Solidity Smart Contract Development

1 Introduction

Solidity is a versatile programming language used for creating smart contracts on the Ethereum blockchain. Smart contracts are self-executing programs that automatically enforce predefined rules and agreements, enabling trustless and automated transactions. In this lab assignment, you will explore the fundamentals of Solidity and Ethereum and then choose one of several interesting problems to solve by developing a Solidity smart contract.

2 Solidity and Ethereum Basics

Before you delve into the problems, let's recap some important concepts:

2.1 Solidity

1. **What is Solidity?** Solidity is a statically-typed, high-level programming language specifically designed for smart contract development on Ethereum.
2. **Ethereum Overview:** Ethereum is a decentralized blockchain platform that allows for the creation of decentralized applications (DApps) and the execution of smart contracts.
3. **Execution Environment:** Solidity code is compiled into bytecode and executed on the Ethereum Virtual Machine (EVM).
4. **Cryptocurrency Usage:** Ether (ETH) is Ethereum's native cryptocurrency used to pay for gas fees, which are essential for executing smart contracts.

2.2 Smart Contracts

1. **Definition:** Smart contracts are self-executing programs that run autonomously on the Ethereum network.
2. **Immutability:** They are immutable once deployed, and their code and state are publicly visible on the blockchain.
3. **Use Cases:** Smart contracts can be used for various purposes, such as financial transactions, voting systems, token creation, and more.

2.3 Ethereum Wallets

1. **Purpose:** To interact with Ethereum and deploy smart contracts, you'll need an Ethereum wallet like MetaMask or MyEtherWallet.
2. **Storage of Private Keys:** Ethereum wallets store private keys securely and enable you to send and receive ETH.

3 Tutorial

Step 1: Environment Setup Ensure you have the necessary tools and environment ready for smart contract development:

1. **Install Ethereum Wallet:** Download and install an Ethereum wallet, such as MetaMask, and create an Ethereum account.
2. **Test Ether Acquisition:** Obtain test Ether (ETH) for a testnet (e.g., Rinkeby, Ropsten) to use during development. Test Ether is used to avoid spending real ETH during testing and experimentation.

Step 2: Problem Selection Choose one of the following problems to solve using a Solidity smart contract:

1. **Problem 1: Decentralized Voting System**

   Description: Design and implement a decentralized voting system that allows users to cast votes securely and transparently. The smart contract should prevent double voting and ensure the integrity of the voting process.

2. **Problem 2: NFT Marketplace**

   Description: Create a decentralized NFT (Non-Fungible Token) marketplace where users can mint, buy, and sell NFTs. Implement functions for creating, transferring, and listing NFTs.

3. **Problem 3: Token Rewards System**

   Description: Develop a token rewards system where users can earn and exchange tokens for specific actions or achievements within a DApp. Implement features like token minting, transferring, and redeeming.

4. **Problem 4: Supply Chain Management**

   Description: Build a smart contract for supply chain management, tracking the movement and ownership of goods. Implement functions to update the status and location of items.

Step 3: Smart Contract Development For the selected problem, follow these steps:

1. **Code Implementation:** Write the Solidity code for your smart contract, including state variables, constructor, and relevant functions.
2. **Logic Implementation:** Implement the necessary logic and data structures using Solidity to solve the problem effectively.
3. **Documentation:** Ensure your code includes comprehensive comments and explanations for clarity, adhering to best practices.

Step 4: Testing and Deployment Test and deploy your smart contract:

1. **Remix IDE:** Use Remix or another Solidity development tool to compile and test your smart contract. Test it with various inputs to ensure its functionality.
2. **Deployment:** Deploy your smart contract to a testnet (e.g., Rinkeby) using Remix or a tool like Truffle. Deploying to a testnet allows you to validate your contract without using real Ether.
3. **Interaction:** Interact with your deployed contract through your Ethereum wallet to validate its functionality in a real-world scenario.

Step 5: Submission Submit the following components:

1. **Solidity Code:** Include the complete source code of your Solidity smart contract.
2. **Explanation:** Write a detailed explanation of your smart contract's purpose, functionality, and how it interacts with the Ethereum blockchain. Clearly describe how your contract addresses the selected problem.
3. **Challenges:** Describe any challenges you encountered during development and how you overcame them. Share your insights and lessons learned from overcoming these challenges.
4. **Experience:** Share your overall experience with deploying and interacting with the smart contract on a testnet. Reflect on the process and its significance in the context of blockchain development.
5. **Future Improvements:** If applicable, provide ideas for future improvements or additional features for your smart contract. Consider how it could be enhanced or extended for broader use cases.

This assignment aims to assess your understanding of Solidity and your ability to create functional smart contracts. Make sure your code adheres to best practices and security considerations.

4 Submission regulation

- Students create a folder <Student's ID> containing the contents following:
  - <Code> folder: contains the whole project.
  - Executable file (optinal).
  - <Report.pdf> file(at most 5 pages, single page, 12pt font) that includes:
    - Project structure.
    - Data structure.
    - Any other remarks about your design and implementation.
    - Table of complete features with self-grading.
    - All links and books related to your submission must be mentioned.
    - **DO NOT** insert you source code in the report.

- Compress the above folder into Student's ID.zip for submission.
- Submission with wrong regulation will result in a "0" (zero).
- Plagiarism and Cheating will result in a "0" (zero) for the entire course and will be subject to appropriate referral to the Management Board for further action.

5 Grading (may be changed later)

1. Coding: 7 pts.
2. Report: 5 pts.
3. Total: 12 pts.

Good luck with your lab assignment!