Programming Assignment 3

There will be 3 tasks per team for this assignment. Coding 2 out of 3 problems in solidity and analyzing 1 out of 2 codes written in solidity.

Problems:

Problem 1: Distributed Cab Service (mandatory)

Make a solidity smart contract application through which customers can ask for cabs and cab drivers can respond to their requests. Customers can provide their source and destination address to the contract so that the drivers can decide whether to entertain the request or not. Let there be an auction among the drivers through which the smart contract will decide which driver to allocate the customer. The auction will be on the rates provided by the drivers for a particular ride. Once the driver is confirmed, the contract should broadcast this result. You are at liberty to decide the auction rules and type.

Problem 2: Simultaneous Tic-Tac-Toe

Make a two player game using solidity smart contracts where the players play tic-tac-toe on a 3x3 grid. There is a catch though, instead of players playing sequentially, here players will play their moves simultaneously. Like the solidity assignments before, please keep in mind all the security issues. Please keep in mind that your smart contract should offer at least these basic features. Anything extra is left towards your creativity.

Features:

- print the matrix
- take players move
- keep each player's move private from the other player until both have played their moves
- broadcast the updated grid after both players have played their moves

Problem 3: Reverse Average Auction

In this auction, the agents provide their costs to complete a particular task. The beneficiary takes all the input costs and choses the agent whose cost is closest to the average. You can refer to the Simple Auction and Blind Auction given on solidity documentation to understand the dynamics of auction contracts. Note: Reverse Auction is many a times used by Government agencies to distribute resources to the private companies.

Note: It is mandatory to do the first problem statement. You can choose 1 of the 2 other problems as the other problem. Also, you are free to code all three problems, if your other 2 implementations are not good enough, the third code can cover for them.

Code Analysis:

Code 1: (Dutch Auction)[https://gitlab.com/snippets/1707726]

Code 2: (Vickrey Auction)[https://gitlab.com/snippets/1707726]

Analyze one of the two above codes for all kinds of bugs and performance improvements that could be incorporated.