Block Chain Project

Program: CSE

Course Code: ***432***

Course Name: ***Computer Security***

Name	Code	Contribution
Omar Mohamed Mohamed	1700907	• Node class • Merkle tree class • Transaction class • Pool class • Main Test case
Omar Mohamed Omar	1700903	• Block class • Blockchain class • User class • Main Test case
Amr Ahmed Mohamed	1700918	• Block class • Blockchain class • User class • Main Test case
Loay Abd-Allah Youssef	1701043	• Node class • Merkle tree class • Transaction class • Pool class • Main Test case
Mohamed Khaled Sayed	1601166	• Node class • Merkle tree class • Transaction class • Pool class • Main Test case

GitHub: https://github.com/OmarRash/Blockchain_project

```python
import hashlib
import time
import copy

class Node:
    def __init__(self, data):
        self.left=None
        self.right=None
        self.data=data

class Markle_tree:
    def __init__(self,transactions_list):
        self.node_list = []
        for i in range (0, 8, 2):
            node_data=hashlib.sha256((transactions_list[i]+transactions_list[i+1]).encode()).hexdigest()
            node = Node(node_data)
            node.left=transactions_list[i]
            node.right=transactions_list[i+1]
            self.node_list.append(node)

        for i in range (0, 6, 2):
            node_data=hashlib.sha256((self.node_list[i].data+self.node_list[i+1].data).encode()).hexdigest()
            node = Node(node_data)
            node.left = self.node_list[i]
            node.right = self.node_list[i + 1]
            self.node_list.append(node)

    def root_node(self):
        return self.node_list[len(self.node_list)-1]

class Transaction:
    def __init__(self, sender, reciever, amount):
        self.sender= sender
        self.reciever= reciever
        self.amount= amount
        self.transaction_time=time.time()
        self.transaction_data=f"{sender} - {reciever}-{self.transaction_time}-{amount}"
        self.hash=hashlib.sha256(self.transaction_data.encode()).hexdigest()

    def hash_value(self):
        return self.hash
```

```
class Block:
    def __init__(self, previous_block_hash, root,timestamp):
        self.previous_block_hash = previous_block_hash
        self.root = root
        self.timestamp=timestamp
        self.nonce = 0
        self.block_data = f"{root.data} - {previous_block_hash}-{timestamp}-{self.nonce}"
        self.block_hash = hashlib.sha256(self.block_data.encode()).hexdigest()


class Pool:
    def __init__(self):
        self.pool = []

    def add_transaction(self,transaction):
        if type(transaction)==str:
            self.pool.append(transaction)
        elif len(transaction)>1:
            for i in range(len(transaction)):
                self.pool.append(transaction[i])

    def generate_block(self):
        if len(self.pool)>=8:
            root=Markle_tree(self.pool).root_node()
            t=time.time()
            self.pool=self.pool[8:]
            return Block("", root, t)
        else:
            return False

class Blockchain:
    def __init__(self):
        self.chain = []
        self.generate_genesis_block()
        self.difficulty=5

    def set_difficulty(self,n):
        self.difficulty=n

    def get_difficulty(self):
        return self.difficulty

    def generate_genesis_block(self):
        genesis_node=Node("Genesis Block")
        self.chain.append(Block("0", genesis_node, 0))
```

```
def mineBlock(self,B,speed):
        x="0"*self.difficulty
        i=0
        for i in range(speed):
            if B.block_hash[:self.difficulty] == x:
                print(B.nonce)
                self.chain.append(B)
                return True
            B.nonce = B.nonce + 1
            B.block_data = f"{B.root.data} - {B.previous_block_hash}-{B.timestamp}-{B.nonce}"
            B.block_hash = hashlib.sha256(B.block_data.encode()).hexdigest()
        return False


    def display_chain(self):
        for i in range(len(self.chain)-1):
            print(f"Data {i + 1}: {self.chain[i+1].block_data}")
            print(f"Hash {i + 1}: {self.chain[i+1].block_hash}\n")

    @property
    def last_block(self):
        return self.chain[-1]


class User:
    def __init__(self,id):
        self.id = id
        self.local_block_chain = Blockchain()

    def set_BC_block_chain(self,bc):
        self.compare(bc)

    def compare(self,bc):
        if len(bc.chain)>len(self.local_block_chain.chain):
            self.local_block_chain = copy.deepcopy(bc)
        elif len(bc.chain)==len(self.local_block_chain.chain):
            if((bc.chain[-1].timestamp-self.local_block_chain.chain[-1].timestamp)<0):
                self.local_block_chain = copy.deepcopy(bc)

    def create_block(self,b):
        b.previous_block_hash = self.local_block_chain.chain[-1].block_hash
        b1 = copy.deepcopy(b)
        return b1
```

```
if __name__ == "__main__":
    user = User(1)
    user1 = User(2)
    attacker = User(-1)
    user2 = User(3)
    transactions = Pool()
    t = []
    for i in range(40):
        transactions.add_transaction(Transaction("joe","bob",str(i+1)).hash_value())
    b = transactions.generate_block()
    b1 = user1.create_block(b)
    b2 = user2.create_block(b)

 while(1):
        if user1.local_block_chain.mineBlock(b1,51):
            user.set_BC_block_chain(user1.local_block_chain)
            user2.set_BC_block_chain(user1.local_block_chain)
            attacker.set_BC_block_chain(user1.local_block_chain)
            print("block 1 : User1 is TOP")
            break
        elif user2.local_block_chain.mineBlock(b2,49):
            user.set_BC_block_chain(user2.local_block_chain)
            user1.set_BC_block_chain(user2.local_block_chain)
            attacker.set_BC_block_chain(user2.local_block_chain)
            print("block 1 : User2 is TOP")
            break

    b = transactions.generate_block()
    b1 = user1.create_block(b)
    b2 = user2.create_block(b)

    while(1):
        if user2.local_block_chain.mineBlock(b2,51):
            user.set_BC_block_chain(user2.local_block_chain)
            user1.set_BC_block_chain(user2.local_block_chain)
            attacker.set_BC_block_chain(user2.local_block_chain)
            print("block 2 : User2 is TOP")
            break

        elif user1.local_block_chain.mineBlock(b1,49):
            user.set_BC_block_chain(user1.local_block_chain)
            user2.set_BC_block_chain(user1.local_block_chain)
            attacker.set_BC_block_chain(user1.local_block_chain)
            print("block 2 : User1 is TOP")
            break
```

