Creating a Lottery

Your assignment is to create a lottery smart contract using Solidity and the in-browser IDE called Remix (https://remix.ethereum.org). This document includes a guide to building the smart contract and includes all the functions you will need. Let's get started!

Outline

The lottery contract should:

- Allow players to join the lottery
- Require players to send in 1 Ether to contribute to the prize pool
- Have a manager that can end the lottery
- Pick a winner and send them the prize pool
- 'Reset' the contract so another lottery can begin

Getting started

Open remix and click the plus (+) button on the upper left side of the screen to create open a new file, name it "lottery" and click ok.

Now we need to tell the IDE which version of Solidity we're working with. On the first line type

```
pragma solidity ^0.4.22;
```

Move down to the next line to create the contract (class)

```
contract Lottery{
    //Code goes here
}
```

Our contract has now been created. Let's declare some variables we will be using throughout the contract.

In the outline, we said we wanted to allow players to join the lottery, we need to keep track of them. The best way to do this is through an array. We will use their addresses to identify them.

```
//Creating an array of addresses called players
//Leaving the brackets empty makes the array size dynamic
address[] public players;
```

We also want to have a manager for the contract. The manager will be the only one that can end the lottery. Ending the lottery will mean that the contract picks a winner and resets for another lottery. We can identify the manager by their address.

```
//Manager of the contract
address public manager;
```

Time to make the constructor for the contract. Constructors should be identified using

```
constructor()  contractName() public {
        //Code goes here
}
```

The constructor will be called when the contract is created. This is the best time to assign the manager. Let's make the contract creator the manager of the contract.

When contracts are deployed, it requires someone (an address) to deploy it. This is a transaction on the network. Solidity has a built in feature to see the address that sent the transaction.

Using msg.sender we can record who sent a transaction.

In the constructor let's assign the contract creator as the manager

```
manager = msg.sender;
```

We will also be using msg.sender to add players to the players array.

I will now stop giving you all of the code, try to figure it out! I will give directions, and if you get stuck, scroll down to the end to find the exact code.

Let's create a function called enter that allows players to join the lottery.

The function should be public, and payable.

Players wanting to join the lottery will call this enter function from their account (address). They need to send at least 1 ether to join the lottery.

Solidity has a function called require() that we can use to check something, sort of like an if statement. You can use ==, <, >, !=, >=

To check that the player sent at least 1 ether to join we can use msg.value and require()

msg.value is the amount of ether that was sent with the transaction

```
require(msg.value >= 1 ether);
```

This ether gets stored in the smart contract.

After the require statement, let's add the new player to the players array.

This can be done using .push() function

```
arrayName.push(address);
```

You can use msg.sender to represent the address of the player that wants to join.

That's it! Players can now join our lottery and they are added into the players array.

Next we will create a pseudo random number generator that will be used to pick a winner of the lottery.

Note: This is a somewhat predictable number and uses publicly available data as parameters. This would not be acceptable if we were working with real value.

Let's create a function called random. The function should be private, should be of view type since it won't modify any data, and should return a uint.

We will use some parameters as inputs and hash all the data to give us a number. In the function put the following

```
return uint(keccak256(block.difficulty, now, players));
```

That will return a number we can use to help pick the lottery winner.

block.difficulty is the current mining difficulty of the blockchain
now is the time whenever the function is called
players is the array of addresses in our contract
keccak256 is a hashing function similar to SHA-3

That's all we need to do for pseudo random number generation.

Now we will create the function to pick the winner of the contract. The manager does not pick the winner, they just call the function that will pick the winner. This function should be public and not return anything.

This function should require that only the manager can call it. Use require(), msg.sender, and our manager variable to check this.

The function should then pick the winner. This can be done using our random number function, modulo (%), and the players array length.

Array length can be found using arrayName.length

We should assign the result to a variable that holds the index of the winning address.

```
uint index = //Some code
```

Assign this variable the result of random() modulo the array length.

This will create a value between 0 and the array length. This number will be the array index of the winner.

We've now picked a winner and have their array index, let's send them their winnings!

Using the .transfer(this.balance) function we can send the address the prize pool.

```
address.transfer(this.balance);
```

Using the index variable and players array we can represent the address of the winner.

Almost done, we've picked a winner and sent them their prize, what's left?

We should clear the players array so that the lottery can start over.

This can be done by making the players variable a new address array.

```
players = new address[](0);
```

The (0) is there to initialize the array with a length of zero. The array is still dynamic though.

Congratulations, you just created a working lottery smart contract!

Contract code is below. Please only look at this if you are stuck and can't figure out how to write a piece of code.

```
pragma solidity ^0.4.22;

contract Lottery{

    address[] public players;
    address public manager;

    constructor() Lottery() public {

        manager = msg.sender;
    }

    function enter() public payable {

        require(msg.value >= 1 ether);

        players.push(msg.sender);
    }

    function random() private view returns (uint) {

        return uint(keccak256(block.difficulty, now, players));
    }

    function pickWinner() public {

        require(msg.sender == manager);

        uint index = random() % players.length;

        players[index].transfer(this.balance);

        players = new address[](0);
    }

}
```