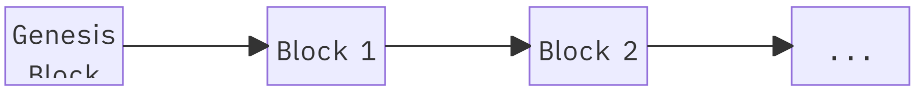
Assignment 2

Theory and Applications of Blockchain

Deadline: 21 August 2025

Problem 1 [60]



Design a simple blockchain that starts with Genesis Block and grows by mining new blocks. The implementation should be done as a web application where the server handles the code for mining each block and the client simply displays blocks being added to the blockchain. You are free to choose how to display addition of a block.

The client contains a button called *Mine* that requests the server to mine a block and returns the response. The block is then shown in the client.

Each block display the following:

```
Index: 0 # The index of the block
Timestamp: # Date and time when this block was mined.
Previous Hash: # The SHA-256 hash of the previous block.
Hash: # The SHA-256 hash of the current block.
Data: # The content of the block. Can contain some transactions.
Nonce: # The number that was used to solve the puzzle.
```

- You may use either Python or JavaScript for this implementation.
- The *previous hash* of genesis block is `0`.

The following shows a starter code written in JavaScript which you do **not** necessarily need to follow.

A sample implementation can use Node.js and use the libraries *express* and *ejs* in the server.

```
// 'main.js' in the server
class Block {
    constructor(index, timestamp, data, prevHash = 0) {
        this.index = index;
        this.timestamp = timestamp;
        this.data = data;
        this.prevHash = prevHash;
        this.nonce = 0;
        this.hash = this.getHashOfBlock();
        // ...
    }

    getHashOfBlock() {
    // Create hash of the block using index, timestamp, data,
nonce
    }
}

function findProofOfWork(block, difficulty) {
    /**
    block: A block (Block)
    difficulty: A number representing how many leading
    zeros there should be in the hash

    Takes a block, difficulty and finds the 'curNonce' that
    satisfies the difficulty.
    **/
    let curNonce = 0;
}

class Blockchain {
...
}
```

Problem 2 [40]

- Add a button *Download* that can be used to download the values shown in the entire blockchain as a JSON/YAML/TXT file.
- Add a button *Validate* that takes (uploads) a downloaded blockchain and validates the entire chain. It also shows the created blockchain.
- You may make some random edits to the downloaded file and check your implementation. When clicked, the block where the edit was made and all following blocks become invalid. Be creative how to show invalidated blocks in the UI.

Submission Instructions

- Submitting *AI-generated code* is **strictly prohibited**. Any such submissions will automatically attract penalty.
- Create one file called *ass_2_TeamName.zip* and upload to Moodle.
- This is a team submission so make sure each team submits **one and only** zip file. Make sure only one member from the team does this.
- The *zip* file should contain:
    - *README.md* mentioning
        - The technologies and programming language used.
        - Clearly mentioning the command to start the server.
        - The URL to access the client.
    - All corresponding code required to execute your application locally.