Yousuf Alkhiyami
Yalkhiya
67-346 Final Project

i. Paypal, one of the worlds biggest online payment systems has been facing so much trouble and backlash due to their recent policies; which allows them to withdraw money from your account if you posted misinformation online. People started withdrawing their money from paypal due to fear from having their money taken away. Also, a crypto exchange called FTX has collapsed due to them using their clients money to pay off their debts and invest without permission. In my project, I decided to create a secure online payment system using blockchain.

ii. a. My motivation is the above, the fact that some of the largest exchanges and online payment systems are doing some very unethical stuff is my motivation. I was going to do something for foodstuffs and consulted with my father and contacted their IT department to see if I could work with them to come up with blockchain solutions, but there wasn't really anything I could use blockchain for in the company, so I settled on this after consulting with my father.

b. One of the key challenges to my program is the scalability, as its rather small scale and not efficient to be implemented on a larger scale. I will not be solving this problem in my program due to my program being a simpler version of an alpha version, but having more nodes and miners to verify the hashes would be very helpful in terms of scalability. Another problem in actually implementing my program would be mining. Mining is verifying hashes, which seems simple, but I have struggled a bit with it in the labs, but I will overcome it using online resources for help and cite my sources. I have my sources at the bottom which I relied on heavily, but I wrote all the code myself and understand every single line of code I wrote.

iii. First, I will make a helper to hash the inputs. Then, I will make a function that creates exchanges between two different users and actually create the transactions we will use. I will also create the blocks afterwards.

From there, we need to start building the blockchain. I will start by creating the genesis block we were shown in the lab, which is the first block in the blockchain. It is different than other blocks since it doesnt have a parent block, it isnt linked to any previous blocks. Then, I will make a few functions that make sure the blockchain is valid and secure, which is the purpose of my project. I will check the hashes, block number and make sure its correct and valid. Finally, I will implement the transactions into the blocks to complete the blockchain.

