2CSDE93 - Blockchain Technology

Practical 3

Aim: To perform thorough study and installation of Anaconda 5 0 1 and Python 3 6 and perform proof of work (consensus mechanism Also, notice the changes in mining rewards and nonce requirement

Author: Darshil Maru 20BCE514

Date: September 27, 2022

Code:

```
import hashlib
from datetime import datetime
import time


class block:
    def __init__(self, index, timestamp, data, previoushashvalue,
difficulty):
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previoushashvalue = previoushashvalue
        self.nonce = 1
        self.hash = self.hashfun()
        # self.print=self.print()

    def hashfun(self):
        total = str(self.index)+str(self.timestamp)+self.data + \
            str(self.previoushashvalue)+str(self.nonce)
        return hashlib.sha256(total.encode()).hexdigest()
 # def print(self):
 # print(self.index)
 # print(self.timestamp)
 # print(self.data)
 # print(self.previoushashvalue)
 # print(self.hash)

    def mine(self, difficulty):
        match = "0"*difficulty
        while(self.hash[0:difficulty] != match):
            self.nonce += 1
            self.hash = self.hashfun()
```

```
class blockchain:
    def __init__(self, difficulty=1):
        self.blockdetails = []
        self.genesisblock = self.genesisblock()
        self.difficulty = difficulty

    def genesisblock(self):
        print("THIS IS GENESIS BLOCK!!!!!\n")
        genesis = block(0, datetime.now(), "GENESIS BLOCK", 0, 0)
        self.blockdetails.append(genesis)

    def newblock(self, data):
        b = self.blockdetails[-1]
        start = time.process_time()
        print("\nMINING BLOCK-"+str(b.index+1)+"\n")
        createblock = block(b.index+1, datetime.now(),
                            data, b.hash, self.difficulty)
        print("Time =", time.process_time() - start)
        createblock.mine(self.difficulty)

        self.blockdetails.append(createblock)

    def hashvalidation(self):
        for i in range(1, len(self.blockdetails)):
            current = self.blockdetails[i]
            previous = self.blockdetails[i-1]
            if(current.previoushashvalue != previous.hash):
                return False
            if(current.hash != current.hashfun()):
                return False
            return True

    def printblocks(self):
        for i in range(len(self.blockdetails)):
```

```
            print(f"\nIndex: {self.blockdetails[i].index}
\nTimestamp: {self.blockdetails[i].timestamp} \nData:
{self.blockdetails[i].data} \nPrevious Hash Value:
{self.blockdetails[i].previoushashvalue} \nHash:
{self.blockdetails[i].hash}\n\n")


b = blockchain()
n = int(input("\nENTER NUMBER OF BLOCK TO MINE: "))
for i in range(n):
    data = input("\nENTER DATA OF BLOCK-"+str(i+1)+" : ")
    b.newblock(data)
print("\nBlock Detials:")
b.printblocks()
print("\nHash Validation:", b.hashvalidation())
```

Output:

Difficulty1:

```
MINING BLOCK-1

Time = 0.0

Block Detials:

Index: 0
Timestamp: 2022-10-11 16:43:30.891668
Data: GENESIS BLOCK
Previous Hash Value: 0
Hash: 0e95644debf3784e2c56cd2b37c6aeb3deb9d14c22fd2d177ee6a9ba9d0a7f9d



Index: 1
Timestamp: 2022-10-11 16:43:36.579055
Data: 100
Previous Hash Value: 0e95644debf3784e2c56cd2b37c6aeb3deb9d14c22fd2d177ee6a9ba9d0a7f9d
Hash: 0b2859d99d9d45e4538468bba213d4b620ab2ce455df9e8d52490b8471b139ae



Hash Validation: True
PS C:\Users\Admin>
```

Difficulty 4:

```
PS C:\Users\Admin> python -u "e:\7Sem\BCT\Practical3\PR3.py"
THIS IS GENESIS BLOCK!


ENTER NUMBER OF BLOCK TO MINE: 1

ENTER DATA -1 : 100

MINING BLOCK-1

Time = 0.0

Block Detials:

Index: 0
Timestamp: 2022-10-11 16:44:29.652492
Data: GENESIS BLOCK
Previous Hash Value: 0
Hash: 98ce7698911add60c336a1051e174bd6804ab2e257fe0976106db5c2cc83a01a



Index: 1
Timestamp: 2022-10-11 16:44:34.014510
Data: 100
Previous Hash Value: 98ce7698911add60c336a1051e174bd6804ab2e257fe0976106db5c2cc83a01a
Hash: 00007cc4fa506fced03904206cd9ee653cfda798323f7a7d81996b2cd7f2356d



Hash Validation: True
PS C:\Users\Admin>
```