Answers - Homework 1 & 2

1. How does consensus affect smart contract design ?
   Differences that may affect the design of your contract

- Block time
- Block gas limit
- Time to finality
- The feasibiity of MEV

1. DogCoin

   - naive approach, we will refine later
   - can inherit from Open ZeppelinERC20 (you can even use the smart contract wizard : https://wizard.openzeppelin.com/)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract DogCoin is ERC20 {
        constructor() ERC20("DogCoin", "DOG") {}
}
```

   - For the holders, use an array (address), and add the array from the transfer (or_transfer) function

```
address[] public holders;
...
        holders.push(_holder);
```

   but we don't want dupes in our array, so we either

   - need to loop through the array and check the address doesn't already exist
   - or check that it is a new holder before we add it (i.e. that the balance is zero before the transfer)
   - removing from the array
     we want to remove items but not leave gaps,
     Remember that deleting an element doesn't remove it, it just sets it to the default

value
Since we do not care about the order of elements in our array, we can use the 'swap and pop' approach. (you can see this in for example OZ EnumerableSet)
Here is a simplified version of that.

```
        function testRemove(uint256 _index) public {
                if (_index != 0) {
                        uint256 lastIndex = holders.length - 1;
                        if (lastIndex != _index) {
                                holders[_index] =
holders[lastIndex];
                        }
                }
                holders.pop();
        }
```

But if we use this approach, be aware that the order of elements in the array will change.
An alternative approach is to copy from an old array to a new array.

Homework 2 - question 3
Generally speaking we cannot create this functionality completely within a contract, but there are some projects based on the incentive execution pattern (see lesson 3), such as theh Ethereum alarm clock, or Chainlink keepers.