iv. COPIED FROM VSC

```
import hashlib,json,sys
import unicodedata
import random
random.seed(0)
```

```
#helper function for the hashing algorithm were using
def hashMsg(msg=""):


    #if the msg were hashing is a string, sort it
    if type(msg)!=str:
        msg = json.dumps(msg,sort_keys=True)

    #hash it using SHA256
    if sys.version_info.major == 2:
        return unicodedata(hashlib.sha256(msg).hexdigest(),'utf-8')
    else:
        return hashlib.sha256(str(msg).encode('utf-8')).hexdigest()

def makeTransaction(maximumV=3):

    # create transactions up to the maximum value
    sign = int(random.getrandbits(1))*2 - 1
    amount = random.randint(1,maximumV)
    AliPays = sign * amount
    YousufPays   = -1 * AliPays
    return {u'Ali':AliPays,u'Yousuf':YousufPays}

txnBuffer = [makeTransaction() for i in range(30)]

def updateState(txn, state):

    #if inputted transaction is valid, update the state
    state = state.copy() # copy the dict to avoid messing up the work
since dicts are mutable
    for key in txn:
        if key in state.keys():
            state[key] += txn[key]
        else:
            state[key] = txn[key]
    return state

def isValidTxn(txn,state):
```

```
    if sum(txn.values()) != 0:
        return False
    #make sure transaction doesnt cause overdrafting issues
    for key in txn.keys():
        if key in state.keys():
            acctBalance = state[key]
        else:
            acctBalance = 0
        if (acctBalance + txn[key]) < 0:
            return False

    return True


#heres an example of the blockchain so far
state = {u'Ali':5,u'Yousuf':5}

print(isValidTxn({u'Ali': -3, u'Yousuf': 3},state))  # working transaction
example
print(isValidTxn({u'Ali': -4, u'Yousuf': 3},state))  # here i tried to
create a token, obviously returns false
print(isValidTxn({u'Ali': -6, u'Yousuf': 6},state))  # if we overdraft, it
also returns false
print(isValidTxn({u'Ali': -4, u'Yousuf': 2,'faisal':2},state)) # we can
create new users, thats fine
print(isValidTxn({u'Ali': -4, u'Yousuf': 3,'faisal':2},state)) # but if we
overdraft it returns false regardless

state = {u'Ali':50, u'Yousuf':50}  # create the first state
# a genesis block is the first block in a blockchain, here we create it
genesisBlockTxns = [state]
genesisBlockContents =
{u'blockNumber':0,u'parentHash':None,u'txnCount':1,u'txns':genesisBlockTxn
s}
#hash the contents of the block using the helper we made
genesisHash = hashMsg(genesisBlockContents)
genesisBlock = {u'hash':genesisHash,u'contents':genesisBlockContents}
genesisBlockStr = json.dumps(genesisBlock, sort_keys=True)
```

```
chain = [genesisBlock]

def makeBlock(txns,chain):
    parentBlock = chain[-1]
    parentHash  = parentBlock[u'hash']
    blockNumber = parentBlock[u'contents'][u'blockNumber'] + 1
    txnCount = len(txns)
    blockContents =
{u'blockNumber':blockNumber,u'parentHash':parentHash,u'txnCount':len(txns)
,'txns':txns}
    blockHash = hashMsg( blockContents )
    block = {u'hash':blockHash,u'contents':blockContents}

    return block

blockSizeLimit = 5  # number of txn in each block

while len(txnBuffer) > 0:
    bufferStartSize = len(txnBuffer)

    # create a set of transactions
    txnList = []
    while (len(txnBuffer) > 0) & (len(txnList) < blockSizeLimit):
        newTxn = txnBuffer.pop()
        validTxn = isValidTxn(newTxn,state) # using the isvalid we
created, it returns false if transaction isnt valid

        if validTxn: # if its valid, then it wont be false
            txnList.append(newTxn)
            state = updateState(newTxn,state)
        else:
            print("ignored transaction")
            sys.stdout.flush()
            continue  # if transaction invalid, ignore it, print ignored
and continue to keep going with the loop

    # here we create the block
    myBlock = makeBlock(txnList,chain)
    # append to the block
    chain.append(myBlock)
```

```
def compareBlockHashes(block):
    # mining part. if its invalid and not equal, make an exception
    expectedHash = hashMsg( block['contents'] )
    if block['hash']!=expectedHash:
        raise Exception('hash doesnt match block content %s'%
block['contents']['blockNumber'])
    return

def isValidBlock(block,parent,state):
    # here we make sure that the transactions are valid, block content is
valid,
    # block number increments by 1 after each time
    parentNumber = parent['contents']['blockNumber']
    parentHash = parent['hash']
    blockNumber = block['contents']['blockNumber']

    for txn in block['contents']['txns']:
         # check validity, if its invalid then raise an exception
        if isValidTxn(txn,state):
            state = updateState(txn,state)
        else:
            raise Exception('block has invalid transaction %s:
%s'%(blockNumber,txn))

    compareBlockHashes(block) # checks the correctness of hashes in a
block
    #if the number of blocks is not equals to the number of parents +1
then raise an exception
    if blockNumber!=(parentNumber+1):
        raise Exception('block number incorrect %s'%blockNumber)

    if block['contents']['parentHash'] != parentHash:
        raise Exception('parent hashed incorrectly, not accurate
%s'%blockNumber)

    return state

def checkChain(chain):
```

```
    # checks the entire chain from the genesis block and make sure that
the chain is correct, valod, no overdraft.
    # returns a dict of balances and such
    # returns false if error is there
    if type(chain)==str:
        try:
            chain = json.loads(chain)
            assert(type(chain)==list)
        except: # this catches all errors
            return False
    elif type(chain) != list:
        return False

    state = {}
    # make sure each block is in a valid state
    for txn in chain[0]['contents']['txns']:
        state = updateState(txn,state)
    compareBlockHashes(chain[0])
    parent = chain[0]

    # make sure the reference to parent block is correct and the block
number is valid
    for block in chain[1:]:
        state = isValidBlock(block,parent,state)
        parent = block

    return state

checkChain(chain)

chainAsText = json.dumps(chain,sort_keys=True)
checkChain(chainAsText)

import copy
nodeBchain = copy.copy(chain)
nodeBtxns  = [makeTransaction() for i in range(5)]
newBlock   = makeBlock(nodeBtxns,nodeBchain)

print("Blockchain on Node A is currently %s blocks long"%len(chain))
```

```
try:
    print("New Block Received; checking validity...")
    state = isValidBlock(newBlock,chain[-1],state) # update the state,
will throw error if incorrect
    chain.append(newBlock)
except:
    print("Invalid block; ignoring and waiting for the next block...")

print("Blockchain on Node A is now %s blocks long"%len(chain))
```

REFERENCES:

https://www.cohenmichael.fr/web/index.php?option=com_content&view=article&id=29&catid=10&Itemid=122 I used this quite a bit to fill the gaps in my knowledge and copied a bit of the code from it, but made sure I understood its functions.

https://www.youtube.com/watch?v=pYasYyjByKI

https://www.activestate.com/blog/how-to-build-a-blockchain-in-python/