd171bfa97060beaffd73ca573e0c1426d7a92cad5e983fee6352bb6"
        }
    ]
}
Is blockchain valid ? true
Is blockchain valid ? false
PS D:\BlockChain\Practicle_2\node_modules>
```

2. Write a program to implementing proof of work for blockchain. (Part-II) Ans:

- **Code –**

blockchain2.js

```
const SHA256=require('crypto-js/sha256');
class Block
{
   constructor(index, timestamp, data, previousHash='')
   {
      this.index=index;
      this.timestamp=timestamp;
      this.data=data;
      this.nonce=0;
      this.previousHash=previousHash;
      this.hash=this.calculateHash();
   }
   calculateHash()
   {
      return
SHA256(this.index+this.previousHash+this.timestamp+JSON.stringify(this.data)+this.
nonce).toString();
   }
   mineBlock(difficulty)
   {
      while(this.hash.substring(0,difficulty) !== Array(difficulty+1).join("0"))
      {
         this.nonce++;
         this.hash =this.calculateHash();
      }
      console.log("Block mined: "+this.hash);
      console.log("Nonce : "+this.nonce);
   }
}
class blockchain
{
   constructor()
   {
      this.chain=[this.createGenesisBlock()];
      this.difficulty=6;
   }
   createGenesisBlock()
   {
      return new Block(0,"04/09/2023","Genesis Block","0");
   }
   getLatestBlock()
   {
      return this.chain[this.chain.length-1];
   }
   addBlock(newBlock)
   {
      newBlock.previousHash=this.getLatestBlock().hash;
```

```
        //newBlock.hash=newBlock.calculateHash();
        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;
            }
            if(currentBlock.previousHash !== previousBlock.hash)
            {
                return false;
            }
        }
        return true;
    }
}
let tanuCoin = new blockchain();
console.log("Mining block....");
tanuCoin.addBlock(new Block(1,"05/09/2023",{amount:4000}));
tanuCoin.addBlock(new Block(2,"06/09/2023",{amount:1000}));
```

Output:

```
PS D:\BlockChain\Practicle_2\node_modules> node blockchain2.js
Mining block....
Block mined: 000000c741f90fc6ee878b9bff73bf09f4eaa7b1369cca2b3ceb9e0522da2241
Nonce : 17405871
Block mined: 0000007558789fc34af6b16e71b582768f333ec96d25f9c55ecbcb91da3dce07
Nonce : 15466889
PS D:\BlockChain\Practicle_2\node_modules>
```

3. Write a program to add multiple transactions into block and give reward to miner for successful mining of block in blockchain. (Part-III) Ans:

- **Code –**

```
const SHA256 = require('crypto-js/sha256');//access a library and stored in one const
class Transaction{
    constructor(fromAddress,toAddress,amount)
    {
        this.fromAddress=fromAddress;
        this.toAddress=toAddress;
        this.amount=amount;
```

```
    }
}
class Block
{
    constructor(timestamp,transactions,previousHash ='')
    {
        this.timestamp=timestamp;
        this.transactions=transactions;
        this.previousHash=previousHash;
        this.nonce=0;
        this.hash=this.calculateHash();
    }
    calculateHash()
    {       return
SHA256(this.previousHash+this.timestamp+JSON.stringify(this.transactions)+this.nonce
).toString();
    }
    mineBlock(difficulty)//difficulty->how many zeros in combine hash
    {
        while(this.hash.substring(0,difficulty) !== Array(difficulty+1).join("0"))
        {
            this.nonce++;//increment nonce by 1
            this.hash=this.calculateHash(); //
        }
        console.log("Block Mined : " +this.hash);
        console.log("Nonce : "+this.nonce);
    }
}
class Blockchain
{
    constructor()
    {
        this.chain= [ this.createGenesisBlock()];
        this.difficulty=2;
        this.pendingTransactions=[];//create array-> array to store transations
        this.miningReward=100;
    }
    createGenesisBlock() //create a genesis block
    {
        return new Block("04/09/2023","Genesis Block","0");
    }
    getLatestBlock()
    {
        return this.chain[this.chain.length-1];
    }
    minePendingTransaction(miningRewardAddress)//methos is used the array
pendingTransactions
    {
        let block = new
Block(Date.now(),this.pendingTransactions,this.getLatestBlock().hash);
```

```
        block.mineBlock(this.difficulty);
        console.log("Block mined Successfully...!!");
        this.chain.push(block);
        this.pendingTransactions=[new
    Transaction(null,miningRewardAddress,this.miningReward)];

