Assignment 2: Implementing a Blockchain

```
Course:              INTE264[1|2] | Blockchain Technology Fundamentals
Presentation Date:   Tutorial Week 8 - Thursday August 21, 2025
Due Date:            End of Week 8 – Friday August 22, 2025 @noon
Weighting:           50 points total; contributes 30% to your final grade
Submission:          Online via Canvas
```

This assignment requires you to design, implement, and report on your own functional blockchain. This is a more intensive project than Assignment 1, demanding a deeper application of blockchain principles and more extensive programming.

Overview

You are to programmatically implement a working blockchain. You have considerable freedom in your design choices (e.g., consensus mechanism, specific features), but your implementation must meet a set of core functional requirements detailed below. Your system should be capable of creating blocks, linking them into a chain, handling (simulated) transactions or data, and operating under a chosen consensus rule.

💻 Programming Requirements

- You may use any programming language of your choice (e.g., Python, Rust, C++, Go).
- Your code must be well-commented, explaining the logic and functionality of all significant parts. Clear comments are essential for understanding your implementation.
- You must submit your source code. This can be done by either:
  - Providing a link to a public repository (e.g., GitHub, GitLab) where your code is hosted. Ensure the repository is accessible.
  - Submitting a link to a shared folder with your code.
- Your code should compile/run and demonstrate the required functionality.

✍️ Written Report

Alongside your functional blockchain implementation, you will submit a written report. This report is a should include:

- System Design & Architecture: Detail the design of your blockchain, including data structures, chosen algorithms, and overall architecture. Justify your design choices.
- Implementation Details: Explain how you implemented the core requirements and any additional features. You can include key code snippets (properly formatted and referenced) to illustrate complex parts, but the bulk of the code should be in the submitted source files.
- Your responses should be researched and clearly articulated.

🤖 Use of Chatbots/LLMs

You are permitted and encouraged to use Large Language Models (LLMs) as a tool for research, understanding concepts, debugging code, and drafting your written explanations. However:

- Your final submitted code and written work must reflect your own understanding and effort.

- You must cite any significant use of LLMs. This includes specifying which LLM was used and for what purpose (e.g., "Used ChatGPT-4/Gemini to help debug the Merkle tree construction algorithm and to draft the initial explanation of pre-image resistance, which was then revised."). A simple statement at the end of each question is sufficient. Failure to cite LLM usage appropriately when it has been significantly used may be considered academic misconduct.

📽️ Tutorial Presentation Requirement

In your tutorial session during Week 8 (August 21), you will be required to present your chosen solution (approximately 5-7 minutes). This presentation should include a brief demonstration or walkthrough of your functioning blockchain, highlighting key features such as block creation, transaction handling, your consensus mechanism in action, and double-spend prevention. Note the tutorial presentation is about 23 hrs before the due date and therefore your solution might only be partially complete. This presentation is a mandatory component of the assignment. While the presentation itself will not receive a separate grade, failure to complete this presentation requirement will result in a penalty of 10% deducted from the total mark awarded for this written assignment.

📬 Submission Guidelines

- Written Report: Submit your answers (explanations and analysis for your three chosen questions) as a single PDF document via Canvas. There is no length requirement. Email submissions will not be accepted.
- Source Code:
  - If using a public repository: Include a clearly marked section in your PDF document with the URL(s) to your public repository (e.g., GitHub). Ensure the link is correct and the repository is publicly accessible. Include a README in your repository if necessary to explain how to run your code.
  - Do not submit files directly, rather, include a link to a shared folder that contains your code.
- Ensure your name and student ID are clearly stated on the first page of your PDF document.
- Use clear headings and sub-headings for each question and its parts.
- Cite any external sources (academic papers, technical articles, specific LLM assistance) appropriately using a consistent referencing style (e.g., IEEE).

🏆 Grading Rubric

The rubric (on Canvas) will be applied to your submission.

This assignment is designed to be challenging and rewarding. Plan your time carefully, start early, and don't hesitate to consult course materials and seek clarification on concepts. Good luck building your blockchain!

🛠️ Blockchain Requirements

Your blockchain implementation must satisfy the following requirements (1-8). The sophistication of each implementation detail will contribute to your grade. These requirements generally build upon each other. It is recommended that you start at step 1, implement the block structure, and you may reuse elements from Assignment 1 as you go. Requirements 9 and 10 are optional extensions that can earn bonus points / contribute to higher grades.

