MOHD ELFARIZ BIN MOHD IRFAN (CA17005)

LAB ASSIGNMENT 4 – BLOCKCHAIN CRYPTOCURRENCY

Question: Design a simple cryptocurrency based blockchain using Python.

Based on the concept of blockchain and open ledger, I managed to design a simple cryptocurrency named ElCoin where it applies the concept of blockchain and secure it with proof of work consensus type for securing the blockchain.

ElCoin started with a genesis block where it is the header of the blockchain. It acts as the starting of transaction for ElCoin.

There are basically 2 classes to hold the method and parameters for this cryptocurrency. First is Block class and second is Blockchain class. The first Block class will act as a single block where it then will be chained to make a chain of blocks at the end. The chaining of blocks takes place such that if one block is tampered, the rest of the chain will become invalid. The initial block class (constructor) method is shown below:

```
import hashlib
import time
import json

class Block:
    def __init__(self, index, proof_no, prev_hash, data, timestamp=None):
# Constructor for block created
        self.index = index # Index for each block
        self.proof_no = proof_no # Proof of work number for each block
        self.prev_hash = prev_hash # Previous block hashes to chain with new b
lock
        self.data = data # Transaction data
        self.timestamp = timestamp or time.time() # Timestamp for each block

    @property
    def calculate_hash(self):
        # This function will calculate the hash of every block
        block_of_string = "{}{}{}{}{}".format(self.index, self.proof_no, self.
prev_hash, self.data, self.timestamp)
        return hashlib.sha256(block_of_string.encode()).hexdigest()

    def __repr__(self): # Representation of the block format
        return "\n{} - {} - {} - {} - {}".format(self.index, self.proof_no, se
lf.prev_hash, self.data, self.timestamp)
```

Basically, there are 3 methods exists in the Block class which is the __init__() method which act as the constructor method for each block, the calculate_hash() method which will calculate hashes for each block where it will use the SHA256 function where the hashes will be use to connect each blocks later.

The last method is for the representation of the data that we have processed to be easily readable in the output later.

As for the second class which is Blockchain class, it will involve the process of chaining those blocks that we have created using the hashes that we already calculated earlier. This is where most of the action is going to take place. The Blockchain class will have several methods to help in completing various tasks in the blockchain. The code for the Blockchain class is shown below:

```
class Blockchain:
    def __init__(self): # Constructor for the blockchain
        self.chain = []
        self.current_data = []
        self.nodes = set()
        self.construct_genesis()

    def construct_genesis(self): # Construct the initial block
        self.construct_block(proof_no=0, prev_hash=0)

    def construct_block(self, proof_no, prev_hash): # Constructs a new block a
nd add it to the chain
        block = Block(index=len(self.chain), proof_no=proof_no, prev_hash=prev
_hash, data=self.current_data)
        self.current_data = []
        self.chain.append(block)
        return block

    @staticmethod
    def check_validity(block, prev_block): # Checks whether the blockchain is
valid
        if prev_block.index + 1 != block.index: # Condition to check the block
 index
            return False

        elif prev_block.calculate_hash != block.prev_hash: # Condition to comp
are previous hashes
            return False

        elif not Blockchain.verifying_proof(block.proof_no. prev_block.proof_n
o): # Condition to verify proof of work
            return False

        elif block.timestamp <= prev_block.timestamp: # Condition to compare b
ased on timestamp
            return False

        return True # If all ok then it can be chained

    def new_data(self, sender, recipient, quantity): # Adds a new transactions
 to the data of the transactions
```

```
        self.current_data.append({
            'sender': sender,
            'recipient': recipient,
            'quantity': quantity
        })
        return True

    @staticmethod
    def proof_of_work(last_proof): # Security features of the blockchain
        proof_no = 0
        while Blockchain.verifying_proof(proof_no, last_proof) is False:
            proof_no +=1

        return proof_no

    @staticmethod
    def verifying_proof(last_proof, proof):
        guess = f'{last_proof}{proof}'.encode()
        guess_hash = hashlib.sha256(guess).hexdigest()
        return guess_hash[:4] == "0000"

    @property
    def latest_block(self): # Returns the last block in the chain
        return self.chain[-1]

    def block_mining(self, details_miner):
        self.new_data(
            sender="0",  # It implies that this node has created a new block
            recipient=details_miner, # Creating a new block (or identifying th
e proof number) is awarded with 1
            quantity=1,
        )

        last_block = self.latest_block
        last_proof_no = last_block.proof_no
        proof_no = self.proof_of_work(last_proof_no)
        last_hash = last_block.calculate_hash
        block = self.construct_block(proof_no, last_hash)

        return vars(block)

    def create_node(self, address):
        self.nodes.add(address)
        return True

    @staticmethod
    def obtain_block_object(block_data): # Obtains block object from the block
 data
```

```
        return Block(
            block_data['index'],
            block_data['proof_no'],
            block_data['prev_hash'],
            block_data['data'],
            timestamp=block_data['timestamp'])
```

As shown in the code below, the Blockchain class will have a minimum of 9 methods to help in completing the tasks. The first method will be the constructor class where it will ensure that the blockchain is instantiated. This is the basic method in object-oriented programming techniques where this is the initialization of the object in the class.