      }
      createTransaction(transactions)
      {
        this.pendingTransactions.push(transactions);
      }
      getBalanceOfAddress(address)
      {
        let balance=0;
        for(const block of this.chain)
        {
          for(const trans of block.transactions)
          {
            if(trans.fromAddress== address)
            {
              balance-=trans.amount;
            }
            if(trans.toAddress==address)
            {
              balance+=trans.amount;
            }
          }
        }
        return balance;
      }
      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;
          }
          if(currentBlock.previousHash !== previousBlock.hash)
          {
            return false;//chain is not valid so we return false
          }
        }
        return true;
      }
    }
    let tanuCoin = new Blockchain(); //default constructor
    tanuCoin.createTransaction(new Transaction('address1','address2',100));
```

```
tanuCoin.createTransaction(new Transaction('address2','address1',50));
console.log("Start Mining...");
tanuCoin.minePendingTransaction('Tata-Address');
console.log("Balance of Tata-Address : ",tanuCoin.getBalanceOfAddress('TataAddress'));
tanuCoin.minePendingTransaction('Tata-Address');
console.log("Balance of Tata-Address : ",tanuCoin.getBalanceOfAddress('TataAddress'));
console.log("Balance of Address1 : ",tanuCoin.getBalanceOfAddress('address1'));
console.log("Balance of Address2 : ",tanuCoin.getBalanceOfAddress('address2'));
```

Output:

```
PS D:\BlockChain\Practicle_2\node_modules> node blockchain4.js
Start Mining...
Block Mined : 00b1d674c3b5065f3d211e640aa8c8d74a3d6d458f7447b582a1f5bc698721f3
Nonce : 124
Block mined Successfully...!!
Balance of Tata-Address :  0
Block Mined : 00a57531dfb8fd74c4a5db8fa98e90bf7904921ed5aa19318637e6de0f3451ce
Nonce : 34
Block mined Successfully...!!
Balance of Tata-Address :  0
Balance of Address1 :  -50
Balance of Address2 :  50
PS D:\BlockChain\Practicle_2\node_modules>
```

4. Write a program to sign the transaction with private key and verify the signed transactions for blockchain. (Part-IV) Ans:

- **Code –**

```
const SHA256 = require('crypto-js/sha256');//access a library and stored in one const class
Transaction{
   constructor(fromAddress,toAddress,amount)
   {
      this.fromAddress=fromAddress;
this.toAddress=toAddress;
      this.amount=amount;
   }
}
class Block
{
   constructor(timestamp,transactions,previousHash ='')
   {
      this.timestamp=timestamp;
this.transactions=transactions;
this.previousHash=previousHash;        this.nonce=0;
      this.hash=this.calculateHash();
   }
   calculateHash()
   {
return
```

```
SHA256(this.previousHash+this.timestamp+JSON.stringify(this.transactions)+this.nonce).to
String();
   }
   mineBlock(difficulty)//difficulty->how many zeros in combine hash
   {
      while(this.hash.substring(0,difficulty) !== Array(difficulty+1).join("0"))
      {
         this.nonce++;//increment nonce by 1
         this.hash=this.calculateHash(); //
      }
      console.log("Block Mined : " +this.hash);
      console.log("Nonce : "+this.nonce);
   }
}
   class Blockchain
   {
      constructor()
      {
         this.chain= [ this.createGenesisBlock()];
         this.difficulty=2;
         this.pendingTransactions=[];//create array-> array to store transations

         this.miningReward=100;
      }
      createGenesisBlock() //create a genesis block
      {
         return new Block("04/09/2023","Genesis Block","0");
      }
      getLatestBlock()
      {
         return this.chain[this.chain.length-1];
      }
      minePendingTransaction(miningRewardAddress)//methos is used the array
pendingTransactions
      {
         let block = new
Block(Date.now(),this.pendingTransactions,this.getLatestBlock().hash);
         block.mineBlock(this.difficulty);
console.log("Block mined Successfully...!!");
         this.chain.push(block);
this.pendingTransactions=[new
Transaction(null,miningRewardAddress,this.miningReward)];
      }
      createTransaction(transactions)
      {
         this.pendingTransactions.push(transactions);
      }
```

