1. 새로운 블록을 생성하고 chain에 추가하는 파이썬 함수를 구현하시오.

def new\_block(self, proof, previous\_hash=None):

block = {

'index': len(self.chain) + 1,

'timestamp': datetime.datetime.now(),

'transactions': self.current\_transaction,

'proof': proof,

'previous\_hash': previous\_hash or self.hash(self.chain[-1]) }

self.current\_transaction = []

self.chain.append(block)

return block

1. block chain의 유효성을 검증하는 파이썬 함수를 작성하시오.

def is\_valid\_chain(self, chain):

previous\_block = chain[0]

for i in range(1, len(chain)):

block = chain[i]

if block['previous\_hash'] != self.hash(previous\_block):

return False

previous\_block = block

return True

1. block chain의 충돌현상을 해결하는 코드를 파이썬 함수로 작성하시오

def resoluve\_conflict():

neighbors = nodes

new\_chain = None

max\_length = len(self.chain)

for node in neighbors:

tmp\_url = 'http://' + str(node) + '/chain'

response = requests.get(tmp\_url)

if response.status\_code = 200:

length = response.json()['length']

chain = response.json()['chain']

if length > max\_length and self.valid\_chain(chain):

max\_length = length

new\_chain = chain

if new\_chain:

self.chain = new\_chain

return True

return Fasle

1. block을 sha256 알고리즘으로 암호화한 헤쉬값을 리턴하는 파이썬 함수를 작성하시오.

def hash(self, block):

encoded\_block = json.dumps(block, sort\_keys = True).encode()

return hashlib.sha256(encoded\_block).hexdigest()

1. 암호화폐 채굴과정을 구현한 파이썬 코드를 작성하시오..

def proof\_of\_work(self, previous\_proof):

new\_proof = 1

check\_proof = False

while check\_proof is False:

hash =hashlib.sha256(str(new\_proof\*\*2-previous\_proof\*\*2).encode()).hexdigest()

if hash[:4] == '0000':

check\_proof = True

else:

new\_proof += 1

return new\_proof

6. Transaction(거래)를 파이썬 List에 담는 코드를 작성하시오.

def add\_transaction(self, sender, receiver, amount):

self.transactions.append({

'sender': sender,

'receiver': receiver,

'amount': amount})

return self.last\_block['index'] + 1

7.  채굴요청(request)를 flask server routing을 통해 구현해주는 코드를 작성하시오..

@app.route('/mine\_block', methods = ['GET'])

def mine\_block():

previous\_block = blockchain.chain[-1]

previous\_proof = previous\_block['proof']

proof = blockchain.proof\_of\_work(previous\_proof)

blockchain.add\_transaction(sender = 'someone', receiver = 'some', amount = 1)

previous\_hash = blockchain.hash(previous\_block)

block = blockchain.new\_block(proof, previous\_hash)

response = {

‘message’ : ‘new block found’,

‘index’ : block[‘index’],

‘transactions’ : block[‘transaction’],

‘proof’ : block[‘proof’],

‘previous\_hash’ : block[‘previous\_hash’]

}

return jsonify(response), 200