Module Code: CS3BC20
Assignment report title: Blockchain Coursework Assignment
Student Number: 26011251
Date of completion: 17/03/21
Actual hours spent for the assignment: 20

Practical 1 - Project Setup

a. Windows Forms and User Interface design

The user interface provides the means through which to interact with the blockchain. It has a central screen to display blockchain information, as well as surrounding buttons for user input.

b. Handling user input

The interface allows for several user interactions:

- Generation of a new wallet
- Validation of the public and private keys
- Checking the wallet coin balance
- Generation of a new block
- Accessing information for one block or the entire chain
- Creation of a transaction with an amount, fee, and receiver key
- Reading the list of pending transactions
- Changing the transaction type
- Validating the blockchain

Practical 2 - Blocks and Blockchain

a. Data-type justifications

Each block is created in the program as an object with several variables:

- The **'DateTime timestamp'** is for logging the time the block was created. Tracking the time of creation is important as the blocks are sequenced in chronological order.
- The **'int index'** is for logging the position of the block in the blockchain. It is an integer value as it is for indexing, and you can't have a fractional index.
- The **'static int difficulty'** is for setting the number of 0s preceding the hash value that the miner has to solve for. It is static so it can be referenced and changed through the user interface and is an integer value as you can't have a non-whole number of 0s.
- The **'prevHash'**, **'hash'**, **'merkleRoot'**, and **'minerAddress'** are all string values as they act as references and identities in each respective use.
- The **'transactionList'** is a list of Transaction objects used to store the transactions within the block
- The **'nonce'** is a long variable as it requires an extended size, the value of the nonce may be very high.
- The **'threadLock'** is an object for use in locking the variables used in threading so only one thread can access them at a time.
- The **'mineTime'** is a float as the time taken to mine is recorded in seconds and milliseconds/1000.
- The **'reward'** is a double so fractional values of the coins can be rewarded.

The blockchain also has several variables:

- The **'blocks'** list contains the blocks that form the blockchain
- The **'interactionsPerBlock'** is an integer value that sets how many transactions will be verified by a block when mined.
- The **'transactions pool'** is a list of Transaction objects that tracks the transactions that still need to be verified.
- The **'transactionType'** integer dictates how the blocks will choose which transactions from the transaction pool to verify.

b. Class hierarchy description

The blockchain is an object which stores two lists: the blocks that form the blockchain, and the newly created transactions that must be verified. Once a block is mined, the transactions are taken from this transaction object pool and stored within the block's own list of verified transactions objects.

c. Special properties of Genesis blocks

The genesis block is the first block within a blockchain. With an index of 0, it does not reference a previous block (no previous blocks exist).

d. Description of hashing and hash properties

Hashing is the mapping of data to a fixed-size key. In the context of a blockchain, the data within a block is represented by a hash. Hashes are deterministic, meaning that the same hash will always be returned by a distinct set of data. The data cannot be reverse engineered from the hash as the computational power required is too much for modern computers; hashing is therefore a key part of maintaining security and privacy within the blockchain network.

Practical 3 - Transactions and Digital Signatures

a. Key usage and mathematical relationship/properties

Blockchains make use of asymmetric cryptography for storing and transferring tokens. The private key is used to access the tokens of a wallet. Via elliptic curve multiplication, a public key is generated and then hashed to create an address. This address acts as an account number (string) for people to send tokens to. These mathematical functions only work one way, the public key cannot be calculated from the address, and the private key cannot be calculated from the public key; as the name suggests, the public key can be safely displayed in public, whereas the private key must remain unknown to all but the owner of the wallet.

b. Use in authentication of transactions

At the start of a transaction, the hash of the data to be sent and the sender's private key are used to produce a digital signature. This signature can then be recreated by the receiver's public key and data hash. Successfully recreating this digital signature verifies the transaction and allows it to complete.

c. Creating and managing Transaction Pools in Blockchains

After creating a transaction, it is placed into the transaction pool. Stored on the RAM of a network node, the transaction pool contains all the transactions waiting to be verified. When a block is mined, a limited number of transactions are taken from the pool and verified. For compensation, a fee is deducted from the tokens sent in the transaction, and rewarded to the miner.

