

CPSC 350: Data Structures and Algorithms

Spring 2025

Programming Assignment 4: Blockchain Ledger

Due: See Canvas for due date

The Assignment

In this assignment, you will build a simplified blockchain from scratch. A blockchain is a linked series of blocks where each block holds a set of transactions (or other data) along with a [cryptographic hash](#) that ties it to the previous block. This structure ensures the integrity and immutability of the entire chain. Your task is to design a custom data structure that supports basic blockchain operations and validates its integrity.

Background on Blockchain Technology

A blockchain is essentially a chain of blocks where each block contains:

- **Index:** The position of the block in the chain.
- **Timestamp:** The time when the block was created.
- **Data:** The information stored in the block (i.e., transactions).
- **Previous Hash:** The hash value of the previous block in the chain.
- **Current Hash:** A hash computed from the block's own content (including the index, timestamp, data, and previous hash).

Each block stores the hash of the block before it. This means that if any block is changed, its hash will also change, breaking the chain and exposing any tampering. Since every block is linked by these hashes, the entire blockchain stays secure—modifying one block would force you to update all following blocks. In real-world systems, many nodes work together to agree on the blockchain's state. For this assignment, just make sure that each block correctly links to the one before it.

More info [here](#).

The Design

Your program will simulate a blockchain ledger with the following operations:

Block Class:

- Implement a Block class with the following attributes:
 - **Index:** Position in the blockchain.
 - **Timestamp:** Time of block creation.

- Reference for getting a timestamp in C++:
https://www.w3schools.com/cpp/cpp_date.asp
- **Data:** A string representing the block's content (i.e., transaction).
- **Previous Hash:** The hash of the previous block.
- **Current Hash:** The hash computed from the block's content.

Blockchain Class:

- Implement a Blockchain class to manage the chain of blocks. It should:
 - Start with a **Genesis Block** in the constructor.
 - The Genesis Block is the first block of the chain and it is made with a predefined value. E.g. current time stamp and "Genesis Block" as the transaction data.
 - Provide a method to **add a new block**. This method must compute the new block's hash using your custom hash function and link it to the previous block.
 - Provide a method to **validate the chain** by ensuring:
 - Each block's stored previous hash matches the computed hash of the preceding block.
 - Each block's current hash is correctly computed based on its contents.
 - **File Input/Output Operations:**
 - **Save the Blockchain to a File:** Write the entire blockchain to an output file (e.g., each block on a separate line with fields separated by a pipe (|)).
 - **Load the Blockchain from a File:** Read the blockchain state from an input file to reconstruct the chain, thereby allowing the blockchain to be restored between program executions.
 - Sample blockchain file format:

0|Sat Mar 22 06:15:37 2025|Genesis Block|0|ad8

1|Sat Mar 22 06:15:41 2025|Elia sent Erik 1000 Panther Coins|ad8|11b7

2|Sat Mar 22 06:15:54 2025|Erik sent Elizabeth 88 Panther Coins|11b7|1386

3|Sat Mar 22 06:16:17 2025|Elizabeth sent Elia 45 Panther Coins|1386|1347

User Interface Class & Main Method:

- Develop a simple console-based interface that allows the user to:
 - **Add a New Block:** Prompt the user for data (e.g., a transaction or message) and create a new block with the current timestamp.
 - **Display the Blockchain:** Print out each block's details in order.
 - **Validate the Blockchain:** Run a chain integrity check and report whether the blockchain is valid.

- **Save Blockchain to File:** Allow the user to save the current state of the blockchain to an output file.
- **Load Blockchain from File:** Allow the user to load a previously saved blockchain state from an input file.
- **Exit**

Hash Function Instructions

For this assignment, you will implement a simplified hash function. Your hash function should:


1. Take the block's content by concatenating the index, timestamp, data, and previous hash into one string.
2. Convert each character in that string to its corresponding ASCII integer value.
3. Sum the ASCII values to produce a final numeric value.
4. Convert the final numeric value into a hexadecimal string, which will serve as the block's hash.
 - a. You may use the following code to turn your number into hex

```
#include <iostream>
#include <sstream>
#include <iomanip>
#include <string>

std::string intToHexStream(int num) {
    std::stringstream ss;
    ss << std::hex << num;
    return ss.str();
}

main() {
    unsigned int finalNumericValue = 305441741;
    std::string hash =
        intToHexStream(finalNumericValue);
    std::cout << "Hexadecimal Hash: " << hash <<
        std::endl;
    return 0;
}
```

Sample Input and Output

 Programming Assignment 4: Blockchain Ledger Sample Input and Output

Rules of Engagement

- You may **NOT** use any non-primitive data structures other than what we have implemented in class. You may not use any data structures from the C++ STL. Of course, to do the file processing you may use any of the standard C++ IO classes.

- For this assignment, you must work **individually**.
- Develop using VSCode and make sure your code runs correctly with g++ using the course docker container.
- Feel free to use whatever textbooks or Internet sites you want to refresh your memory with C++ IO operations, just cite them in a README file turned in with your code. All code you write, of course, must be your own. In your README please be sure to include the g++ command for compiling your code.

Due Date

This assignment is due at 11:59 pm on see Canvas for due date. Submit all your commented code as a zip file to Canvas. The name of the zip file should be LastName_FirstInitial_A4.zip

Grading

Please fill out [this form](#) if you have any inquiries on your PA grades! This will go to our grader. If you have a personal question, please message your instructor first and they'll let you know how to proceed.

Grades will be based on correctness, adherence to the guidelines, and code quality (including the presence of meaningful comments). An elegant, OO solution will receive much more credit than procedural spaghetti code. I assume you are familiar with the standard style guide for C++, which you should follow. (See the course page on Canvas for a C++ style guide and Coding Documentation Requirements.)

Code that does not follow the specification EXACTLY will receive an automatic 25% deduction. Code that does not compile will receive an automatic 50% deduction.

Readme & References

More info on Canvas!

- **README file:** All source code will be accompanied with a plain text README file. We encourage students to take advantage of markdown technology if they are inclined to do so. This file will contain:
 - The following identifying information:
 - Full name
 - Student ID
 - Chapman email
 - Course number and section
 - Assignment or exercise number
 - A list of all source files submitted for the assignment
 - A description of any known compile or runtime errors, code limitations, or deviations from the assignment specification (if applicable)

- A list of all references used to complete the assignment, including peers (if applicable)
- Instructions for running the assignment. (Typically applicable in advanced courses using build systems or third party libraries) for us it would be something like:
 - To compile: `g++ *.cpp -o A4.exe`
 - To run: `./A4.exe`
- **In-code citations:** You are allowed to use small, isolated lines of code from external sources in your programming assignments as long as they are appropriately cited. Any time you are including code that you did not write yourself, it must be cited. This includes sources from StackOverflow and similar forums, other current or previous students, tutors, books, tutorials, etc. You should wrap the copied code in a comment denoting the start and end of said code, like this:


```
# code that is not copied (you wrote this)
someCode = 0;
print("hi I wrote this!")

"""BEGIN CODE FROM SOURCE: link/name of source"""
print("I did not write this");
"""END OF CODE FROM SOURCE: link/name of source"""

"""BEGIN CODE FROM CHAT GPT, PROMPT ASKED: how do you
...? """
print ("I did not write this");
"""END OF CODE FROM CHAT GPT"""

# back to code that you personally wrote
codeIWrote = 10;
```
- **Info in code files:** Additionally, all source files will start with a header comment containing the following items (one per line):
 - Full name
 - Student ID
 - Chapman email
 - Course number and section
 - Assignment or exercise number