TREASURE HUNT

Objective:
Create a smart contract for a simple on-chain game called "Treasure Hunt," where multiple players can participate. The goal of the game is to find the hidden treasure, but the twist is that the location of the treasure is determined dynamically based on player interactions and a set of predefined rules.

Rules:

1. **Grid Setup:** The game operates on a 10x10 grid. Each grid position has a unique identifier (0 to 99).
2. **Players:** Any Ethereum address can participate by sending a small amount of ETH to the contract. Each player can only move once per turn.
3. **Treasure:** The treasure starts at a random position on the grid. The initial position is determined by the hash of the block number when the contract is deployed.
4. **Movement:**
    - Players can move to any adjacent grid position (up, down, left, right) from their current position.
    - The treasure moves each time a player makes a move. The movement of the treasure is determined by a rule:
    - If a player moves to a grid position that is a multiple of 5, the treasure moves to a random adjacent position.
    - If a player moves to a grid position that is a prime number, the treasure jumps to a new random position on the grid.
5. Winning Condition: A player wins if they move to the grid position where the treasure is located.
6. Reward: The winner receives 90% of the contract's ETH balance, and the remaining 10% is locked for future game rounds.

Additional Requirements:

• **Testing:** Include a suite of unit tests to demonstrate the correctness of the contract, especially around edge cases (e.g., treasure movement when multiple conditions are met).

Hints:

• Consider using modular arithmetic to manage the grid and movement.
• Think carefully about how you can create a secure and fair source of randomness.
• Be mindful of gas costs and how to minimize them.

Deliverables:

1. The Solidity contract code.
2. A set of unit tests in JavaScript/TypeScript using Hardhat or Truffle.
3. A short explanation of the design choices, especially how the randomness is handled.
4. Make sure your deliverables are submitted and documented via a GitHub repository.