Practical 4 - Consensus Algorithms (Proof-of-work)

a. Blockchain - Block relationship

The block is the object that, through linking with other blocks, form a blockchain. Each block makes reference to the previous block's hash (and so keeps a record of prior transactions). If the data within a block is changed (and thus its hash), each following block in the chain will become unverified. Each block is added without any transactions, mining the block will then pull a limited number of transactions from the transaction pool to be verified.

b. Block composition

Each block contains a header which stores the metadata for the block. The first item is the previous block's hash, which is then used to construct the block's own hash. This hash contains the timestamp of when it was created, the nonce, and the mining difficulty. The header also contains the merkle tree root which is used to summarise the transactions stored in the block.

c. Properties, Advantages, and Disadvantages of Proof-of-work

To decide who will create the next block in the chain, a hash code must be generated, with a number of preceding zeros equal to the difficulty of the block. Miners able to generate this hash will be placed into a lottery, the winner of which will create the next block and receive a set reward and the transaction fees.

One disadvantage of the algorithm is that by accounting for 51% of the hash rate, a single entity, or group of miners, may gain control of the network. However, since the computational power required to mine is so great, particularly for popular blockchain networks such as Bitcoin, the likelihood of this happening without a major party or government's intervention is low. Additionally, the high computational power required for mining deters spammers from trying to disrupt the network; wrong-doers would have more to gain from mining and supporting the network than from trying to disrupt it.

A major disadvantage of blockchain is the massive electricity use. The high computational power required to successfully mine Bitcoin and make a profit has necessitated the creation of increasingly vast mining farms. The cumulative demand of these miners has led to energy consumption 'greater than Argentina' [1]. Continuing into an increasingly climate conscious future without regulation or dramatic change seems highly unlikely for Bitcoin and other highly demanding blockchains.

d. Requirement for nonce in Blocks

The nonce is the number used to generate the hash during the mining process. The nonce is incremented until a hash is generated to satisfy the difficulty target.

e. Justification of value selected

The difficulty level is the number of zeros preceding the target hash. The difficulty is adjusted at fixed intervals to ensure a constant number of blocks is mined over a period of time. If mining takes longer than this time on average, the difficulty will lower, and if mining takes quicker than this time on average, the difficulty will increase; this change allows the network to adapt to an increasing or decreasing number of miners. The bitcoin network started at a difficulty level of 1 and as of writing is operating at a difficulty level of over 21 trillion [2].

f. Mining and Incentives: Driving transactions

Miners who successfully generate an appropriate hash and are fortunate enough to be selected to make the next block are rewarded with a set amount of tokens and the transaction fees. In the bitcoin network, if the miner can verify 1mb worth of transactions, as well as satisfy the target hash, they may earn 6.25 BTC. The number of new blocks mined and the reward are halved every four years to control the circulation of new coins. Though the last Bitcoin is predicted to be mined in 2140 [3], miners will still be incentivised to mine by the prospect of earning fees from validating transactions.

Practical 5 - Validation

a. How trustability is achieved

Trustability is created by the use of a public blockchain ledger. By keeping millions of copies of the transactions in the blockchain, any attempt to alter preexisting blocks may be detected. Through referencing the previous block's hash, each block can detect when a change has been made to the list of transactions. A distributed ledger ensures that the validity of the chain is not dictated by one group or user, but by the general consensus of all users.

b. Double spend prevention

Double spending is the concept of spending the same digital token more than once through duplication of counterfeit coins or falsification of transaction records. Double spending of blockchain tokens is prevented by the public ledger, as with each transaction recorded, the location of all tokens is known to the world, so any discrepancies can be easily detected and prevented.

c. Merkle root properties and benefits

The merkle root is the final hash in a tree of transactions known as the merkle tree. Each transaction is hashed with another to create a pair hash. These pair hashes are then hashed together, so on and so forth, until two hashes create the final merkle root hash. By creating a tree, only the branch containing the transaction needs to be checked when determining a transaction's validity. Moreover, hashing the transactions into a tree greatly reduces the file size, enabling storage within the header of each block; this allows for quick verification of transactions without having to download the entire blockchain.

