Homework 8

- Follow the instructions very carefully. Answers that do not conform to the instructions will not be given credit.
- Understand thoroughly all the code given to you in this lab. Search for documentation online if there is a primitive or API you have not encountered before.
- Submit only one file for this homework: DutchAuction.sol.
- **Your grade on this homework will be weighted by three regular homeworks. It is very important that you score well on it.**

1. Read about Ethereum smart contract security:
    a. https://consensys.github.io/smart-contract-best-practices/
    b. Your score on the homework will be reduced if you submit insecure code.
2. Start up the vagrant-managed virtual machine included in the homework 8 zip, log into it using ssh. Then, start the Ethereum node server, and then use truffle to test the provided contracts. The tests should run, but most should not pass. Note: You must run "npm install" once under the homework8/auction directory before running truffle. The command installs code packages needed for the project.
3. Read about Dutch auctions: https://en.wikipedia.org/wiki/Dutch_auction
4. Read and thoroughly understand the provided code
    a. contracts/DutchAuction.sol: an incomplete, skeletal implementation of a Dutch auction.
    b. test/<k>_dutch_auction.js: mocha tests for the contract
    c. test/framework.js: a testing harness to make tests easier to write
5. **Part A:** You will write a smart contract that implements a Dutch auction.
    a. The seller instantiates a DutchAuction contract to manage the auction of a single, physical item at a single auction event. The contract is initialized with the following parameters:
        i. reservePrice: the minimum amount of wei that the seller is willing to accept for the item
        ii. numBlocksAuctionOpen: the number of blockchain blocks that the auction is open for
        iii. offerPriceDecrement: the amount of wei that the auction price should decrease by during each subsequent block.
        iv. judgeAddress: used by another part in this assignment.
    b. The seller is the owner of the contract.
    c. The auction begins at the block in which the contract is created.
    d. The initial price of the item is derived from reservePrice, numBlocksAuctionOpen, and offerPriceDecrement: initialPrice = reservePrice + numBlocksAuctionOpen*offerPriceDecrement
    e. A bid can be submitted by any Ethereum externally-owned account.
    f. The first bid processed by the contract that sends wei greater than or equal to the current price is the winner. The wei should be transferred immediately to the sender and the contract should not accept any more bids. All bids besides the winning bid should be refunded immediately.
6. **Part B:** You will enhance the DutchAuction contract so that a judge can be designated for the auction. The judge behaves like an escrow in that he ensures that the seller received the wei of the winning bid if and only if the winner receives the item. A judge is specified by the judgeAddress parameter in the contract constructor. If a judgeAddress is not provided then, the contract should behave exactly as in Part A. However, if a judge is specified, then the behavior of the contract should be enhanced as follows:
    a. The wei of the winning bid is no longer automatically sent to the seller.
    b. Instead, a call to the finalize() function is now required after the winning bid is made, which will send the wei of the winning bid to the seller.
    c. The finalize() function can either be called by the judge or by the winner. The winner is expected to call this function to acknowledge that he received the physical item. If the winner doesn't call it, the judge may intervene and call the finalize() function himself to forcibly send the wei to the seller.

d. If the winner does not call finalize() and the judge also does not call finalize, then the judge can call the refund() function to send the wei of the bid back to the winner. You must ensure that the judge can only call one of these functions once and that the judge cannot send the wei to himself.

7. Notes:
    a. You may assume Ethereum block numbers are sequential and consecutive.
    b. The ganache-cli test utility is configured by default to generate a new block for each transaction submitted to the local Ethereum test network. You may assume this behavior when writing your tests.
    c. Minimal tests are provided for you, but you are required to thoroughly test your work. The tests used to grade your submission will be different than the provided tests.
    d. You may only make modifications to DutchAuction.sol. All other code changes will be ignored during grading.
    e. Truffle may not properly recompile your code unless you first delete the build directory and then run "truffle compile".

Acknowledgement: This homework is based on an assignment designed at Stanford University.