1. **Block Structure [10 points]**
    - Define a clear and robust block structure. Each block must contain at least:
        - A unique identifier (e.g., block index/height).
        - A timestamp (indicating when the block was created/validated).
        - Data payload (e.g., a list of transactions, or other forms of data).
        - The hash of the previous block (to ensure chain linkage).
        - Its own cryptographic hash (calculated from all critical block contents, including the previous block's hash).
        - Fields relevant to your chosen consensus mechanism (e.g., nonce for PoW, validator signature for PoA).

2. **Cryptographic Hashing & Chain Integrity [10 points]**
    - Blocks must be securely linked using cryptographic hashes. The hash of each block must depend on the hash of the previous block.
    - Implement functionality to calculate and verify block hashes.
    - Demonstrate immutability: Show (either programmatically or through clear explanation and testing steps in your report) that if data in a previous block is tampered with, it invalidates the hash of that block and all subsequent blocks in the chain.

3. **Transaction Handling (or Data Management) [6 points]**
    - Implement a mechanism to create and include transactions (or generic data entries) within blocks.
    - Transactions should be part of the data that is hashed to form the block's hash (e.g., by including a Merkle root of transactions, or hashing a serialized list of transactions).
    - Maintain a pool of pending transactions if your consensus mechanism involves selecting transactions for a new block (e.g., miners selecting from a mempool).

4. **Consensus Mechanism [6 points]**
    - Choose, implement, and justify a consensus mechanism to govern how new blocks are created, validated, and added to the chain. Options include (but are not limited to):
        - Proof-of-Work (PoW): Must include a mining process, a nonce, a difficulty target, and a mechanism for difficulty adjustment (even if simplified).
        - Proof-of-Stake (PoS) (Simplified): Must include a way to simulate stake, a mechanism for validator selection based on stake (can be simplified), and block validation by chosen stakers.
        - Proof-of-Authority (PoA) (Simplified): Must include a predefined set of authorized signers/validators and a mechanism for them to propose and validate blocks.
        - Other recognized or justified consensus algorithms.

- The process of achieving consensus (e.g., finding a nonce in PoW) must be clearly demonstrable.

5. **Double-Spend Prevention [6 points]**
   - Implement and clearly demonstrate a mechanism to prevent the same digital asset (even if simulated as simple balances or unique transaction IDs) from being spent more than once.
   - This could involve:
     - Checking transaction history for conflicting spends.
     - Implementing a UTXO (Unspent Transaction Output) model.
     - Other valid approaches appropriate to your blockchain design.
   - You must describe in your report how a user would attempt a double spend and how your system prevents it.

6. **Global Ordering of Blocks [6 points]**
   - The blockchain must maintain a clear, chronologically consistent global ordering of blocks, primarily enforced by timestamps and the chain structure.
   - The consensus mechanism should ensure that valid blocks are added in an orderly fashion.

7. **Data Persistence [3 points]**
   - The blockchain (the chain of blocks and any relevant state like account balances or UTXO sets) must be persistable to storage (e.g., file(s) on disk, a simple database).
   - The system should be able to reload the blockchain state from persistent storage upon restarting.

8. **Basic User Interface [3 points]**
   - Provide a simple way for a user to interact with your blockchain. This could be:
     - A command-line interface (CLI)
     - A web interface
   - Functionality should include, as appropriate for your design:
     - Creating and submitting new transactions/data.
     - Initiating the block creation/mining/validation process.
     - Viewing the contents of blocks and the overall chain.
     - Checking balances or querying data (if applicable).

9. **Simplified Peer-to-Peer (P2P) Networking [Optional Extension +4 points]**
   - This is an optional feature that can contribute to higher grades.
   - Simulate basic network interactions between a few nodes (can run as separate processes/threads on the same machine). This could include:
     - Broadcasting new transactions to other nodes.
     - Broadcasting newly created blocks to other nodes.
     - A simple chain synchronization mechanism (e.g., nodes adopting the longest valid chain they are aware of).

10. **Wallet Functionality [Optional Extension +4 points]**
    - This is an optional feature that can contribute to higher grades.

- Implement basic generation of public/private key pairs for users, signing transactions with private keys, and verifying signatures with public keys. This builds upon concepts from Assignment 1.