Test Assignment: Solana Program with Merkle Tree Implementation

Overview

Your task is to implement a **Solana program** that includes basic operations for a **Merkle tree**. The program should support:

1. **Inserting a leaf node** into the Merkle tree.
2. **Emitting an event message** with the updated Merkle tree root after each insertion.

This assignment is designed to test your understanding of:

- Rust programming in the context of the Solana blockchain.
- Efficient state handling and serialization/deserialization.
- Hashing and binary tree concepts (Merkle tree).
- Solana-specific mechanisms for logging and events.

Requirements

1. **Solana Program**:
    - Use vanilla solana library
    - Define accounts to store the Merkle tree state, including the root hash and leaf nodes.
    - Implement functionality to:
        - Insert a new leaf into the Merkle tree.
        - Recalculate the root hash after each insertion.
        - Emit an event message with the updated root hash.
2. **Client-side Interaction**:
    - Create a simple Rust-based client program to interact with the Solana program.
    - The client should:
        - Insert a leaf node into the Merkle tree by sending a transaction.
        - Read event messages to verify the updated root hash.
3. **Unit Tests**:
    - Write unit tests for the on-chain program to validate the correct behavior of:
        - Leaf insertion.
        - Root hash updates.
        - Event message emission.

Implementation Details

1. **Hashing**:
    - Use a hashing algorithm (e.g., SHA-256) to compute the Merkle tree hashes.
2. **State Management**:
    - Store the Merkle tree state in a Solana program account.
    - The account should include:
        - **The current root hash.**
        - **A list of all leaf hashes.**
3. **Serialization**:
    - Use a serialization library like `borsh` or `serde` for account data.
4. **Instructions**:
    - Implement one instruction:
        - `InsertLeaf`: Insert a new leaf into the Merkle tree, update the root, and emit an event.
5. **Events**:
    - Use Solana's logging mechanism (`msg!`) to emit a message containing the updated root hash.
6. **Testing Framework**:
    - Use `cargo test` or `solana-program-test` to write tests for your program.

Notes

1. This assignment will be a part of the technical interview process. Your submission will be reviewed for code quality, correctness, and adherence to best practices in the Solana ecosystem. Be prepared to discuss your implementation and thought process during the interview.
2. You can write this code in any way you prefer. However, even if you generate this code using large language models (LLMs), you should have 100% ownership of it and be able to fully explain and justify all aspects of your implementation.