**Name: Gondaliya Krish D. Practical: 3**

**Roll no: 19BCE072 Date: 28/09/2022**

**Course: Blockchain Technology (2CS701)**

**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 )

**Code**

from datetime import datetime

import hashlib

from time import time

class block:

    def \_\_init\_\_(self,index,time\_stamp,data,Pre\_hash,difficulty):

        self.index=index

        self.time\_stamp=time\_stamp

        self.data=data

        self.Pre\_hash=Pre\_hash

        self.nonce=1

        self.hash=self.get\_hash()

        if(difficulty>0):

            self.mine(difficulty)

    def get\_hash(self):

        block\_data=f"{self.index}{self.time\_stamp}{self.data}{self.Pre\_hash}{self.nonce}"

        return hashlib.sha256(block\_data.encode()).hexdigest()

    def mine(self,difficulty):

        match="0"\*difficulty

        while(self.hash[0:difficulty]!=match):

            self.nonce+=1

            self.hash=self.get\_hash()

class blockChain:

    def \_\_init\_\_(self,difficulty=3):

        self.blocks\_chain=[]

        self.\_\_Genesis()

        self.difficulty=difficulty

    def \_\_Genesis(self):

        self.blocks\_chain.append(block(0,datetime.now(),"Genesis Block",None,0))

    def Add\_block(self,data):

        b=self.blocks\_chain[-1]

        self.blocks\_chain.append(block(b.index+1,datetime.now(),data,b.hash,self.difficulty))

    def show\_blocks(self):

        i=1

        for b in self.blocks\_chain:

            print("\n Output of Block : ",i,end="\n\n")

            print(f" Block Index          :  {b.index} \n Block Data           :  {b.data} \n Block Time Stamp     :  {b.time\_stamp} \n Block Previous Hash  :  {b.Pre\_hash} \n Block Hash           :  {b.hash} \n Block Nonce          :  {b.nonce}")

            i+=1

    def HashValidation(self):

        for i in range(1,len(self.blocks\_chain)):

            current\_block=self.blocks\_chain[i]

            previous\_block=self.blocks\_chain[i-1]

            if(current\_block.Pre\_hash!=previous\_block.hash):

                return False;

            if(current\_block.hash!=current\_block.get\_hash()):

                return False

        return True

if \_\_name\_\_=="\_\_main\_\_":

    n=int(input(" Enter number of Block : "))

    d=int(input(" Enter difficulty : "))

    MyChain=blockChain(d)

    for i in range(1,n+1):

        MyChain.Add\_block(f"Block {i} DATA")

    print("\n\n"+f"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | NEW BLOCKCHAIN WITH {n} BLOCKS | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*".center(180),end="\n\n")

    MyChain.show\_blocks()

**Output:**



