Assignment 1: Implementing Core Blockchain Components

```
Course:              INTE264[1|2] | Blockchain Technology Fundamentals
Presentation Date:   Tutorial Week 4 - Thursday July 24, 2025
Due Date:            End of Week 4 — Friday July 25, 2025 @noon
Weighting:           30 point total; contributes 20% to your final grade
Submission:          Online via Canvas
```

This assignment requires you to implement and explore fundamental cryptographic and data structuring techniques that underpin blockchain technology. You will *choose three (3) out of the following four (4)* questions to answer. Each question carries equal weight and has a significant programming component.

💻 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

For each chosen question, alongside your code, you will submit a written component. This should:

- Briefly explain the functionality of your implemented code.
- Discuss the underlying concepts related to the programming task.
- Critically analyze the role and significance of these components in the context of blockchain systems.
- 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 4 (July 24), you will be required to present one (1) of your chosen solutions (approximately 5-7 minutes), focusing on a brief demonstration or walkthrough of your code and its output, followed by a concise explanation of the core concepts. 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 each of the three questions you choose to answer. The final mark for the assignment will be an aggregation of the marks for the three chosen questions.

Remember to select your best three answers, implement the programming tasks thoroughly, and prepare to discuss one of your solutions in your tutorial. Good luck!

Question 1: Hash Functions [10 points]

This question explores cryptographic hash functions, their fundamental properties, and their critical role in ensuring the integrity and security of blockchain systems. You will implement programs to demonstrate these properties and analyze their significance.

Requirements for Question 1:

- Your written analysis should clearly link your code's output to the theoretical properties of hash functions.
- Provide clear explanations of hash function properties, their roles in blockchain, and potential vulnerabilities if these properties were compromised. Use specific blockchain examples.

1A. Programming Task: Demonstrating Hash Function Properties

i. Using a standard cryptographic hash function library available in your chosen programming language, write a program that:

1. Takes an arbitrary string as input from the user.
2. Computes and displays its hash.
3. Demonstrates the avalanche effect: Take the original input string, make a minimal change (e.g., flip one bit, change one character), compute the hash of this modified string, and display both the original and modified hashes. Highlight or calculate the difference (e.g., Hamming distance if feasible, or visually show they are very different).

ii. Write code to demonstrate the difficulty of finding a pre-image. Take a target hash output (this can be the hash of a known, short string you define in your code). Write a loop that iteratively hashes random or sequential input strings and compares the output to your target hash. Count the number of attempts. Your program should run for a limited number of iterations or a short period and report if the pre-image was found (it is highly unlikely for secure hashes) and the number of attempts made.

1B. Written Analysis

i. Explain the cryptographic hash function you used in your program.

ii. Define and explain the three fundamental properties of cryptographic hash functions: 1) Pre-image resistance, 2) Second pre-image resistance, and 3) Collision resistance. Relate these properties to what your code demonstrates. Specifically, discuss how your pre-image finding attempt illustrates pre-image resistance.

iii. Discuss at least two distinct roles hash functions play in blockchain systems, explaining how their properties are crucial for these roles.

iv. If a practical method to find collisions for the hash function you used were discovered, discuss two specific security vulnerabilities that could arise in a blockchain system.

Question 2: Merkle Trees [10 points]

This question focuses on Merkle Trees, a hierarchical data structure crucial for efficient data verification in blockchains. You will implement the construction of a Merkle Tree, along with the generation and verification of Merkle proofs.

Requirements for Question 2:

- Your written analysis should clearly explain your implementation, the structure of Merkle Trees, the concept of Merkle proofs, and their significance in blockchain, particularly for SPV clients.
- Diagrams illustrating your tree structure and an example proof are highly encouraged to support your explanations.

