Advanced Smart Contracts - Final Assignment

For this assignment you must create a smart contract or smart contract library using the advanced features you have learned in this course.

Requirements

At your option, choose one of the following two requirement streams:

1. Usage of assembly for non-trivial functionality. This can either be a full smart contract, or a tested library. It must also include benchmarking or gas profiling information.
2. Usage of off-chain computation such as signatures or merkle trees. This should be a tested, full smart contract along with an off-chain component.

Your submission must fulfill all the following requirements:

- Includes tests, using truffle or similar framework.
- Contain a README file that explains the high-level design, implementation details, gas cost optimizations, security considerations, and anything else you think is relevant. Include as much detail as possible.
- You should develop your code in a git repository. When you are ready, submit the public URL to the github (or similar) repository.
- Remember not to include "node_modules" folder in your repository (see https://help.github.com/en/github/using-git/ignoring-files on how to ignore files, e.g. using .gitignore).

Grading

The grading break-down is as follows, out of a total of 40 points:

- 20 points: Fulfills either of the two requirement streams
  - Use of assembly or off-chain computation makes sense
  - Contract and all supporting material is uploaded to github, compiles and deploys correctly
  - More difficult or ambitious projects will earn higher points
- 5 points: Testing
  - Contract is well unit tested using truffle or similar
- 5 points: Documentation
  - Thorough documentation that describes your design
- 5 points: Security
  - Your smart contract should not have any security bugs

    - Documentation that describes the different attacks you have considered, and your contract prevents them
- 5 points: Efficient
    - Your smart contract should not consume any more gas than needed for its functionality
    - Documentation that describes what optimizations you have done, and how you have checked that your contract/library doesn't use excess gas

Ideas (assembly stream)

- Assembly to implement string operations which are not possibly with normal solidity
- Contracts that interact with other contracts (may require assembly)
    - Upgradeable smart contracts
- Custom storage layout (requires assembly to bypass solidity layouts)
    - Diamond layout
- Find a common smart contract operation, and make it more efficient using assembly

Ideas (off-chain stream)

- Efficient token Air drop using merkle trees
- Voting system using off-chain voting roll call
- Ability to validate chunks of a large file on-chain without uploading the file to the chain, just a merkle root
- Off-chain orderbook using ecrecover() to validate orders
- Multi-signature wallet that requires multiple participants to sign a message