Lottery

Core Idea

The core idea is to distribute the lottery mechanism in phases with each phase being implemented using a smart contract. We define a few states in the lottery to ensure fairness in the process: OPEN, CLOSED, FINISHED, TERMINATED. It prevents the owner and users from executing functions that cannot be executed in that particular state.

User Tickets phase

The user can choose any number as he pleases (which lies in the domain of uint256) and enter that number by paying a certain small fee into the contract (decided by the organizer). With the entry of that number, the user essentially gets the lottery ticket which is marked by the number entered by him.

A user cannot enter a number that has previously been entered by another user before him because that lottery ticket has already been allocated (similar to a first come first serve scenario). It is similar to the serial number of a lottery. If the serial number is taken, we cannot do anything. We would need to take another one. The blockchain stores each smart contract that takes the user number and allocates the ticket to him.

A user can enter multiple numbers by paying the fees multiple times with the only condition that all the numbers have to be different. It is like buying multiple lottery tickets to increase the chance of winning.

The money will be paid to the owner of the lottery from the person who buys the lottery.

The number chosen by a user is kept secret and can only be accessed by that user as the hash of the smart contract is visible to others and not the actual link of the number with the user. In this way, a generic user can only see all the possible numbers taken when he wants to choose his number and not which user has taken a particular number.

In this phase the lottery is OPEN.

Evaluation phase

In this phase the lottery is CLOSED.

The blockchain has stored all the numbers entered by millions of users who have taken their lottery tickets and paid to the owner.

All the numbers are stored in a very large array of smart contracts in order of the tickets sold (can be verified by the timestamp of the initial smart contract).

A random number generator can generate a number between the first and last index of the large array of numbers. The generated number is verified that lies within the range of the array indices.

It terminates with the FINISHED state as all lottery computations are completed.

Winner announcement

In this phase, the lottery is FINISHED.

The generator valid number is taken and is mapped to the index in the array. With the numerical value, the user details are stored.

Since all numbers are distinctly similar to the serial number distinction, the random number will pick a distinct value, so there will be a single winner only.

The winning lottery money is transferred from the owner to the winning user through a normal smart contract money transfer.

It terminates with the TERMINATED state as the winners have been announced.

We also implement a change state mechanism to prevent changes that are prohibited which ensures that no action is taken to spoil the lottery system. The only valid state change is from OPEN to CLOSED to FINISHED to TERMINATED. All other state changes are invalid and need to be disregarded.

Smart Contract Implementation

```
// Create a smart contract
contract Lottery{

   // Specified fee value
   uint FEES = OWNER_SPECIFIED_FEES;
   uint WIN_MONEY = OWNER_SPECIFIED_WINNING_FEES;

   // To store the winning index
   uint256 ans;

   // OPEN = 1, CLOSED = 2, FINISHED = 3, TERMINATED = 4
   // Only valid order is 1 -> 2 -> 3 -> 4
   // Initially Lottery is OPEN
   uint Lotterystate = 1;

   // Contains an array pair, number and user address
   store = [];

   // Function to give ticket to user
   function ticket(uint fees, uint256 num, address user){

      // Requirements

      // Lottery must be OPEN to give tickets
      require Lotterystate == 1;
```

```
    // The user must pay the full fees
    require fees == FEES;

    // The input number must be valid
    require num NOT present in the first value of pairs in store array;

    // Give ticket
    store.push((num, user));

    // Payment
    payable(owner().transfer(fees));
}

// Get the winner evaluation
function evaluation(){

    // As the evaluation has started, no more tickets can be given
    // Lottery is CLOSED
    changestate(2);

    // Lottery must be CLOSED to evaluate the winner
    require Lotterystate == 2;

    // Generate an integral random number using any function within the range
    ans = generaterandomnumber(0, store.length);

    // Verification within the indices
    require 0 <= ans <= store.length;

    // As the index for the winner has been computed,
    // lottery is FINISHED
    changestate(3);
}

// Pay the money to the winner
function winner(uint256 ans){

    // Lottery must be FINISHED to pay the winner
    require Lotterystate == 3;

    // Obtain the winner from index
    num, user = store[ans];

    // Payment to the winner
    payable(user.transfer(WIN_MONEY));
```

```
        // Winner is displayed
        return user;

        // As the winner has been announced,
        // lottery is TERMINATED
        changestate(4);
    }

    // Valid change of the Lottery state
    function changestate(uint state){

        // Either we can move 1 -> 2 or 2 -> 3 or 3 -> 4
        require (Lotterystate == 1 && state == 2) ||
                (Lotterystate == 2 && state == 3) ||
                (Lotterystate == 3 && state == 4);

        // Change the current Lottery state
        Lotterystate = state;
    }
}
```