2A. Programming Task: Merkle Tree Construction and Proof Generation

i. Write a program that takes a list of data items (e.g., strings representing transaction IDs) as input, and constructs a Merkle Tree from this list of data items. Your implementation should correctly handle cases with an odd number of leaf nodes at any level. The function should output the Merkle root.

ii. Implement a function that, given the original list of data items and one specific item from that list, generates a Merkle proof (the list of hashes and their positions – left/right – needed to reconstruct the Merkle root for that specific item).

iii. Implement a function that takes a Merkle proof, the specific data item, and the Merkle root, and verifies whether the proof is valid (i.e., if the item, when combined with the proof hashes, correctly reconstructs the given Merkle root).

2B. Written Analysis

i. Explain the structure of your Merkle Tree implementation and the logic behind your construction, proof generation, and proof verification functions. You can use diagrams to illustrate your tree structure.

ii. Discuss how Merkle Trees contribute to the efficiency and data integrity of blockchain blocks.

iii. Explain the concept of a Merkle proof and how it enables Simplified Payment Verification (SPV) clients in a blockchain to efficiently verify transaction inclusion without downloading the entire blockchain.

Question 3: Digital Signatures [10 points]

This question involves Public Key Cryptography (PKC) and its application in creating and verifying digital signatures, key for transaction authorization and establishing ownership in blockchain systems. You will implement a basic digital signature scheme.

Requirements for Question 3:

- Your written analysis must clearly explain your code, the underlying cryptographic principles of PKC and digital signatures, their security properties, and their applications in blockchain.

3A. Programming Task: Creating and Verifying Digital Signatures

i. Using a cryptographic library that supports Public Key Cryptography (e.g., RSA or ECC), write a program that:

1. Generates a public-private key pair.
2. Takes a message string as input.
3. Signs the message using the private key to create a digital signature.
4. Verifies the signature against the original message using the public key.
   ii. Your program should clearly output the public key, private key (for illustrative purposes in this academic context), the original message, the generated signature, and the validation result.

3B. Written Analysis

i. What are the fundamental principles of Public Key Cryptography (PKC)? Contrast it with Symmetric Key Cryptography.

ii. Explain the core components of your program: key generation, the signing process (including hashing the message before signing if applicable by the library/standard), and the verification process.

iii. Detail what a digital signature is and enumerate the security properties it provides. Explain how your implementation (and PKC in general) achieves these properties.

iv. Discuss at least two critical applications of PKC and digital signatures in blockchain systems.

Question 4: Simulating Basic Timestamping in a Chain-of-Blocks [10 points]

This question involves creating a simplified simulation of a blockchain by defining a block structure and linking blocks together in a chain, with a focus on the role of timestamping in establishing temporal order and contributing to the ledger's integrity. *Note this question will be good preparation for Assignment 2.*

Requirements for Question 4:

- Your written analysis should explain your simulation's logic, connect it to the concept and importance of timestamping in actual blockchain systems, and discuss its role in ensuring properties like immutability and preventing double-spending.

4A. Programming Task: Timestamped Blockchain

i. Define a simple data structure or class for a "Block". Each block should at least contain:

1. block ID
2. timestamp
3. data
4. previous hash
5. nonce

ii. Write a program that simulates the creation of a chain of blocks:

1. The first block (genesis block) can have a "previous hash" of "0" or a predefined string.
2. Each subsequent block must have a blockcounter that increases and contain the hash of the previous block.
3. Create dummy data to go in each block.
4. Choose a method to 'delay' the creation of new blocks. Note that Assignment 2 will focus on blockchain consensus, you do not need to implement it here.
5. Your program should print out the details of each block in the chain in a clear format, showing all its fields including its own hash and the previous hash, thus demonstrating the linkage.

4B. Written Analysis:

i. Explain your Block data structure and the logic your program uses to create blocks, calculate their hashes, and link them into a chain, ensuring chronological timestamping.
ii. Describe how time stamping is implemented in real blockchain technologies (e.g., in block headers by miners/validators) and its significance for achieving properties such as immutability (tamper-evidence) and traceability.
iii. Discuss the role of time stamping in preventing issues like double-spending (conceptually, as your simple model won't implement full consensus or transaction validation, but explain how ordering helps).
iv. Compare and contrast decentralized time stamping in blockchains with traditional centralized time stamping methods, discussing advantages and challenges.