Web3 Solidity Developer Interview Assignment

Objective:

We want you to help implement a set-like structure in Solidity (using hardhat to help out) that only allows a certain amount of "elements" in the set. Once you've created the Solidity contract, we want you to test it in TypeScript, not in Solidity.

The CappedSet contract:

Each "element" in the set is a pair: an address and a value.

When the set gets too big (i.e., it reaches its max), it should boot out the element with the lowest value.

Its constructor takes an argument numElements that represents the maximum number of elements that can be in the set.

There are four methods that we would like you to implement in CappedSet.sol:

insert(address addr, uint256 value) returns (address newLowestAddress, uint256 newLowestValue)
This method should add a new element (addr and value) and return the new element with the lowest value, which might be itself. If this is the first element being inserted, this method should return (0,0).

update(address addr, uint256 newVal) returns (address newLowestAddress, uint256 newLowestValue)
This method should update the existing element with address addr, if it exists, and return the new element with the lowest value, which might be itself.

remove(address addr) returns (address newLowestAddress, uint256 newLowestValue)
This method should delete the existing element with address addr, if it exists, and return the new element with the lowest value.

getValue(address addr) returns (uint256)
Retrieves the value for the element with address addr, if it exists.

You're free to design the contract however you want as long as it follows the above behaviour. You can add as many contract members/fields as you wish. You can also use any open-source solidity libraries such as openzeppelin.

Testing the contract:

For interacting with the contracts, you can use hardhat. Test as much as you think is necessary.

Remember to use TypeScript and not Solidity to test the contracts.

What we're looking for:

We're interested in your coding style, your familarity with smart contract development, and your JavaScript/TypeScript proficiency. Bonus points for low gas usage!

How to complete this challenge:

Send us the GitHub link of repo when you're done. Also share the contract address of a deployed and verified CappedSet contract on any of the test networks.

Good luck!