Aayush Shah
19BCE245
18 September 2022

BlockChain Technology

Practical 3

Proof of Work

- **Code :**

```
import datetime
import hashlib
import random
import json
# import JSON
# from flask import jsonify

# def compute_hash(index, previous_hash, timestamp,
data):
#     return hashlib.sha256((str(index) +
str(previous_hash) + str(timestamp) +
json.dumps(data)).encode('utf-8')).hexdigest()

def compute_hash(hash_data):
    return
hashlib.sha256(hash_data.encode('utf-8')).hexdigest()

def proof_of_work(hash_data, difficulty):
        nonce = 1
        if difficulty==-1:
            difficulty = random.randint(1, 5)
        print('current difficulty level is :
',difficulty)
        while True:
            temp_hash =
compute_hash(hash_data+str(nonce))
            if temp_hash[:difficulty] == '0'*difficulty:
```

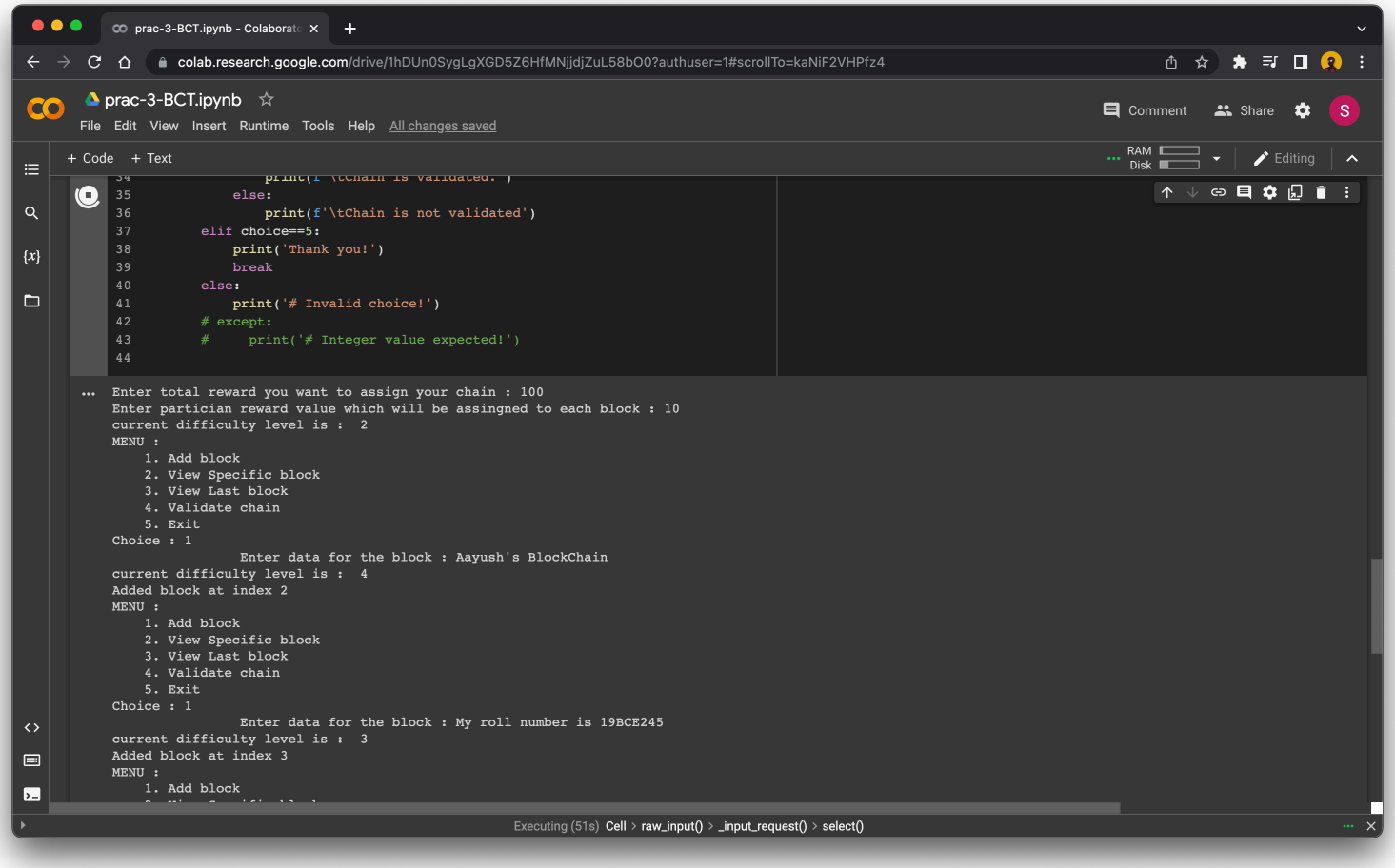
```
                break
            nonce +=1
        return temp_hash,nonce,difficulty

class Block:
    def __init__(self, index, data, previous_hash,
reward):
        self.index = index
        self.data = data
        self.previous_hash = previous_hash
        self.timestamp = str(datetime.datetime.now())
        hash_data = str(self.index) +
str(self.previous_hash) + str(self.timestamp) +
json.dumps(self.data)
        self.hash,self.nonce,self.difficulty =
proof_of_work(hash_data,-1)
        self.reward = reward

    def print_block_details(self):
        print(f'Details for block indexed at {self.index}
: ')
        print(f'\tData : {self.data}')
        print(f'\tTimeStamp : {self.timestamp}')
        print(f'\tHash : {self.hash}')
        print(f'\tPrevious Hash : {self.previous_hash}')
        print(f'\tReward : {self.reward}')
        print(f'\tNonce : {self.nonce}')
        print(f'\tDifficulty : {self.difficulty}')

class BlockChain:
    # chain = []
    def __init__(self, total_reward, partician):
        self.chain = []
        self.partician = partician
        self.total_reward = total_reward - partician
        genesis_block = Block(len(self.chain)
+1,'Aayush\'s BlockChain!',0, self.partician)
        self.chain.append(genesis_block)

    def add_block(self, data):
        assigned_reward = 0
        if self.total_reward-self.partician>0:
            self.total_reward -= self.partician
```

```
            assigned_reward = self.partician
        elif self.total_reward>0:
            assigned_reward = self.total_reward
            self.total_reward = 0
        new_block = Block(len(self.chain)+1, data,
self.chain[-1].hash, assigned_reward)
        self.chain.append(new_block)


    def get_previous_block(self):
        return self.chain[-1]

    def get_specific_block(self,index):
        return self.chain[index]

    def print_block(self, block):
        block.print_block_details()

    def chain_validation(self):
        hash_data,_,__ =
proof_of_work(str(self.chain[0].index) +
str(self.chain[0].previous_hash) +
str(self.chain[0].timestamp) +
json.dumps(self.chain[0].data), self.chain[0].difficulty)
        if self.chain[0].hash != hash_data:
            return False
        print('\t> genesis block is validated.')
        for i in range(1,len(self.chain)):
            if self.chain[i].previous_hash !=
self.chain[i-1].hash:
                return False
            hash_data,_,__ =
proof_of_work(str(self.chain[i].index) +
str(self.chain[i].previous_hash) +
str(self.chain[i].timestamp) +
json.dumps(self.chain[i].data), self.chain[i].difficulty)
            if self.chain[i].hash != hash_data:
                return False
            if i != len(self.chain)-1 and
self.chain[i].hash != self.chain[i+1].previous_hash:
                return False
            print(f'\t> {i+1}th block is validated.')
        return True
```

```
if __name__ == "__main__":
    total_reward = int(input('Enter total reward you want
to assign your chain : '))
    partician = total_reward
    while partician>=total_reward:
        partician = int(input('Enter partician reward
value which will be assingned to each block : '))
        if partician>=total_reward:
            print('Partician value should be less then
reward value.')

    myBlockChain = BlockChain(total_reward, partician)

    while True:
        print("""MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit""")
        choice = int(input("Choice : "))
        # try:
        if choice==1:
            data = input('\t\tEnter data for the block :
')
            myBlockChain.add_block(data)
            print(f'Added block at index
{len(myBlockChain.chain)}')
        elif choice==2:
            index = int(input('\t\tEnter block index :
'))
            try:

myBlockChain.print_block(myBlockChain.get_specific_block(
index-1))
            except:
                print('# Invalid index entered!')
        elif choice==3:

myBlockChain.print_block(myBlockChain.get_previous_block(
))
        elif choice==4:
```

```
        if myBlockChain.chain_validation():
            print(f'\tChain is validated.')
        else:
            print(f'\tChain is not validated')
    elif choice==5:
        print('Thank you!')
        break
    else:
        print('# Invalid choice!')
    # except:
    #     print('# Integer value expected!')
```