Output:

Try 1:

```
nonce: 1998819
block 1: User1 is TOP

nonce: 194319
block 2: User2 is TOP

after adding 2 blocks:
Data 1: 571f7f9c36e3f22d8eaf3a74f200e4479e91b24a7838a08a279d49157a47fab5 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644516947.0642586-1998819
Hash 1: 000005e16da5e94779bf6204d6a130890034ea768e6fd2534fea59eef2d2a0fe

Data 2: cf68956a00be594d489e4ecc99b8448ea69146a9d719822f365c35e159aadf1b - 000005e16da5e94779bf6204d6a130890034ea768e6fd2534fea59eef2d2a0fe-1644516956.8168552-194319
Hash 2: 00000bee467f1073fb31b92106fe4f09fc9641eff890e6f3df51086cbc00525a
```

Try 2:

```
nonce: 52566
block 1: User1 is TOP

nonce: 40372
block 2: User2 is TOP

after adding 2 blocks:
Data 1: 77c3d117908daeef33f5b354a9f41943b136a2bc84280a08f078bd31687444e0 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644517783.0614755-52566
Hash 1: 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408

Data 2: b49703c6ca457bc53cd4effcb278829f398ff9aba463fca49eb21b7a57a4afe5 - 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408-1644517783.3163407-40372
Hash 2: 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada
```

```
b = transactions.generate_block()

b1 = user1.create_block(b)
b2 = user2.create_block(b)
attacker.local_block_chain.chain.pop()

while(1):
    if attacker.local_block_chain.mineBlock(b1,51):
        user.set_BC_block_chain(attacker.local_block_chain)
        user1.set_BC_block_chain(attacker.local_block_chain)
        user2.set_BC_block_chain(attacker.local_block_chain)
        print("block 3 : Attacker is TOP")
        break
    elif user2.local_block_chain.mineBlock(b2,49):
        user.set_BC_block_chain(user2.local_block_chain)
        user1.set_BC_block_chain(user2.local_block_chain)
        attacker.set_BC_block_chain(user2.local_block_chain)
        print("block 3 : User2 is TOP")
        break

b = transactions.generate_block()

b1 = user1.create_block(b)
b2 = user2.create_block(b)

while(1):
    if attacker.local_block_chain.mineBlock(b1,51):
        user.set_BC_block_chain(attacker.local_block_chain)
        user1.set_BC_block_chain(attacker.local_block_chain)
        user2.set_BC_block_chain(attacker.local_block_chain)
        print("block 4 : Attacker is TOP")
        break
    elif user2.local_block_chain.mineBlock(b2,49):
        user.set_BC_block_chain(user2.local_block_chain)
        user1.set_BC_block_chain(user2.local_block_chain)
        attacker.set_BC_block_chain(user2.local_block_chain)
        print("block 4 : User2 is TOP")
        break
```

Attack:

Try 1:

```
nonce: 137534
block 3: Attacker is TOP

block chain:
Data 1: 571f7f9c36e3f22d8eaf3a74f200e4479e91b24a7838a08a279d49157a47fab5 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644516947.0642586-1998819
Hash 1: 000005e16da5e94779bf6204d6a130890034ea768e6fd2534fea59eef2d2a0fe

Data 2: cf68956a00be594d489e4ecc99b8448ea69146a9d719822f365c35e159aadf1b - 000005e16da5e94779bf6204d6a130890034ea768e6fd2534fea59eef2d2a0fe-1644516956.8168552-194319
Hash 2: 00000bee467f1073fb31b92106fe4f09fc9641eff890e6f3df51086cbc00525a

forked block chain:
Data 1: 571f7f9c36e3f22d8eaf3a74f200e4479e91b24a7838a08a279d49157a47fab5 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644516947.0642586-1998819
Hash 1: 000005e16da5e94779bf6204d6a130890034ea768e6fd2534fea59eef2d2a0fe

Data 2: b995b2f4d7303260f4b8897bebe67dff7d1f16f36a2ef72d10671f6e2d52c4f2 - 00000bee467f1073fb31b92106fe4f09fc9641eff890e6f3df51086cbc00525a-1644516957.7183414-137534
Hash 2: 00000d321075d78e5dc1d7a585157b6912c71066fa51ddbc5921ae6737f37ff2
```

Try 2:

Attacker chain before contesting

Attacker dropped the last block and tried to keep the chain

```
attacker:
Data 1: 77c3d117908daeef33f5b354a9f41943b136a2bc84280a08f078bd31687444e0 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644517783.0614755-52566
Hash 1: 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408
```

Notice the attacker was too slow to even fork the chain

```
nonce: 487984
block 3: User2 is TOP

block chain:
Data 1: 77c3d117908daeef33f5b354a9f41943b136a2bc84280a08f078bd31687444e0 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644517783.0614755-52566
Hash 1: 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408

Data 2: b49703c6ca457bc53cd4effcb278829f398ff9aba463fca49eb21b7a57a4afe5 - 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408-1644517783.3163407-40372
Hash 2: 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada

Data 3: a167006808e48efd54c76d69505e8e858ab93dcd60fb7b168c219324d179705e - 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada-1644517783.5077372-487984
Hash 3: 0000074634585b93919ea929dbacc37fccdebec6ec17471a5ebfa7bc7fff3f68

forked block chain:
Data 1: 77c3d117908daeef33f5b354a9f41943b136a2bc84280a08f078bd31687444e0 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644517783.0614755-52566
Hash 1: 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408

Data 2: b49703c6ca457bc53cd4effcb278829f398ff9aba463fca49eb21b7a57a4afe5 - 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408-1644517783.3163407-40372
Hash 2: 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada

Data 3: a167006808e48efd54c76d69505e8e858ab93dcd60fb7b168c219324d179705e - 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada-1644517783.5077372-487984
Hash 3: 0000074634585b93919ea929dbacc37fccdebec6ec17471a5ebfa7bc7fff3f68
```

```
b = transactions.generate_block()

b1 = user1.create_block(b)
b2 = user2.create_block(b)


while(1):
        if user1.local_block_chain.mineBlock(b1,51):
            user.set_BC_block_chain(user1.local_block_chain)
            attacker.set_BC_block_chain(user1.local_block_chain)
            user2.set_BC_block_chain(user1.local_block_chain)
            print("block 5 : User1 is TOP")
            break
        elif user2.local_block_chain.mineBlock(b2,49):
            user.set_BC_block_chain(user2.local_block_chain)
            user1.set_BC_block_chain(user2.local_block_chain)
            attacker.set_BC_block_chain(user2.local_block_chain)
            print("block 5 : User2 is TOP")
            break
```

Try 1:

The attacker successed

```
nonce: 112046
block 4: Attacker is TOP

after attack:
Data 1: 571f7f9c36e3f22d8eaf3a74f200e4479e91b24a7838a08a279d49157a47fab5 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644516947.0642586-1998819
Hash 1: 000005e16da5e94779bf6204d6a130890034ea768e6fd2534fea59eef2d2a0fe

Data 2: b995b2f4d7303260f4b8897bebe67dff7d1f16f36a2ef72d10671f6e2d52c4f2 - 00000bee467f1073fb31b92106fe4f09fc9641eff890e6f3df51086cbc00525a-1644516957.7183414-137534
Hash 2: 00000d321075d78e5dc1d7a585157b6912c71066fa51ddbc5921ae6737f37ff2

Data 3: abb0fe35bf4a3050218d71a39062a52e41393ea18aaef566c706304999ffe793 - 00000bee467f1073fb31b92106fe4f09fc9641eff890e6f3df51086cbc00525a-1644516958.3862674-112046
Hash 3: 0000085cc3925689b06524f1d532c5c00b139bcbce387d411909602b9b70b01c
```

```
nonce: 910059
block 5: User1 is TOP

trusted block chain:
Data 1: 571f7f9c36e3f22d8eaf3a74f200e4479e91b24a7838a08a279d49157a47fab5 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644516947.0642586-1998819
Hash 1: 000005e16da5e94779bf6204d6a130890034ea768e6fd2534fea59eef2d2a0fe

Data 2: b995b2f4d7303260f4b8897bebe67dff7d1f16f36a2ef72d10671f6e2d52c4f2 - 00000bee467f1073fb31b92106fe4f09fc9641eff890e6f3df51086cbc00525a-1644516957.7183414-137534
Hash 2: 00000d321075d78e5dc1d7a585157b6912c71066fa51ddbc5921ae6737f37ff2

Data 3: abb0fe35bf4a3050218d71a39062a52e41393ea18aaef566c706304999ffe793 - 00000bee467f1073fb31b92106fe4f09fc9641eff890e6f3df51086cbc00525a-1644516958.3862674-112046
Hash 3: 0000085cc3925689b06524f1d532c5c00b139bcbce387d411909602b9b70b01c

Data 4: a6ba737233d02e3e8197e884ec652025a9ed97e3889a126b579a6abcab6187fc - 0000085cc3925689b06524f1d532c5c00b139bcbce387d411909602b9b70b01c-1644516958.9103744-910059
Hash 4: 0000011dcb5a46736f809b443d11f2f3c382e84c74b7b888c26c1a7f3641a93e
```

Try 2:

The attacker failed so no blocks dropped

```
nonce: 344316
block 4: User2 is TOP

after attack:
Data 1: 77c3d117908daeef33f5b354a9f41943b136a2bc84280a08f078bd31687444e0 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644517783.0614755-52566
Hash 1: 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408

Data 2: b49703c6ca457bc53cd4effcb278829f398ff9aba463fca49eb21b7a57a4afe5 - 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408-1644517783.3163407-40372
Hash 2: 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada

Data 3: a167006808e48efd54c76d69505e8e858ab93dcd60fb7b168c219324d179705e - 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada-1644517783.5077372-487984
Hash 3: 0000074634585b93919ea929dbacc37fccdebec6ec17471a5ebfa7bc7fff3f68

Data 4: b8239e265f8c3d404c3f23b033623d056fae168d1c76008da875e7fe3b2eaa3b - 0000074634585b93919ea929dbacc37fccdebec6ec17471a5ebfa7bc7fff3f68-1644517785.6956205-344316
Hash 4: 00000d89ca197bd274a718f2c3f85bb02868f3580deebfa84729d800b29a028c
```

```
nonce: 621843
block 5: User1 is TOP

trusted block chain:
Data 1: 77c3d117908daeef33f5b354a9f41943b136a2bc84280a08f078bd31687444e0 - 103b6c7202d51787b8154af93a8f794498c5230f405c5b35663738274ee0391e-1644517783.0614755-52566
Hash 1: 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408

Data 2: b49703c6ca457bc53cd4effcb278829f398ff9aba463fca49eb21b7a57a4afe5 - 00000bd04068584e134d45d6963234824334cc2b25b5922d37965ee6d586c408-1644517783.3163407-40372
Hash 2: 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada

Data 3: a167006808e48efd54c76d69505e8e858ab93dcd60fb7b168c219324d179705e - 00000a5fb37a5284b0eacad8e64f6fac050e40bb698e085327482ee948112ada-1644517783.5077372-487984
Hash 3: 0000074634585b93919ea929dbacc37fccdebec6ec17471a5ebfa7bc7fff3f68

Data 4: b8239e265f8c3d404c3f23b033623d056fae168d1c76008da875e7fe3b2eaa3b - 0000074634585b93919ea929dbacc37fccdebec6ec17471a5ebfa7bc7fff3f68-1644517785.6956205-344316
Hash 4: 00000d89ca197bd274a718f2c3f85bb02868f3580deebfa84729d800b29a028c

Data 5: 936299a03fc95fc63803d07a9ade320f3bddef720d88bbcd04afa810a0e88e04 - 00000d89ca197bd274a718f2c3f85bb02868f3580deebfa84729d800b29a028c-1644517787.2642038-621843
Hash 5: 00000f1cde5441f7b979e05a55c6414b8af68cc3d323f731abec3123697e3d99
```

Time:

When Difficulty = 5

Avg time = 1.5 sec for one block

```
nonce: 441592
block 1: User1 is TOP

time taken to mine block = 2.1011180877685547 sec
```

```
nonce: 280125
block 1: User1 is TOP

time taken to mine block = 1.3382413387298584 sec
```

When Difficulty = 4

Avg time is 0.1 sec for one block

```
nonce: 23198
block 1: User1 is TOP

time taken to mine block = 0.1152350902557373 sec
```

```
nonce: 18728
block 1: User1 is TOP

time taken to mine block = 0.09182047843933105 sec
```