Technical Challenge: Speculator's Auction

Aim: We would like to see how you approach writing smart contracts on Ethereum, and gain some insight into how you think about technical problems. Please complete the coding challenge below, and the follow-up design question.

Time required: ~3-4 hours

Part 1: Auction Contract and ERC20

Please write a simple smart contract system in Solidity that comprises an auction contract and an ERC20 token called RAND.

Let's assume away the economics: there is continuous demand for RAND on the open market, and its price is always non-zero.

Auctions for RAND occur daily. A user must perform a one-time sign-up to participate in auctions: they should fund their auction balance with an ETH amount that will be used for bids, and they should indicate whether they want refunds paid in ETH or RAND. The system should also record their date of sign-up.

The system has one owner, and an upper limit of 1000 users.

Every day, the system mints a random amount of RAND (between 1 and 100 tokens) that go into a pot for auction.

Also once per day, participants can bid, in ETH, for the daily pot of RAND tokens. The user with the highest bid of the day wins the whole pot, but there is a catch: their RAND winnings remain locked in the contract for 30 days after they have been won.

Similarly, each non-winning bid becomes refundable after 30 days from when it was made. The bid is refunded in the currency the user chose when they signed up. If they chose RAND, the system mints them an amount of RAND equal in value to their ETH bid, and burns their ETH bid.

We can pretend the contract is already connected to a price oracle - you may use a dummy constant to represent the latest ETH:RAND price.

Once per year, the owner draws revenue from the system, and takes a 10% cut of all the RAND winnings locked in the system at that point in time.

Please keep security and gas efficiency in mind overall.

Optional feature (if time permits): at any time, the owner should be able to stop the auction process, and initiate a shutdown that ends the auctions forever. Please implement this in a way that is fair to all users.

Part 2: Outline a gas compensation mechanism (no coding required)

The owner would like to compensate users for their gas costs, so that the system is free to use for everyone.

Please outline a design for a gas compensation mechanism: a way for the system to reimburse users for their gas costs

How could you make gas compensation decentralized?

Please give reasons for the design choices you make. What challenges do you come across?

Thank you! We are looking forward to reading your code and answer.