d. Authenticity and Integrity achieved as a result of usage

Through validation of transactions via the maintenance of a public ledger, users of the blockchain can sleep easily knowing their tokens are secure. The ownership of a token is indisputable as its movement between owners is logged. Transactions are irreversible so users cannot fall victim to payment cancellation scams, though they can still be tricked into sending tokens to users under false pretence, as seen with the countless Bill Gates and Elon Musk scams on Twitter. Using digital signatures in transactions means only the intended recipient can receive the token, the transactions cannot be intercepted and manipulated.

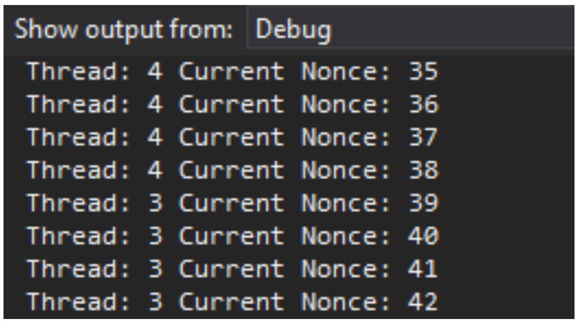
e. Incorporation of "rules"

Before a transaction can be verified, a set of protocol rules must be checked against. Examples include: making sure the sender and receiver addresses are included, the size of the transaction is smaller than the size of the block, the value is in a legal money range, the transaction has not already been verified, and the amount to send is not greater than the sender's wallet value [4]. Using a set of rules ensures consistency and efficiency, preventing mistakes that may occur without such regulation.

Assignment Task 1

In the basic configuration of the blockchain, when a new block is mined, the Mine( ) method will increment the nonce and generate a hash until the number of preceding zeros matches the difficulty. Through the implementation of threading, the time taken to find the correct nonce can be greatly reduced.

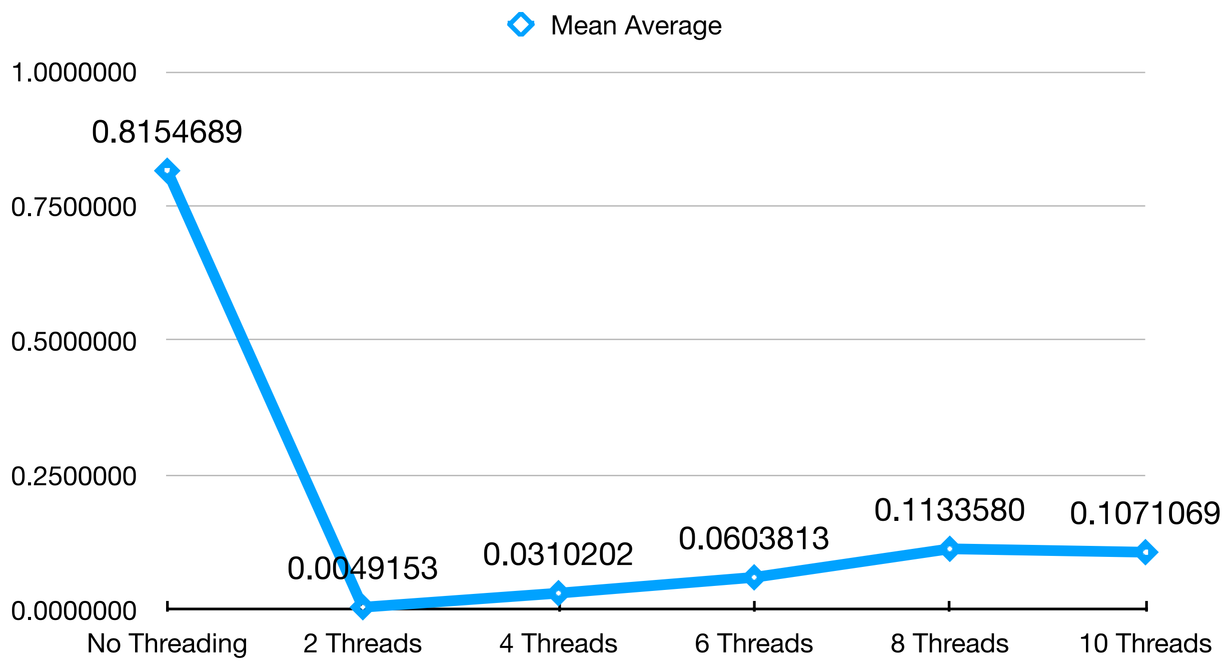
The incrementation of the nonce and generation of the hash has been moved to a separate method called 'findNonce', this method is assigned to each thread. A 'threadLock' object has been created and is locked by the method so that the nonce can only be accessed by one thread at a time, eliminating the potential for duplication of work. As shown in the screenshot, the nonce is passed over from one thread to the other (two threads in use).



```
Thread: 4 Current Nonce: 35
Thread: 4 Current Nonce: 36
Thread: 4 Current Nonce: 37
Thread: 4 Current Nonce: 38
Thread: 3 Current Nonce: 39
Thread: 3 Current Nonce: 40
Thread: 3 Current Nonce: 41
Thread: 3 Current Nonce: 42
```