```
    getBalanceOfAddress(address)
    {
      let balance=0;
      for(const block of this.chain)
      {
        for(const trans of block.transactions)
        {
          if(trans.fromAddress== address)
          {
            balance-=trans.amount;
          }
          if(trans.toAddress==address)
          {
            balance+=trans.amount;
          }
        }
      }
      return balance;
    }
    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;
        }
        if(currentBlock.previousHash !== previousBlock.hash)
        {
          return false;//chain is not valid so we return false
        }
}
      return true;
    }
  }
  let tanuCoin = new Blockchain(); //default constructor
tanuCoin.createTransaction(new Transaction('address1','address2',100));
tanuCoin.createTransaction(new Transaction('address2','address1',50));
  console.log("Start Mining...");
  tanuCoin.minePendingTransaction('Tata-Address');
  console.log("Balance of Tata-Address : ", tanuCoin.getBalanceOfAddress('TataAddress'));
  tanuCoin.minePendingTransaction('Tata-Address');
  console.log("Balance of Tata-Address : ", tanuCoin.getBalanceOfAddress('TataAddress'));
      console.log("Balance of Address1 : ", tanuCoin.getBalanceOfAddress('address1'));
    console.log("Balance of Address2 : ", tanuCoin.getBalanceOfAddress('address2'));
```

- **Output –**

```
PS E:\MCA_Sem_3\Blockchain\PracticalNo2> node blockchain3.js
Start Mining...
Block Mined : 008e75c1550d6e61e9274d7d1c27d07b292e8aacd7bb3b825e66c5ad2e758885
Nonce : 116
Block mined Successfully...!!
Balance of Tata-Address :  0
Block Mined : 00e342e2afe3451a340f4ca9ba0c813034a52f28c17605120cc0b07e065eff67
Nonce : 296
Block mined Successfully...!!
Balance of Tata-Address :  100
Balance of Address1 :  -50
Balance of Address2 :  50
PS E:\MCA_Sem_3\Blockchain\PracticalNo2>
```

4. Write a program to sign the transaction with private key and verify the signed transactions for blockchain. (Part-IV) Ans:

- **Code –**

1)keygenerator.js

```
const EC = require('elliptic').ec;//elliptic curve
const ec = new EC('secp256k1'); const
key = ec.genKeyPair(); const
publicKey = key.getPublic('hex');
const privateKey = key.getPrivate('hex');
console.log();
console.log("Private Key : ",privateKey); console.log();
console.log("Public Key : ",publicKey);
```

2)Blockchain.js

```
const SHA256 = require('crypto-js/sha256');//access a library and stored in one const const
EC = require('elliptic').ec;//elliptic curve
const ec = new EC('secp256k1'); class
Transaction{
   constructor(fromAddress,toAddress,amount)
   {
      this.fromAddress=fromAddress;
this.toAddress=toAddress;
      this.amount=amount;
   }
   calculateHash()
   {
      return SHA256(this.fromAddress + this.toAddress + this.amount).toString();
   }
   signTransaction(signingKey)
```

```
    {
        if(signingKey.getPublic('hex')!==this.fromAddress)
        {
            throw new Error('You can not sign the transactions for other wallets..!!');
        }
        const hashTx = this.calculateHash();
const sig = signingKey.sign(hashTx,'base64');
this.signature= sig.toDER('hex');
    }
    isValid()
    {
        if(this.fromAddress == null)
        return true;
        if(!this.signature || this.signature.length==0)
        {
            throw new Error('No signature in this transactions..!!');
        }       const
publicKey=ec.keyFromPublic(this.fromAddress,'hex');
return publicKey.verify(this.calculateHash(),this.signature);
    }
}
class Block
{
    constructor(timestamp,transactions,previousHash ='')
    {
        this.timestamp=timestamp;
this.transactions=transactions;
this.previousHash=previousHash;        this.nonce=0;
        this.hash=this.calculateHash();
    }
    calculateHash()
    {
return
SHA256(this.previousHash+this.timestamp+JSON.stringify(this.transactions)+this.nonce).to
String();
    }
    mineBlock(difficulty)//difficulty->how many zeros in combine hash
    {
        while(this.hash.substring(0,difficulty) !== Array(difficulty+1).join("0"))
        {
            this.nonce++;//increment nonce by 1
            this.hash=this.calculateHash();
        }
        console.log("Block Mined : " +this.hash);
        console.log("Nonce : "+this.nonce);
    }
    hasValidTransactions()
```