- **Output :**



• **Full output text :**

```
Enter total reward you want to assign your chain : 100
Enter partician reward value which will be assingned to each
block : 10
current difficulty level is :  2
MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit
Choice : 4
current difficulty level is :  2
     > genesis block is validated.
     Chain is validated.
MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit
Choice : 1
          Enter data for the block : Aayush's BlockChain
current difficulty level is :  3
Added block at index 2
MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit
Choice : 1
          Enter data for the block : My roll number is
19BCE245
current difficulty level is :  2
Added block at index 3
MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit
Choice : 3
Details for block indexed at 3 :
     Data : My roll number is 19BCE245
     TimeStamp : 2022-10-11 16:19:01.592967
     Hash :
0011c6931286dc936956d9d0543933ec2cd0537c667c8f54af5355130a5e59
55
```

```
     Previous Hash :
0005f950f7a42b592680cd11d94894ec713c6894819d1720af17c6011f8537
93
     Reward : 10
     Nonce : 175
     Difficulty : 2
MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit
Choice : 2
          Enter block index : 2
Details for block indexed at 2 :
     Data : Aayush's BlockChain
     TimeStamp : 2022-10-11 16:18:51.893747
     Hash :
0005f950f7a42b592680cd11d94894ec713c6894819d1720af17c6011f8537
93
     Previous Hash :
00b44e7051101ec8071492f121db6ae2810101ae879c17bbcaaf007b9cd4ef
50
     Reward : 10
     Nonce : 4922
     Difficulty : 3
MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit
Choice : 4
current difficulty level is :  2
     > genesis block is validated.
current difficulty level is :  3
     > 2th block is validated.
current difficulty level is :  2
     > 3th block is validated.
     Chain is validated.
MENU :
    1. Add block
    2. View Specific block
    3. View Last block
    4. Validate chain
    5. Exit
Choice : 5
Thank you!
```

• Learning Outcomes :

From this practical, I learned about proof of work.

Proof of work (PoW) is a type of cryptographic proof in which one party (the prover) demonstrates to others (the verifiers) that a specific amount of computational effort has been expended. Verifiers can then confirm this expense with little effort on their part.

Here I generate difficulty level randomly and then generate suitable hash value based on that in proof of work. Also chain validation is performed accordingly.