Next is the construct_genesis() method to build the initial block in the chain. Which means that the first block in blockchain is created. It will work together with the construct_block() method. I give the initial value of the proof_no and prev_hash as 0. The output of this can be seen later.

For the construct_block() method, it is used for creating new blocks in the blockchain. The attributes/parameters that involved in this method is the index, proof_no and prev_hash, data (will consists of ElCoin transaction data), self.current_data where it is used to reset the transaction list on the node to make sure that future transactions data are added into this list continuously, self_chain.append() to join the newly created blocks to the chain. Then we return it.

The check_validity() method is used to assess the integrity of the blockchain and ensuring no anomalies are present. This method uses if statements to check whether the hash of every block is correct which also verify the index or the arrangement of all the blocks with the usage of the hashes. If everything is correct, then it will return True and False if vice versa.

The new_data() method is used for adding the data of transactions to a block. It will accepts 3 parameters (sender, recipient, and quantity) which then will be appended to self.current_data list. Anytime a new block is created, this list is allocated to that block and reset once more.

Proof_of_work() method is used to identify a number that solves a problem after a certain amount of computing work is done. In this case, a simple algorithm is used where it will identifies a number f' such that hash(ff') contains 4 leading zeroes which later will be verified with verifying_proof() method.

Lastly, the latest_block() method is a helper method that assists in obtaining the last block in the blockchain.

Results:

The codes are tested by running the code below to print the genesis block:

```
# We add the transaction here
blockchain = Blockchain()

print("Starting ElCoin mining ...")
print(blockchain.chain) # Print current chain
```

The output is the genesis block data which is shown below:

```
PS C:\Users\MOHDELFARIZ\Desktop\Lab6_BLOCKCHAIN> & C:/Users/MOHDELFARIZ/AppData/Local/Programs/Python/Python39-32/python.exe c:/
Starting ElCoin mining ...
[
0 - 0 - 0 - [] - 1607500249.4948373]
PS C:\Users\MOHDELFARIZ\Desktop\Lab6_BLOCKCHAIN>
```

We can see that the genesis block contain all the parameters that we have initialized in the constructor method, The first zero indicate the index of the block, the second and third 0 indicate the proof_no. and prev_hash that we have set to 0, the transaction data is nothing in the [] bracket, and lastly is the timestamp in the last column.

Next, we try to add this code as a new transaction using ElCoin.

```
# We add the transaction here
blockchain = Blockchain()

print("Starting ElCoin mining ...")
print(blockchain.chain) # Print current chain

# We try to add new transaction (AHMAD sends ALI some ElCoin)
last_block = blockchain.latest_block
last_proof_no = last_block.proof_no
proof_no = blockchain.proof_of_work(last_proof_no)
blockchain.new_data(
    sender="AHMAD",
    recipient="ALI",
    quantity=5,
)
last_hash = last_block.calculate_hash
block = blockchain.construct_block(proof_no, last_hash)

print("\nElCoin Transaction has been successful")
print(blockchain.chain)
```

The output can be shown below:

```
PS C:\Users\MOHDELFARIZ\Desktop\Lab6_BLOCKCHAIN> & C:/Users/MOHDELFARIZ/AppData/Local/Programs/Python/Python39-32/python.exe c:/Users/MOHDELFARIZ/Desktop/Lab6_
Starting ElCoin mining ...

ElCoin Transaction has been successful
[
0 - 0 - 0 - [] - 1607500595.549463,
1 - 88914 - 54621db7a28d6b8a84363ccd74da51e7892e6cad859355680546373521409 8c8 - [{'sender': 'AHMAD', 'recipient': 'ALI', 'quantity': 5}] - 1607500595.7409458]
PS C:\Users\MOHDELFARIZ\Desktop\Lab6_BLOCKCHAIN>
```

We can see that the first transaction is successful as we can see that it has been chained into the ElCoin blockchain. We can see the first transaction have the block index of 1, the proof_no. and its prev_hash, the transaction data which consists of the three parameter earlier (sender, recipient, and quantity) and lastly the timestamp for the newly created block.

That is the basic design of blockchain cryptocurrency with ElCoin.