```
   {
      for(const tx of this.transactions)
      {
         if(!tx.isValid())
         {
            return false;
         }
}
      return true;
   }
}
 class Blockchain
 {
    constructor()
    {
        this.chain= [ this.createGenesisBlock()];
        this.difficulty=4;
        this.pendingTransactions=[];//create array-> array to store transations

        this.miningReward=100;
      }
      createGenesisBlock() //create a genesis block
      {
         return new Block("04/09/2023","Genesis Block","0");
      }
      getLatestBlock()
      {
         return this.chain[this.chain.length-1];
      }
      minePendingTransaction(miningRewardAddress)//methos is used the array
pendingTransactions
      {
         let block = new
Block(Date.now(),this.pendingTransactions,this.getLatestBlock().hash);
         block.mineBlock(this.difficulty);
console.log("Block mined Successfully...!!");
         this.chain.push(block);
this.pendingTransactions=[new
Transaction(null,miningRewardAddress,this.miningReward)];
      }
      addTransaction(transactions)
      {
         if(!transactions.fromAddress || !transactions.toAddress)
         {
            throw new Error('Transaction Must include from and to address..!!');
         }
         if(!transactions.isValid())
```

```
        {
            throw new Error('Can not add invalid transactions to blockchain..!!');
        }
        this.pendingTransactions.push(transactions);
    }
    getBalanceOfAddress(address)
    {
        let balance=0;
        for(const block of this.chain)
        {
            for(const trans of block.transactions)
            {
                if(trans.fromAddress== address)
                {
                    balance-=trans.amount;
                }
                if(trans.toAddress==address)
                {
                    balance+=trans.amount;
                }
            }
        }
        return balance;
    }
    isChainValid()
    {
        for(let i=1;i<this.chain.length;i++)
        {
            const currentBlock = this.chain[i];
const previousBlock =this.chain[i-1];
            if(!currentBlock.hasValidTransactions())
            {
                return false;
            }
            if(currentBlock.hash !== currentBlock.calculateHash())
            {
                return false;
            }
            if(currentBlock.previousHash !== previousBlock.hash)
            {
                return false;//chain is not valid so we return false
            }
}
        return true;
    }
  }
  module.exports.Blockchain = Blockchain;
```

```
module.exports.Transaction = Transaction;//exports classes for main.js
```

3)main.js

```
const {Blockchain,Transaction} = require('./Blockchain');
const SHA256 = require('crypto-js/sha256');//access a library and stored in one const const
EC = require('elliptic').ec;//elliptic curve
const ec = new EC('secp256k1');
const myKey =
ec.keyFromPrivate('b11de05167fdfbd2529081febfad8b2baef9bff6cace979b8b92be875f798fd
d');//private key
const myWalletAddress = myKey.getPublic('hex');
   let tanuCoin = new Blockchain(); //default constructor
   const tx1 = new Transaction(myWalletAddress,'address2',70);
   tx1.signTransaction(myKey);
   vaishCoin.addTransaction(tx1);
 console.log("Start Mining...");
   tanuCoin.minePendingTransaction(myWalletAddress);
console.log("Balance of myWalletAddress : ",
tanuCoin.getBalanceOfAddress(myWalletAddress));
   tanuCoin.minePendingTransaction(myWalletAddress);
console.log("Balance of myWalletAddress : ",
tanuCoin.getBalanceOfAddress(myWalletAddress));
console.log("Balance of Address1 : ", tanuCoin.getBalanceOfAddress('address1'));
console.log("Balance of Address2 : ", tanuCoin.getBalanceOfAddress('address2'));
```

• **Output –**

1) Keygenerator

```
PS E:\MCA_Sem_3\Blockchain\PracticalNo2> node keygenerator.js

Private Key :  b11de05167fdfbd2529081febfad8b2baef9bff6cace979b8b92be875f798fdd

Public Key :  044125c1d2566112d8016bec70034f17bcbf7df0b0a3fc0620779e7ae066b33ca2672c81b7e55906d78a32edbf8f94e0fa366741b800d1541e6f91b204b9fb8f02
PS E:\MCA_Sem_3\Blockchain\PracticalNo2>
```

2)main.is

```
PS E:\MCA_Sem_3\Blockchain\PracticalNo2> node main.js
Start Mining...
Block Mined : 0000f31ac3a4eff9bcef450e0cd17d440aed1a2ff7cfea98743d1f717f49a8a4
Nonce : 43966
Block mined Successfully...!!
Balance of myWalletAddress :  -70
Block Mined : 0000b445f314b722546cb8bffc5fc8919bf8def8636bdc7f0ffa42a6777be21d
Nonce : 126479
Block mined Successfully...!!
Balance of myWalletAddress :  30
Balance of Address1 :  0
Balance of Address2 :  70
PS E:\MCA_Sem_3\Blockchain\PracticalNo2>
```