Time to satisfy difficulty level (Find correct nonce)

	No Threading	**2 Threads**	**4 Threads**	**6 Threads**	**8 Threads**	**10 Threads**
	0.8299514	0.0070893	0.0136982	0.0479917	0.1133656	0.1223131
	0.5567837	0.0038442	0.0396941	0.0882558	0.1046595	0.1378698
	0.6381573	0.0044225	0.0307061	0.0793022	0.1658629	0.1549371
	1.3414332	0.0047567	0.0348948	0.0375960	0.0897843	0.0551951
	0.7110190	0.0044639	0.0361076	0.0487609	0.0931176	0.0652195
Mean Average	0.8154689	0.0049153	0.0310202	0.0603813	0.1133580	0.1071069



Without threading, it takes an average of just under one second to mine a block. Splitting the task up over two threads decreases the time taken to mine to only 0.0049153 seconds on average. Increasing the number of threads by two thereafter does not increase performance, in fact additional threads only serve to make the process slower; using ten threads is nearly 20 times slower than with two. Compared to without threading however, this is still a substantial improvement.

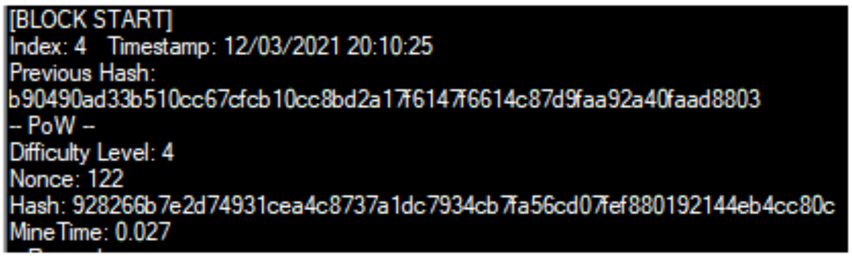
Task 2 - Adjusting Difficulty Level for Proof-of-Work

A blockchain's 'block time' is the expected time it will take to mine a new block; in the Bitcoin network this value is every ten minutes and in Ethereum every ten to twenty seconds [5]. In my implementation, the 'targetTime' is derived from 'BLOCK_GENERATION_INTERVAL' (how often a block should be found) multiplied by the 'DIFFICULTY_ADJUSTMENT_INTERVAL' (after how many new blocks the difficulty will be adjusted).

