CMPE 483 Sp. Top. in CMPE Blockchain Programming, Fall 2024
Homework 1 (due Nov. 25)

(Project can be done in groups of up to 3)

In this project, you will implement a multi-lottery project for a company in the form of a smart contract (called CompanyLotteries). Once a lottery is created by the company by entering it into the CompanyLotteries smart contract, the lottery should proceed autonomously. In order to create each lottery, the following information will be entered:

1. Unixtime for the end of the lottery (start time will be submission time implicitly),
2. Number of tickets issued,
3. Number of winners,
4. Minimum percentage of the tickets that need to be purchased in order to let the lottery run. If minimum percentage is not reached, lottery will be canceled and tickets payments will be refunded to the users (by letting each user withdraw the refund).
5. Price of each ticket (in terms of payment tokens which is assumed to be deployed as an ERC20 contract).
6. Hash of the contents of the html page that describes the lottery.
7. URL of the html page that describes the lottery.

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 (ii) random number submission 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 lotteries will be identified by consecutive numbers (i=1,2,...). Each lottery i will have to distinct phases (a) purchase and (b) reveal. The duration of each of purchase and reveal stages will be half of the total duration of the lottery. The stages of each lottery round are scheduled as follows

a) Ticket purchase and random number submission stage.

b) Random number reveal stage : Note that if previously submitted random numbers are not submitted correctly in the reveal stage, the chance of winning is lost. Also, no ticket refund is made in this case.

Lottery i	Purchase/ Random No Submission	Reveal

At the end of each lottery, the company will be able to withdraw the proceeds from the lottery, if the lottery is not cancelled.

Some additional notes:

- Tickets are non-transferrable,
- Users can purchase as many tickets as they like. But in each purchase transaction, they can purchase at most 30 tickets.

The interface functions of the CompanyLotteries contract are as follows:

```
function createLottery(uint unixbeg, uint nooftickets, noofwinners, uint minpercentage,
                                   uint ticketprice, bytes32 htmlhash, string url) public  onlyOwner
                                   returns(uint lottery_no)
function buyTicketTx(uint lottery_no,uint quantity,bytes32 hash_rnd_number)
                                   public returns(uint sticketno)
function revealRndNumberTx(uint lottery_no,uint sticketno, quantity, uint rnd_number)  public
function getNumPurchaseTxs(uint lottery_no)  public view returns(uint numpurchasetxs)
function getIthPurchasedTicketTx(uint i,uint lottery_no)  public view
                                            returns(uint sticketno, uint quantity)
function checkIfMyTicketWon(uint lottery_no, uint ticket_no)  public view returns (bool won)
function checkIfAddrTicketWon(address addr, uint lottery_no, uint ticket_no)
                                           public view returns (bool won)
function getIthWinningTicket(uint lottery_no,uint i)  public view returns (uint ticketno)
function withdrawTicketRefund(uint lottery_no, uint sticket_no)  public
function getCurrentLotteryNo()  public view returns (uint lottery_no)
function withdrawTicketProceeds(uint lottery_no)  public onlyOwner
function setPaymentToken(address erctokenaddr) public onlyOwner
function getPaymentToken(uint lottery_no) public returns (address erctokenaddr)
function getLotteryInfo(uint lottery_no) public returns (uint unixbeg, uint nooftickets,
                                     noofwinners, uint minpercentage, uint ticketprice)
function getLotteryURL(uint lottery_no) public  returns(bytes32 htmlhash, string url)
 function getLotterySales(uint lottery_no) public (uint numsold)
```

You can also implement other functions as you like.

OpenZeppelin ERC20 token contracts can be used to implement payment tokens: https://openzeppelin.com/contracts/ .

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.

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 createLottery is implemented and works.			
Function buyTicketTx is implemented and works.			
Function revealRndNumberTx is implemented and works.			
Function getNumPurchaseTxs is implemented and works.			
Function getIthPurchasedTicketTx is implemented and works.			
Function checkIfMyTicketWon is implemented and works.			
Function checkIfAddrTicketWon is implemented and works.			
Function getIthWinningTicket is implemented and works.			
Function withdrawTicketRefund is implemented and works.			
Function getCurrentLotteryNo is implemented and works.			
Function withdrawTicketProceeds is implemented and works.			
Function setPaymentToken is implemented and works.			
Function getPaymentToken is implemented and works.			
Function getLotteryInfo is implemented and works.			
Function getLotteryURL is implemented and works.			
Function getLotterySales is implemented and works.			
I have tested my smart contract with 100 user addresses and documented the results of these tests.			
I have tested my smart contract with 200 user addresses and documented the results of these tests.			
I have tested my smart contract with 300 user addresses and documented the results of these tests.			
I have tested my smart contract with more than user 300 addresses and documented the results of these tests.			