CMPE 483 Blockchain Programming

Homework-1, Spring 2022

(This homework can be done in groups of at most 3 students)

(due May 2nd)

In this project, you will implement an autonomous decentralized lottery. One lottery round lasts two weeks and consists of two stages. A new lottery round starts right after the previous lottery is completed. A lottery ticket costs 10 TL. You should:

1. Use OpenZeppelin ERC20 token contract to implement TL tokens,
2. Use OpenZeppelin ERC721 token contract to implement tickets. Each ticket will be represented as a token (i.e. as an NFT). The tickets will also be transferable to others.

OpenZeppelin contracts are available at: https://github.com/OpenZeppelin/openzeppelin-contracts .

Winner tickets will be selected by computing random numbers that determine each winner ticket. A random number is to be supplied by the ticket purchasers. The lottery should employ (i) ticket purchase and random number submission (ii) random number reveal stages. The details of how a random number can be generated in order to determine winners is given here:

https://ethereum.stackexchange.com/questions/191/how-can-i-securely-generate-a- random-number-in-my-smart-contract

The stages of each lottery round are scheduled as follows in an overlapping manner:

a) Ticket purchase and random number submission stage : 4 days.
b) Random number reveal stage : 3 days. Note that if previously submitted random numbers are not submitted correctly in the reveal stage, the chance of winning is lost. Half of the money can be refunded if collectTicketRefund function is called.

<table>
<tr><td>Purchase/ Random No. Submission (4 days)</td><td>Reveal (3 days)</td><td>Purchase/ Random No. Submission (4 days)</td><td>Reveal and Propose (3 days)</td><td>…</td><td>Purchase/ Random No Submission (4 days)</td><td>Reveal (3 days)</td></tr>
<tr><td colspan="2">Lottery 1</td><td colspan="2">Lottery 2</td><td>…</td><td colspan="2">Lottery n</td></tr>
</table>

Let *M* be the amount money collected from the sale of tickets at the current lottery. The *i*th prize P_i will be awarded to the winners as follows:

$$P_i = \lfloor M/2^i \rfloor + (\lfloor M/2^{i-1} \rfloor \mod 2) \qquad i = 1,...,\lceil log_2(M) \rceil + 1$$

Note that a winning user should be able to withdraw his prize anytime after the lottery round ends. Also, it is possible that a ticket may win more than one prize.

The lottery implementation should provide the following interface to the external world:

```
function depositTL(uint amnt) public
function withdrawTL(uint amnt) public
function buyTicket(bytes32 hash_rnd_number) public
function collectTicketRefund(uint ticket_no) public
function revealRndNumber(uint ticketno, uint rnd_number) public
function getLastOwnedTicketNo(uint lottery_no) public view returns(uint,uint8 status)
```

function getIthOwnedTicketNo(uint i,uint lottery_no) public view returns(uint,uint8 status) function checkIfTicketWon(uint ticket_no) public view returns (uint amount) function collectTicketPrize(uint ticket_no) public function getIthWinningTicket(uint i, uint lottery_no) public view returns (uint ticket_no,uint amount) function getLotteryNo(uint unixtimeinweek) public view returns (uint lottery_no) function getTotalLotteryMoneyCollected(uint lottery_no) public view returns (uint amount)

Note that Money (TL) must be deposited to the lottery contract. The TL balance of each user must be kept. When a user wants to purchase a ticket, the user will pay for it by deducting TL from his/her balance in Lottery contract.

You can also implement other additional functions as you like.

.

Grading

Your project will be graded according to the following criteria:

Documentation (written document describing how you implemented your project and also showing the correctness of your implementation). You should also provide average gas usages for the interface functions.	30%
Comments in your code	10%
Correctly functioning Solidity code, test scripts, and tests.	60%

Homework Submission:

Please submit your project to Moodle. Before you submit your project, please timestamp (notarize) your project zip file at https://certify.bloxberg.org/ Do NOT lose the certification and the submitted project zip file. The certification is a proof that your project zip file existed during the time of submission.

Please also answer the following questions and submit it with your project.

Task Achievement Table	Yes	Partially	No
I have prepared documentation with at least 6 pages.			
I have provided average gas usages for the interface functions.			
I have provided comments in my code.			
I have developed test scripts, performed tests and submitted test scripts as well documented test results.			
I have developed smart contract Solidity code and submitted it.			
Function depositTL is implemented and works.			
Function withdrawTL is implemented and works.			
Function buyTicket is implemented and works.			
Function collectTicketRefund is implemented and works.			
Function revealRndNumber is implemented and works.			
Function getLastOwnedTicketNo(uint lottery_no) is implemented and works.			
Function getIthOwnedTicketNo is implemented and works.			
Function checkIfTicketWon is implemented and works.			
Function collectTicketPrize is implemented and works.			
Function getIthWinningTicket is implemented and works.			
Function getLotteryNo is implemented and works.			

Function getTotalLotteryMoneyCollected(uint lottery_no) is implemented and works.			
Function depositTL is implemented and works.			
Function withdrawTL is implemented and works.			
Function buyTicket is implemented and works.			
Function collectTicketRefund is implemented and works.			
Function revealRndNumber is implemented and works.			
Function getLastOwnedTicketNo is implemented and works.			
Function getIthOwnedTicketNo is implemented and works.			
Function checkIfTicketWon is implemented and works.			
I have tested my smart contract with 5 addresses and documented the results of these tests.			
I have tested my smart contract with 10 addresses and documented the results of these tests.			
I have tested my smart contract with 100 addresses and documented the results of these tests.			
I have tested my smart contract with 200 addresses and documented the results of these tests.			
I have tested my smart contract with more than 200 addresses and documented the results of these tests.			