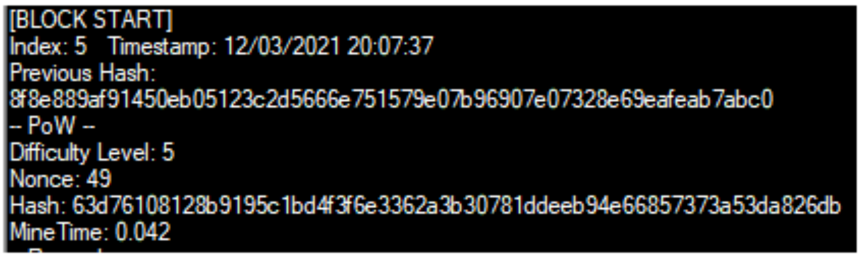
If the average time to mine n blocks is greater than twice the target time, the difficulty will increase by one. Else if the average time to mine n blocks is less half of the the target time, the difficulty will decrease by one. Though a rather basic implementation, it ensures that if it is too easy to mine (because of a good computer or many miners working together), mining will become harder. Conversely, if it is too hard to mine (because of a slow computer or too few miners working together), mining will become easier.

When the Bitcoin network was first established in 2009, the difficulty was set to only 1, nearly twelve years later the difficulty is over 21 trillion. A dynamic difficulty accounts for a changing number of miners operating on the network. Bitcoin's difficulty has increased greatly as it has become more and more popular (though more for its potential as a "get-rich-quick-scheme", rather than practical use as a day-to-day traded currency).

To showcase my implementation of a dynamic difficulty, the difficulty is adjusted after every five blocks, with new blocks expected to be found every 0.05 seconds. The screenshot shows that from the fifth block (index 4), to the sixth (index 5), the difficulty increases by one; the time to mine however only increases by around 1/2.



```
[BLOCK START]
Index: 4   Timestamp: 12/03/2021 20:10:25
Previous Hash:
b90490ad33b510cc67cfcb10cc8bd2a17f6147f6614c87d9faa92a40faad8803
-- PoW --
Difficulty Level: 4
Nonce: 122
Hash: 928266b7e2d74931cea4c8737a1dc7934cb7fa56cd07fef880192144eb4cc80c
MineTime: 0.027
```



```
[BLOCK START]
Index: 5   Timestamp: 12/03/2021 20:07:37
Previous Hash:
8f8e889af91450eb05123c2d5666e751579e07b96907e07328e69eafeab7abc0
-- PoW --
Difficulty Level: 5
Nonce: 49
Hash: 63d76108128b9195c1bd4f3f6e3362a3b30781ddeeb94e66857373a53da826db
MineTime: 0.042
```

Task 3 - Implementing Alternative Mining Preference Settings

Implementing mining preference settings enables the miner to choose which transactions to prioritise. With the default preference selected, enough transactions will be taken from the pool to fill the block, from index 0 to n transactions from the pool.

In greedy selection, the transactions with the highest fees will be selected for verification. The transaction pool list is sorted so the transactions with the highest fees are placed at the front of the queue. This preference maximises the reward a miner can make though it means that if large transactions keep getting added to the pool, the smaller transactions will remain unverified for a long time.

In altruistic selection, the transactions waiting in the pool for the longest will be selected first. The transaction pool list is sorted so the transactions with the earliest timestamp are placed at the front of the queue. This preference benefits the participants of a transaction as it prioritises verifying each transaction as quickly as possible. Ultimately, for a blockchain network such as Bitcoin to be used as a day to day currency, the speed at which a transaction takes place must be prioritised. As it stands, the Bitcoin network takes far too long to process transactions, particularly for instances in which tokens need to be transferred quickly.

In random selection, the transactions are randomly selected from the transaction pool. A transaction is randomly selected and removed from the transaction pool until the block is full. Using random selection eliminates bias in the selection process, though through doing so it acts as a compromise that doesn't benefit any particular party.

In address preference selection, the miner prioritises transactions in which they are a participant. The sender and recipient address of each transaction are looked through until the miner's address is found, at which point the transaction is selected. This selection method encourages miners to participate for their own benefit as the transaction fees are reimbursed to the miner, however this penalises transactions made by users who do not mine; most of the general public do not mine blockchain tokens so only a small group of people would benefit from this selection method.

Bibliography:
1. Cristina Criddle Bitcoin consumes 'more energy than Argentina' 10/02/21
2. Current Bitcoin Difficulty
3. Adam Hayes What Happens to Bitcoin After All 21 Million Are Mined 28/02/21
4. Protocol Rules
5. Prabath Siriwardena The Mystery Behind Block Time 15/10/17