BLOCKCHAIN TECHNOLOGY LAB (20CP406P)

LAB ASSIGNMENT - 2

B.Tech in Computer Science and Engineering Dept.,

Pandit Deendayal Energy University,

Gandhinagar

Name: Mire Kishorkumar Patel

Roll No.: 19BCP080

Branch: Computer Engineering

Lab Assignment 2

Aim: Create Blocks with all the parameters and link them with Hash Pointers

Introduction:

Hash:

There is a Hash value associated with every data. Single change in data will change the Hash value. For Example, for the data “Mire”, Hash value is XYZ. Now change the data to “Mire Patel”, then the Hash value will be ABC. Again if the data is changed to “Mire”, then the Hash value will be XYZ.

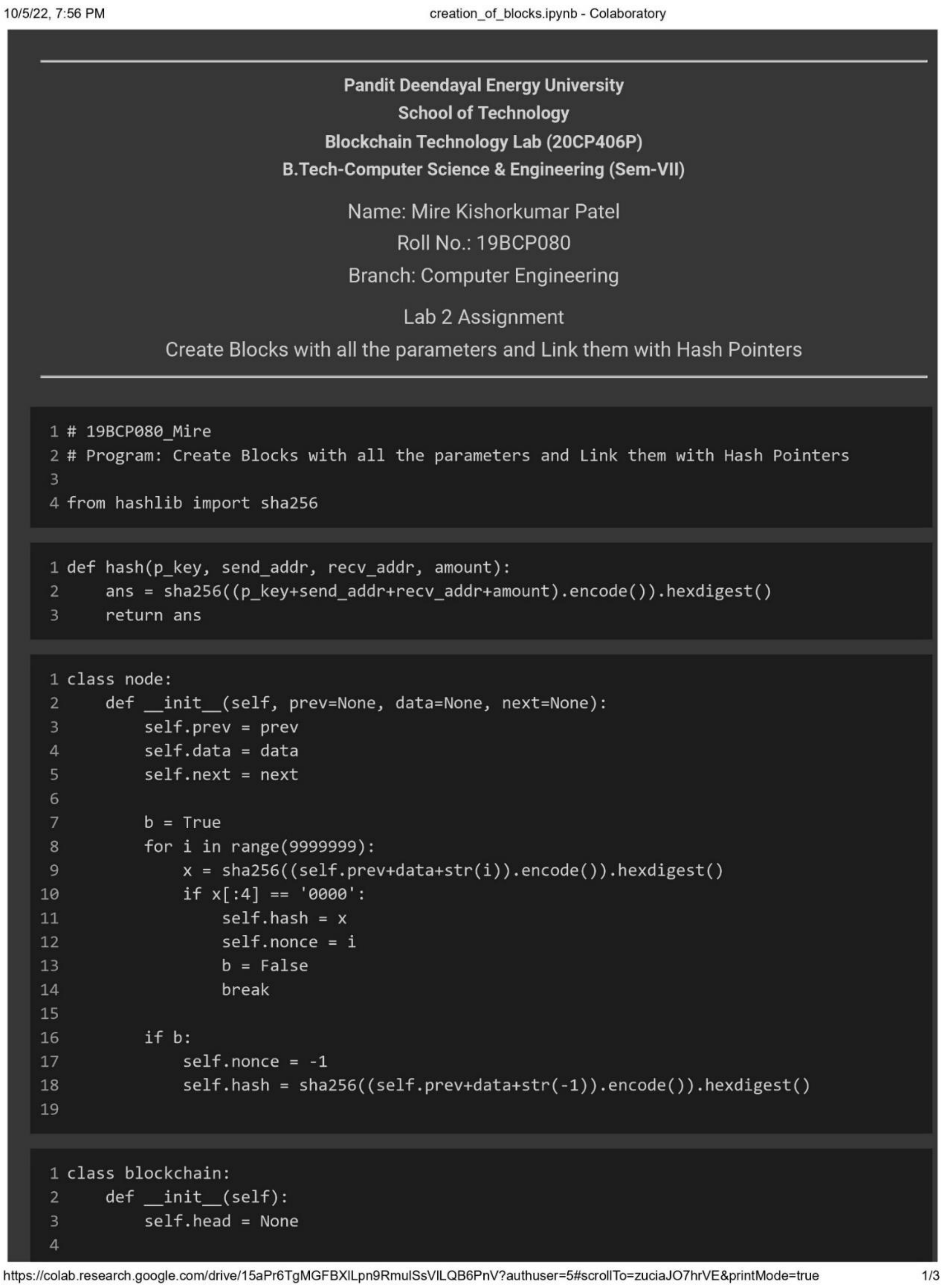
Block:

Each block contains a cryptographic hash of the data of the previous block. The nonce is calculated by the miners by solving cryptographic puzzles to propose the next block in the chain. It is known as proof of work.

Blockchain:

The blockchain is said to be immutable because of its cryptographic properties. But this does not mean that changing the data is impossible. It means that it is extremely hard to change the data and any change can be easily detected. A merkle tree is a binary tree with hash pointers. A merkle tree is a structure that allows for efficient and secure verification of content in a large body of data. The first block is known as the genesis block.

Code and the result:



10/5/22, 7:56 PM creation_of_blocks.ipynb - Colaboratory

Pandit Deendayal Energy University
School of Technology
Blockchain Technology Lab (20CP406P)
B.Tech-Computer Science & Engineering (Sem-VII)

Name: Mire Kishorkumar Patel
Roll No.: 19BCP080
Branch: Computer Engineering

Lab 2 Assignment
Create Blocks with all the parameters and Link them with Hash Pointers

```
# 19BCP080_Mire
# Program: Create Blocks with all the parameters and Link them with Hash Pointers

from hashlib import sha256
```

```
def hash(p_key, send_addr, recv_addr, amount):
    ans = sha256((p_key+send_addr+recv_addr+amount).encode()).hexdigest()
    return ans
```

```
class node:
    def __init__(self, prev=None, data=None, next=None):
        self.prev = prev
        self.data = data
        self.next = next

        b = True
        for i in range(9999999):
            x = sha256((self.prev+data+str(i)).encode()).hexdigest()
            if x[:4] == '0000':
                self.hash = x
                self.nonce = i
                b = False
                break

        if b:
            self.nonce = -1
            self.hash = sha256((self.prev+data+str(-1)).encode()).hexdigest()

```

```
class blockchain:
    def __init__(self):
        self.head = None

```

https://colab.research.google.com/drive/15aPr6TgMGFBXlLpn9RmulSsVlLQB6PnV?authuser=5#scrollTo=zuciaJO7hrVE&printMode=true 1/3

```
    def add_end(self, data):
        # new_node = node(data=data)
        if self.head is None:
            self.head = node(data=data, prev="00000000000000000000000000000000000000000000
            return

        last = self.head
        while last.next:
            last = last.next

        new_node = node(data=data, prev=last.data)
        last.next = new_node
        return

    def display_all(self):
        current = self.head
        while current:
            print("Data :", current.data)
            print("Hash :", current.hash)
            print("Nonce :", current.nonce)
            print()
            current = current.next
        return
```

```
x = blockchain()
x.add_end('Mire ')
x.add_end('Patel ')
x.add_end('-')
x.add_end('19BCP080')

x.display_all()
```

```
Data : Mire
Hash : 00003357cc3ec7f533f413639f068d52bfda9c707803dac6651677e1f24fa1dc
Nonce : 40072

Data : Patel
Hash : 000056660d4a58febb38c4b46772b73bee0ade3191d02b9da215b5238d8723df
Nonce : 150012

Data : -
Hash : 0000c4e3f260afadfcba33f4ad28bb4fbf2c3c952063a7e39838cfbfba0f299c
Nonce : 131937

Data : 19BCP080
Hash : 0000c0fcdd169d8972f8b64122f7c056cf048984f4fb9286b85c1cf8786d5e6c
Nonce : 41815
```